Tyler Bohan

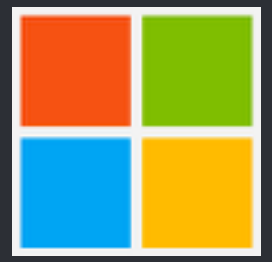
HushCon West 2018

IOKit Revisited
UAF in OSX Graphics

Introduction

- Tyler Bohan
  - Senior Research Engineer
  - Cisco Talos






- Talos Vulndev
  - Third party vulnerability research
    - 170 bug finds in last 12 months
      - Microsoft
      - Apple
      - Oracle
      - Adobe
      - Google
      - IBM, HP, Intel
      - 7zip, libarchive, NTP
  - Security tool development
    - Fuzzers, Crash Triage
  - Mitigation development
    - FreeSentry

TALOS

Why?

Used in most (all?) iOS jailbreaks

Nearly all pwn2own entries against OSX utilize IOKIT

- Since June IOKIT patched 6(+) —
  - CVE-2018-4402 CVE-2018-4341
  - CVE-2018-4354 CVE-2018-4341
  - CVE-2018-4354 CVE-2018-4383

TALOS

RANT

- Multi[illegible]e!

Vulns submitted in June

Patched Mojave in October

e patches vulnerabilities silently and alerts the
months later

Patched High Sierra in December

TALOS

Objective

- Understand how a user space program may interact with the kernel
- Introspect communications between an application and the kernel
- Extract information at run time to make analysis simpler

Why?

- Accessed by every rendering process on OSX
- Including from inside a Sandbox!

SAFARI SANDBOX PROFILE

```
(allow iokit-open
       (iokit-connection "IOAccelerator")
       (iokit-registry-entry-class "IOAccelerationUserClient")
       (iokit-registry-entry-class "IOSurfaceRootUserClient")
       (iokit-registry-entry-class "IOSurfaceSendRight"))
(allow iokit-open
       (iokit-registry-entry-class "IOFramebufferSharedUserClient"))
(allow iokit-open
       (iokit-registry-entry-class "AppleIntelMEUserClient")
       (iokit-registry-entry-class "AppleSNBFBUserClient"))
(allow iokit-open
       (iokit-registry-entry-class "AGPMClient")
       (iokit-registry-entry-class "AppleGraphicsControlClient")
       (iokit-registry-entry-class "AppleGraphicsPolicyClient"))
(allow iokit-open
       (iokit-registry-entry-class "AppleMGPUPowerControlClient"))
```

TALOS

What is it again?

- I/O Kit is a collection of system frameworks, libraries, tools, and other resources for creating kernel extensions in OS X
- restricted form of C++
- multiple levels of inheritance
- registers its own methods to handle user interaction

TaLOS

What is it again?

- TLDR; Direct access into the kernel from user-space code
- Nearly all system applications interact with IOKit at some level

How do I access it?

- IOKit drivers register themselves in the IORegistry
- device drivers can expose an interface to user space by implementing a UserClient
- all access through the IOKit comes through IOKit master port (IOServiceOpen)
- all calls go through IOConnectCall variant

```
3  // Look up a registered IOService object whose class is AppleLMUController
4  serviceObject = IOServiceGetMatchingService(kIOMasterPortDefault,\
5                  IOServiceMatching("AppleLMUController"));
6
7
8  // Create a connection to the IOService object
9  kr = IOServiceOpen(serviceObject, mach_task_self(), 0, &dataPort);
10 IOObjectRelease(serviceObject);
11
12
13 //Call Into IOKIT
14 kr = IOConnectCallScalarMethod(
15      dataPort,
16      kSetLEDBrightnessID,
17      inputValues,
18      inputCount,
19      outputValues,
20      &outputCount
21 );
22
```

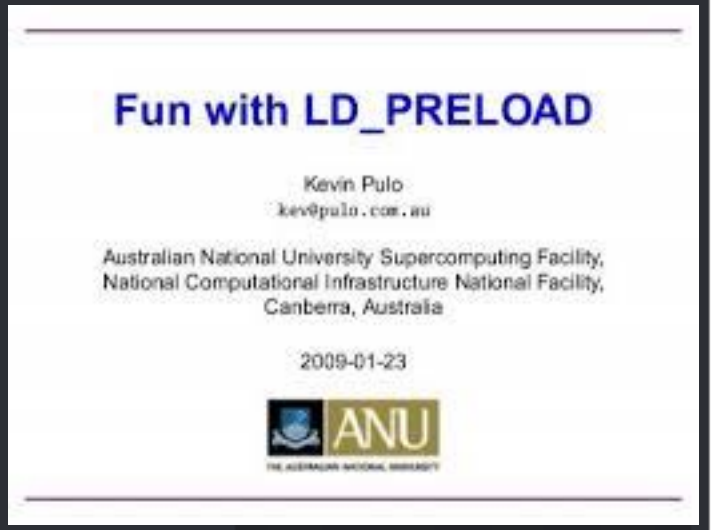
Service Matching Name

Get Connection Port

Call into Kernel

How do I access it?

- Library interposing - DYLD_INSERT_LIBRARIES
  - similar to LD_PRELOAD
- Allows dynamic link time function interposing
- Watch how programs interact with IOKit



Fun with LD_PRELOAD

Kevin Pulo
kev@pulo.com.au

Australian National University Supercomputing Facility,
National Computational Infrastructure National Facility,
Canberra, Australia

2009-01-23

ANU

How do I access it?

- Lets build a
- Interposing is inherently confusing
- Learn the in

Talos

```
kern_return_t
fake_IOServiceOpen(
  io_service_t service,
  task_port_t owningTask,
  uint32_t  type,
  io_connect_t  * connect )
{
  kern_return_t ret = IOServiceOpen(service, owningTask, type, connect);

  io_name_t className;
  IOObjectGetClass(service, className);
  printf("IOServiceOpen(service=%x, owningTask=%x, type=%x, connect=%p)
         (*connect after call = %x)\nclassName %s\n", \
          service, owningTask, type, connect, *connect,c
  return ret;
}

kern_return_t
fake_IOConnectCallMethod(
  mach_port_t connection,
  uint32_t    selector,
  uint64_t   *input,
  uint32_t    inputCnt,
  void       *inputStruct,
  size_t      inputStructCnt,
  uint64_t   *output,
  uint32_t   *outputCnt,
  void       *outputStruct,
  size_t     *outputStructCntP)
{

  printf("[>] IOConnectCallMethod connection %x SELECTOR %d\n",connection,selector);
  return IOConnectCallMethod(
```

Open Connection

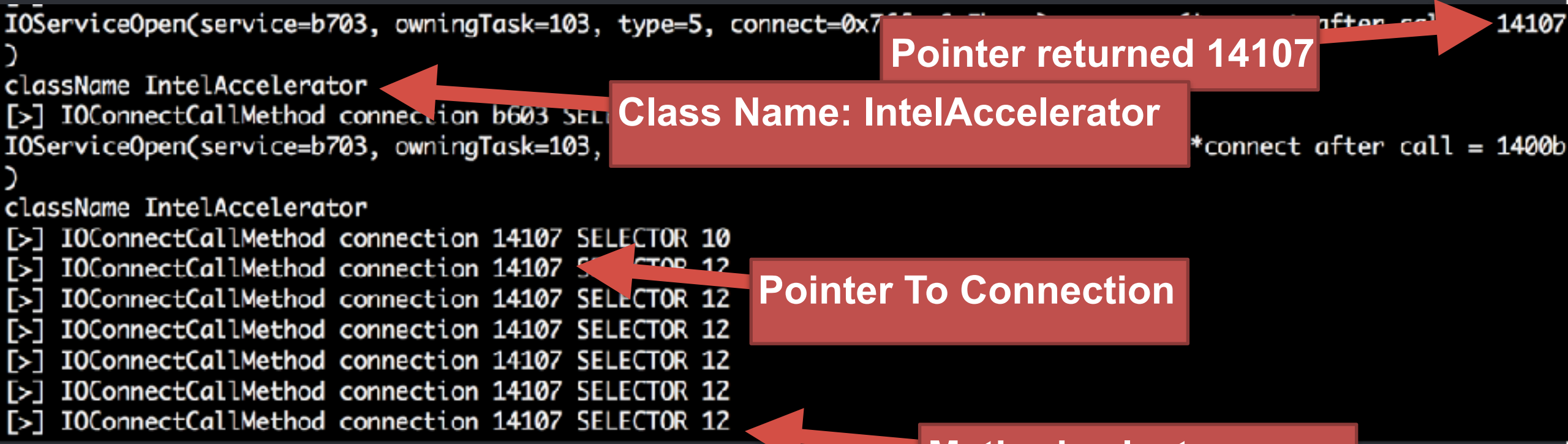
Extract Service Name

IOConnectCall* multiple variants

Credit: Ian Beer - 2014

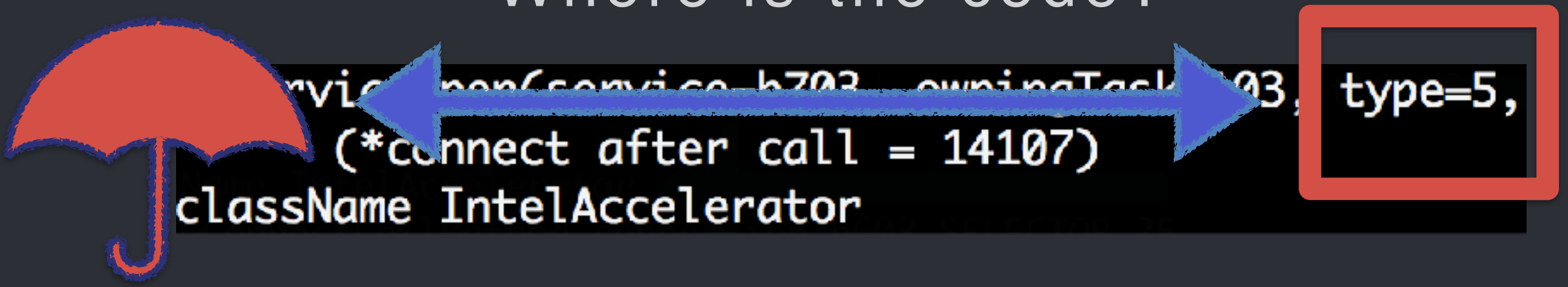
TALOS

Introspection Tool



TALOS

Where is the code?



- IntelAccelerator is an umbrella name holding multiple classes
- Class is determined by type variable
- IORegistry stores information to access type

Where is the code?

@s1guza

```
IOServiceOpen(service=b703, owningTask=10, type=5,
        (*connect after call = 14107)
className = IntelAccelerator
```

```
Name                Type Spawn
IntelAccelerator       0 (os/kern) successful
IntelAccelerator       1 (os/kern) successful
IntelAccelerator       2 (os/kern) successful
IntelAccelerator       3 (iokit/common) unsupported functio
IntelAccelerator       4 (os/kern) successful
IntelAccelerator       5
IntelAccelerator       6 (os/kern) successful
IntelAccelerator       7 (os/kern) successful
IntelAccelerator       8 (os/kern) successful
IntelAccelerator       9 (os/kern) successful
```

```
IGAccelSurface
IGAccelGLContext
IGAccel2DContext

IOAccelDisplayPipeUserClient2
IGAccelDevice
IGAccelSharedUserClient
IOAccelMemoryInfoUserClient
IGAccelCLContext
```

https://github.com/Siguza/iokit-utils

Talos

Building a Tool

What do we know?

- Class name of the device we are talking to
  - **IntelAccelerator**
- Type name using tooling and provided type number
  - **5** (**IGAccelDevice**)

Building a Tool

What do we have?

- Dynamic library written in C then injected into process at launch
- C can be cumbersome, slow to develop, error prone etc
- **Idea? Scripting!**

IntelAccelerator = 14103

Before: IOConnectCallMethod connection **14103**

After: IOConnectCallMethod connection **IntelAccelerator**

- Dynamic instrumentation toolkit
- Written using Javascript
- Ability to introspect dynamically rather than only at load time


```
➜  code DYLD_INSERT_LIBRARIES=flip.dylib /Applications/Chess.app/Contents/MacOS/Chess

dyld: could not load inserted library 'flip.dylib' because image not found

[1]    42120 abort      DYLD_INSERT_LIBRARIES=flip.dylib /Applications/Chess.app/Cont
```

- System Integrity Protection
  - Protected file cannot be modified before load

- Frida injects at runtime — no checks -_-

TALOS

```
39  /Handles taken from IORegistry
40  var handles = { "IntelAccelerator":["IGAccelSurface","IGAccelGLContext",\
41              "IGAccel2DContext","NA","IOAccelDisplayPipeUserClient2",\
42              "IGAccelDevice","IGAccelSharedUserClient","IOAccelMemoryInfoUserClient",\
43              "IGAccelCLContext","IGAccelCommandQueue"] }
44
45  Interceptor.attach(Module.findExportByName("IOKit", "IOServiceOpen"), {
46      onEnter: function(args) {
47          // console.log("IOServiceOpen called");
48          connect_ptr = args[3]; // io_connect_
49          classname = Memory.alloc(256);
50          // Determine the class name of the IO
51          var IOObjectGetClass = Module.findExportByName(null, "IOObjectGetClass");
52          var IOObjectGetClassFunc = new NativeFunction(pt                'int', [po
53          IOObjectGetClassFunc(args[0], classname);
54          type = args[2];
55      },
56      onLeave: function(retval) {
57          // If we have a valid connection
58          if (retval == 0) {
59              var handle = Memory.readU32(connect_ptr);
60              var userclient = Memory.readUtf8String(classnam
61              service_ids[handle] = [userclient,type];
62          }
63      }
64  });
65
```

Library and function to hook

Memory Storage

Retrieve ptr to library function

Call library func

Retrieve previous

Dictionary storage for later use

```
kern_return_t
fake_IOServiceOpen(
  io_service_t service,
  task_port_t owningTask,
  uint32_t  type,
  io_connect_t  * connect )
{
  kern_return_t ret = IOServiceOpen(service, owningTask, type, connect);

  io_name_t className;
  IOObjectGetClass(service, className);
  printf("IOServiceOpen(service=%x, owningTask=%x, type=%x, connect=%p)
         (*connect after call = %x)\nclassName %s\n", \
          service, owningTask, type, connect, *connect,c
  return ret;
}

kern_return_t
fake_IOConnectCallMethod(
  mach_port_t connection,
  uint32_t    selector,
  uint64_t   *input,
  uint32_t    inputCnt,
  void       *inputStruct,
  size_t      inputStructCnt,
  uint64_t   *output,
  uint32_t   *outputCnt,
  void       *outputStruct,
  size_t     *outputStructCntP)
{

  printf("[>] IOConnectCallMethod connection %x SELECTOR %d\n",connection,selector);
  return IOConnectCallMethod(
```

Open Connection

Extract Service Name

IOConnectCall*
multiple variants

TALOS

```
66  Interceptor.attach(Module.findExportByName("IOKit", "IOConnectCallMethod"), {
67      onEnter: function(args) {
68          total += 1;
69
70          var conn = args[0].toInt32();
71          var sel = args[1].toInt32();
72          var client = ""
73          var type = ""
74
75          try {
76              client = service_ids[conn][0]
77              type = service_ids[conn][1]
78          }catch (e) {
79              client = "UNK"
80              type = "UNK"
81          }
82          var new_type = ""
83          var new_sel = ""
84          try{
85              for (var i in handles){
86                  if(type == "UNK") break;
87                  var tipe = parseInt(type,16);
88                  if(i == client)
89                      var new_type = handles[i][tipe]
90              }
91          } catch (e){}
92
93          if(new_sel) sel = new_sel;
94          if(new_type) type = new_type;
95
96          console.log("[IOConnectCallMethod # " + total + "] "+ client + " TYPE "
97                      + type +" SELECTOR " + sel)
98      }
```

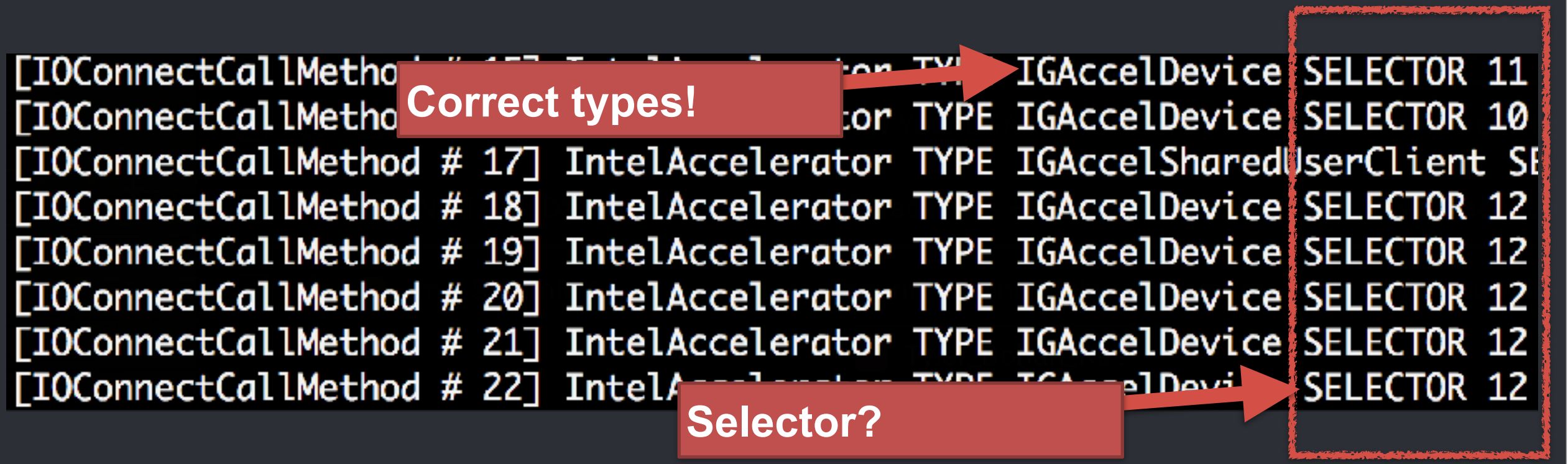
Library and function to hook

Grab arguments

Query dictionary for proper type

Output information with correct client

Putting it Together

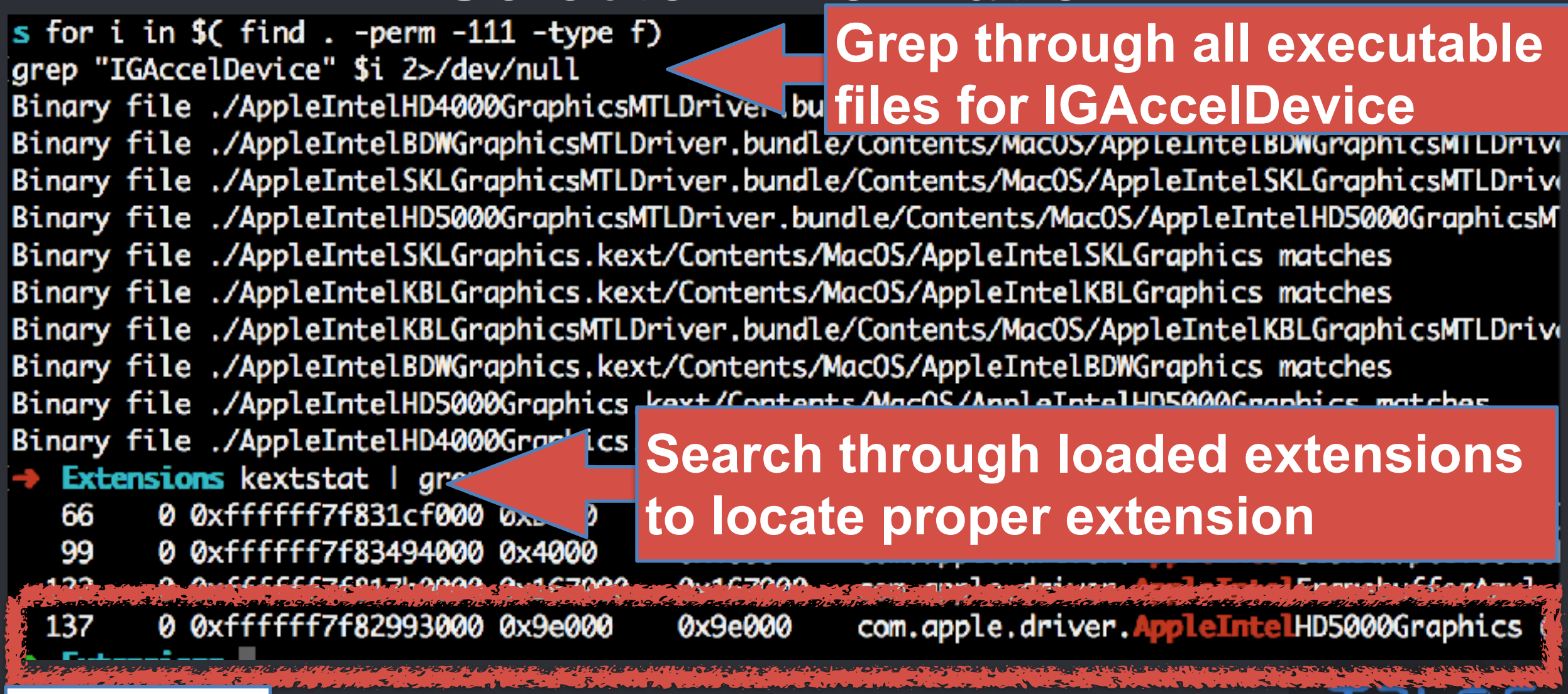


TALOS

Selector Information

- Resolved client to be used but still unsure of function being called
- Extensions located in **/System/Library/Extensions**
- Grep for classname: **IGAccelDevice**

Selector Information



AppleIntelHD5000

Talos

Yay IDA!

- Located proper binary
- IOKit UserClients dispatch functions via
  - **getTargetAndMethodForIndex**
  - **externalMethod**
- Selector value is passed in and proper function is dispatched

IOKit

```
147 IOExternalMethod* SimpleUserClientClassName::getTargetAndMethodForIndex(IOService** target, UInt32 index)
148 {
149     // Make sure that the index of the function we're calling actually exists in the function table.
150     if (index < (UInt32) kNumberOfMethods) {
151         if (sMethods[index].object == (IOService *) kMethodObjectThis) {
152             *target = this;
153         }
154         else {
155             *target = fProvider;
156         }
157         return (IOExternalMethod *) &sMethods[index];
158     }
159     return NULL;
```

MORE COMMON METHOD

Returns method pointer

TALOS

IOKit

```
228 IOReturn SimpleUserClientClassName::externalMethod(uint32_t selector, IOExternalMethodArguments* arguments,
229                                                    IOExternalMethodDispatch* dispatch, OSObject* target, void* reference)
230
231 {
232     if (selector < (uint32_t) kNumberOfMethods) {
233         dispatch = (IOExternalMethodDispatch *) &sMethods[selector];
234
235         if (!target) {
236             if (selector == kMyScalarIScalarOMethod) {
237                 target = fProvider;
238             }
239             else {
240                 target = this;
241             }
242         }
243     }
244     return super::externalMethod(selector, arguments, dispatch, target, reference);
245 }
```

Calls function in kernel directly

TALOS

IOKit RE

- Recap:
  - Binaries found in /System/Library/Extensions
  - Grep through locate correct binary
- Next Step:
  - Load into IDA and locate dispatch routine
  - Follow X-refs to locate the method table

```
const IOExternalMethodDispatch SimpleUserClientClassName::sMethods[kNumberOfMethods] = {
    {   // kMyUserClientOpen
        (IOExternalMethodAction) &SimpleUserClientClassName::sOpenUserClient,
        0,
        0,
        0,
        0
    },
    {   // kMyUserClientClose
        (IOExternalMethodAction) &SimpleUserClientClassName::sCloseUserClient,
        0,
        0,
        0,
        0
    },
    {   // kMyScalarIStructIMethod
        (IOExternalMethodAction) &SimpleUserClientClassName::sScalarIStructI,
        1,
        -1,
```

```
                                                    // No
                                                    // No
                                                    // No scalar output values.
                                                    // No struct output value.
```

```
                                                    // Method pointer.
                                                    // No scalar input values.
                                                    // No struct input value.
                                                    // No scalar output values.
                                                    // No struct output value.
```

```
                                                    // Method pointer.
                                                    // One scalar input value.
                                                    // The size of the input st
```

Method Table & Entry

Descriptor of expected arguments

TALOS

RE

```
signed __int64 __fastcall IGAccelDevice::getTargetAndMethodForIndex(
{
  signed __int64 result; // rax

  *a2 = a1;
  if ( selector <= 9 )
    JUMPOUT(__CS__, `vtable for'IOAccelDevice2[298]);
  if ( selec
    result =
  else
    result =                                    r - 10);
  return result;
}
```

Exercise for the viewer to locate functions for IOAccelDevice2 :)

Talos

- Following references it becomes easy to locate function definitions

```
db    0
dq offset IGAccelDevice::get_wa_table(_WA_TABLE *,_WA_TABLE *,ulon
db 8 dup(0), 3, 7 dup(0), 4 dup(0FFh), 4 dup(0), 4 dup(0FFh)
db 0Ch dup(0)
dq offset IGAccelDevice::get_hw_caps(_IntelHwCapsInfo *,_IntelHwCa
db 8 dup(0), 3, 7 dup(0), 4 dup(0FFh), 4 dup(0), 4 dup(0FFh)
db 0Ch dup(0)
dq offset IGAccelDevice::debug_control(IntelDeviceDebugControlInOu
db 8 dup(0), 3, 7 dup(0), 4 dup(0FFh), 4 dup(0), 4 dup(0FFh)
db 0Ch dup(0)
dq offset IGAccelDevice::telemetryMapStatsMem(IntelDeviceMapStatsM
db 8 dup(0), 3, 7 dup(0), 4 dup(0FFh), 4 dup(0), 4 dup(0FFh)
db 0Ch dup(0)
```

Values below the function determine arguments needed for each function call

```
db 8 dup(0), 3, 7 dup(0), 4 dup(0FFh), 4 dup(0), 4 dup(0FFh)
db 0Ch dup(0)
dq offset IGAccelDevice::telemetryMapOABufferMemory(IntelDeviceMap
db 8 dup(0), 3, 7 dup(0), 4 dup(0FFh), 4 dup(0), 4 dup(0FFh)
db 0Ch dup(0)
dq offset IGAccelDevice::logStatus(IntelDeviceLogStatus *,IntelDev
db 8 dup(0), 3, 7 dup(0), 4 dup(0FFh), 4 dup(0), 4 dup(0FFh)
db 0Ch dup(0)
dq offset IGAccelDevice::get_vram_size(ulong long *)
db 18h dup(0), 1, 0Fh dup(0)
dq offset IGAccelDevice::get_max_frequency(ulong long *)
db    0
```

TALOS

What do we have?

- Goal
  - Build an introspection tool to log kernel communications
- Known
  - Client communicating with
  - Selector of function to call
  - Functions for each selector

- Simple dictionary key value pairs in Javascript
- Methodology:
  - Grab client name (IGAccelDevice) from dictionary
  - Index in to correct selector

TALOS

```
var IGAccelDevice = ["IOAccelDevice2::get_config(IOAccelDeviceConfigData *)",
"IOAccelDevice2::get_name(char *)",
"IOAccelDevice2::get_event_machine(IOAccelDeviceEventMachineData *)",
"IOAccelDevice2::get_surface_info(uint,IOAccelDeviceSurfaceData *)",
"IOAccelDevice2::set_stereo(uint,uint)",
"IOAccelDevice2::get_next_g                                    eGlobalObjectIDData *)",
"IOAccelDevice2::get_current_trace_f                           TraceFilterData *)",
"IOAccelDevice2::get_device_info(IOA                           nData *)",
"IOAccelDevice2::get_next_gid_group(                           upData *)",
"IOAccelDevice2::set_api_property(IO                           rty *)",
"get_wa_table(_WA_TABLE *,_WA_TABLE *,ulong long,ulong long *                ",
"get_hw_caps(_IntelHwCapsInfo *,_IntelHwCapsInfo *,ulong long ulong",
"debug_control(IntelDeviceDebugControlInOutng,ulong long *)",
"telemetryMapStatsMem(IntelDeviceMapStatsMemInOutg,ulong lon                )",
"telemetryOperation(TelemetryOperation *,TelemetryOperati                   ",
"telemetryInitOABuffer(MDAPIInitOABufferOp *,MDAPIInitOABuff                ",
"telemetryReadOABuffer(MDAPIReadOABufferOpInng long *)",
"telemetryMapOABufferMemory(IntelDeviceMapStatsMemInOutg,ulo                ",
"logStatus(IntelDeviceLogStatus *,IntelDeviceLogStatus *,ulong",
"get_vram_size(ulong long *)",
"get_max_frequency(ulong long *)"]


clients["IGAccelDevice"] = IGAccelDevice
```

IOAccelDevice2 Methods from Ida

IGAccelDevice Methods from Ida


```
147] IntelAccelerator TYPE IGAccelDevice SELECTOR IOAccelDevice2::set_api_property
148] IntelAccelerator TYPE IGAccelDevice SELECTOR                    nt_machin

149] IntelAccelerator TYPE IGAccelDevice SELECTOR                    fig(IOAcc
150] IntelAccelerator TYPE IGAccelDevice SELECTOR IOAccelDevice2::get_device_info(
151] IntelAccelerator TYPE IGAccelSharedUserClient SELECTOR 9
152] IntelAccelerator TYPE IGAccelSharedUserClient SELECTOR 10
153] IntelAccelera                                  get_hw_caps(_IntelHwCapsInfo *,_

154] IntelAccelerator TYPE IGAccelDevice SELECTOR get_wa_table(_WA_TABLE *,_WA_TAB
155] IntelAccelerator TYPE IGAccelSharedUserClient SELECTOR 14
156] IntelAccelerator TYPE IGAccelDevice SELECTOR debug_control(IntelDeviceDebugCo
157] IntelAccelerator TYPE IGAccelDevice SELECTOR debug_control(IntelDeviceDebugCo
```

Client

Method

Talos

From here?

- Data manipulation
  - Modifying data as it travels through hooks
  - Reverse engineering specific areas to determine more intelligent fuzz
  - Auditing specific implementations using data flow as a guide

```
*** Panic Report ***
panic(cpu 2 caller 0xffffff800a4d7bc6): Kernel trap at 0xffffff7f8c977c55, type 14=page fault, regis
CR0: 0x0000000080010033, CR2: 0x0000023000030000, CR3: 0x0000000033279a06, CR4: 0x00000000001627e0
RAX: 0xffffff8043596030, RBX: 0x0000023017b37000, RCX: 0x0000023000030000, RDX: 0xffffff8030875f00
RSP: 0xffffff91fcea38d0, RBP: 0xffffff91fcea3950, RSI: 0x0000000000000000, RDI: 0x0000000000000000
R8:  0xffffff7f8b51c5fa, R9:  0x0000023000030002, R10: 0x0000000000001000, R11: 0xffffff802fa9b800
R12: 0xffffff8043657000, R13: 0xffffff81d98f7030, R14: 0xffffff91fcea390c, R15: 0xffffff8043657000
RFL: 0x0000000000210246, RIP: 0xffffff7f8c977c55, CS:  0x0000000000000008, SS:  0x0000000000000010
Fault CR2: 0x0000023000030000, Error code: 0x0000000000030002, Fault CPU: 0x2, PL: 0, VF: 0

Backtrace (CPU 2), Frame : Return Addre
0xffffff91fcea33a0 : 0xffffff800a37e1f6
0xffffff91fcea33f0 : 0xffffff800a4e6b74
0xffffff91fcea3430 : 0xffffff800a4d79d8
0xffffff91fcea34e0 : 0xffffff800a3221f2
0xffffff91fcea34c0 : 0xffffff800a37d8ea
0xffffff91fcea35f0 : 0xffffff800a37d6bc
0xffffff91fcea3650 : 0xffffff800a4d7bc6
0xffffff91fcea37c0 : 0xffffff800a3221f2
0xffffff91fcea37e0 : 0xffffff7f8c977c55
0xffffff91fcea3950 : 0xffffff7f8b602771
0xffffff91fcea3990 : 0xffffff7f8b602c03
0xffffff91fcea39c0 : 0xffffff7f8b60915e
0xffffff91fcea3a20 : 0xffffff7f8c97e095
0xffffff91fcea3b20 : 0xffffff800aa56168
0xffffff91fcea3b70 : 0xffffff800aa6ed67
0xffffff91fcea3cc0 : 0xffffff800a40c124
0xffffff91fcea3dc0 : 0xffffff800a383ca7
0xffffff91fcea3e10 : 0xffffff800a356cad
0xffffff91fcea3e60 : 0xffffff800a371a9b
0xffffff91fcea3ef0 : 0xffffff800a4bff8a
0xffffff91fcea3fa0 : 0xffffff800a3229f8
      Kernel Extensions in backtrace:
         com.apple.iokit.IOAcceleratorFamily2(378.18.1)[BAA0383C-9650-3934-B04A-69000F757A2C]@0xffff
            dependency: com.apple.driver.AppleMobileFileIntegrity(1.0.5)[54CD80E5-9FD7-30FC-09A0-E40
            dependency: com.apple.iokit.IOSurface(211.12)[E998B853-3174-3C25-B82B-C0D8BD9720E5]@0xff
            dependency: com.apple.iokit.IOPCIFamily(2.9)[1858E7DA-E767-3027-A3AA-637C80857219]@0xfff
            dependency: com.apple.iokit.IOGraphicsFamily(519.15)[D5F2A20D-CAB0-3382-91B9-E8755DFC34C
         com.apple.driver.AppleIntelHD5000Graphics(10.3.2)[18CF1D1B-BC94-3F6A-8711-8921CE16E38B]@0xf
            dependency: com.apple.iokit.IOSurface(211.12)[E998B853-3174-3C25-B82B-C0D8BD9720E5]@0xff
            dependency: com.apple.iokit.IOPCIFamily(2.9)[1858E7DA-E767-3027-A3AA-637C80857219]@0xfff
            dependency: com.apple.iokit.IOGraphicsFamily(519.15)[D5F2A20D-CAB0-3382-91B9-E8755DFC34C
            dependency: com.apple.iokit.IOAcceleratorFamily2(378.18.1)[BAA0383C-9650-3934-B04A-69000

BSD process name corresponding to current thread: VLC
Boot args: debug=0x146 kdp_match_name=en6 kext-dev-mode=1 pmuflags=1 -v keepsyms=1

Mac OS version:
17E199

Kernel version:
Darwin Kernel Version 17.5.0: Fri Apr 13 19:32:32 PDT 2018; root:xnu-4570.51.2~1/DEVELOPMENT_X86_64
Kernel UUID: 82028F9D-0818-353F-9453-77740F1F2D33
Kernel slide:
Kernel text base:
__HIB  text base:
System model name: MacBookPro11,4 (Mac-06F11FD93F0323C5)
```

Hiding stack trace

TALOS

Updates

twitter.com/1blankwall1

github.com/blankwall/

TALOS