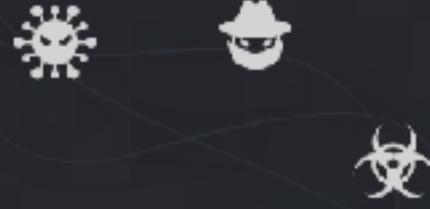


TALOS



PROTECTING YOUR NETWORK

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
- 

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



RANT

Multi

Vulns submitted in June!

Patched Mojave in October

We patches vulnerabilities silently and alerts them
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"))
```

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



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,\n      IOServiceMatching("AppleLMUController"));
5
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**
THE AUSTRALIAN NATIONAL UNIVERSITY

How do I access it?

- Lets build a
- Interposing confusing
- Learn the in



is inherently

```
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", \n  
          service, owningTask, type, connect, *connect, className);  
    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

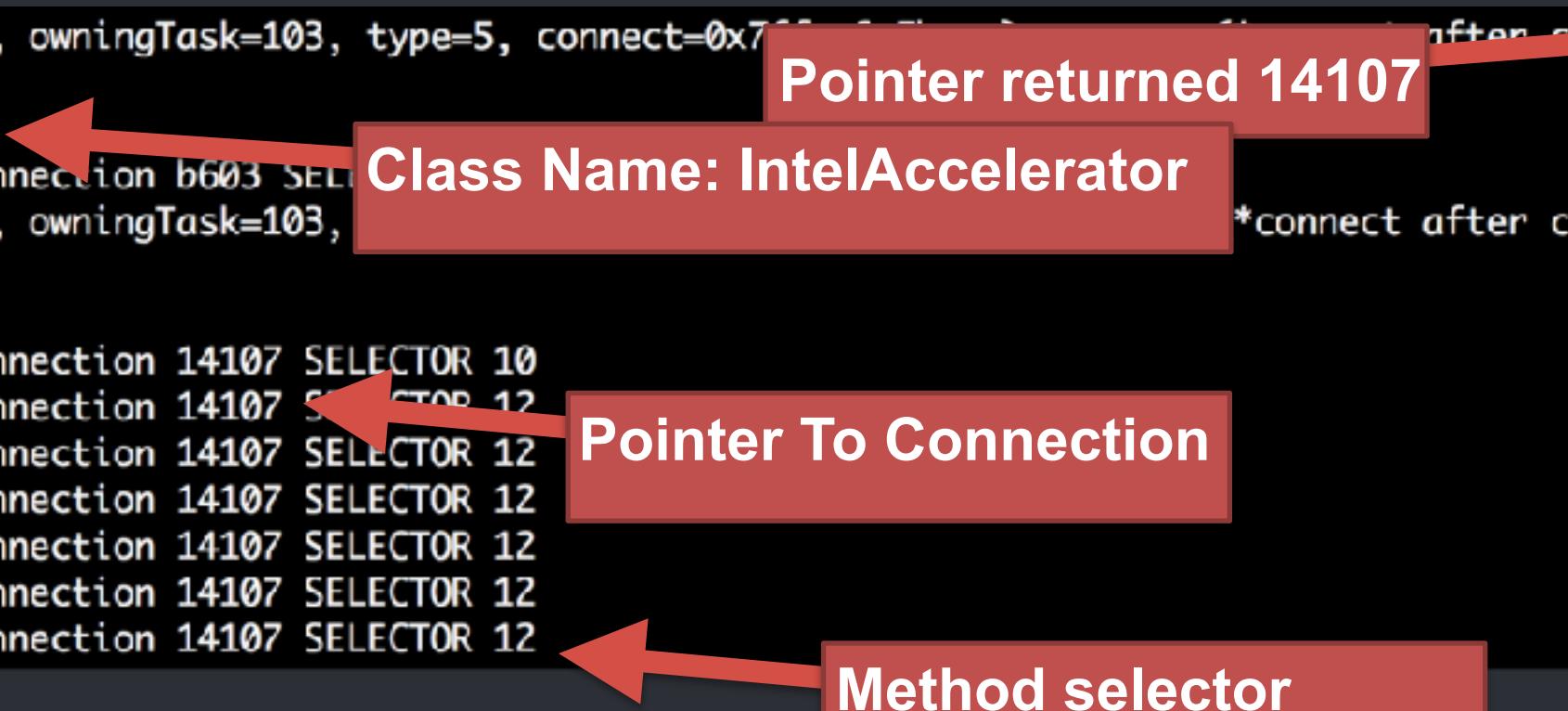
```
[>] IOServiceOpen(service=b703, owningTask=103, type=5, connect=0x7000000000000000) 14107  
)  
className IntelAccelerator  
[>] IOConnectCallMethod connection b603 SEL...  
IOServiceOpen(service=b703, owningTask=103, *connect after call = 1400b  
)  
className IntelAccelerator  
[>] IOConnectCallMethod connection 14107 SELECTOR 10  
[>] IOConnectCallMethod connection 14107 SEL...  
[>] IOConnectCallMethod connection 14107 SELECTOR 12  
[>] IOConnectCallMethod connection 14107 SELECTOR 12
```

Pointer returned 14107

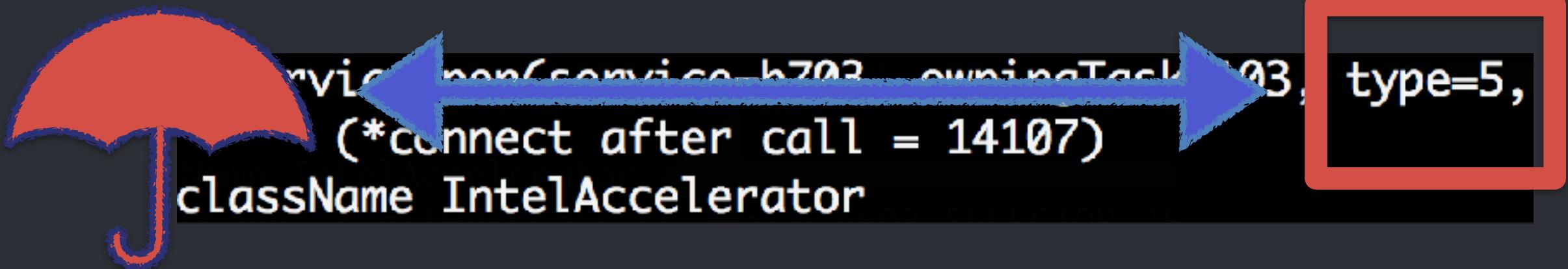
Class Name: IntelAccelerator

Pointer To Connection

Method selector



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=12, type=5,  
(*connect after call = 14107)  
className= IntelAccelerator
```

Name	Type	Spawn	Call ID
IntelAccelerator		0 (os/kern) successful	
IntelAccelerator		1 (os/kern) successful	
IntelAccelerator		2 (os/kern) successful	
IntelAccelerator		3 (iokit/common) unsupported function	
IntelAccelerator		4 (os/kern) successful	
IntelAccelerator		5 (os/kern) successful	51
IntelAccelerator		6 (os/kern) successful	
IntelAccelerator		7 (os/kern) successful	
IntelAccelerator		8 (os/kern) successful	
IntelAccelerator		9 (os/kern) successful	

Call ID 51 Flow:

```
IntelAccelerator → IOAccelSurface  
IntelAccelerator → IOAccelGLContext  
IntelAccelerator → IOAccel2DContext  
IntelAccelerator → IOAccelDisplayPipeUserClient  
IntelAccelerator → IOAccelDevice  
IntelAccelerator → IOAccelSharedUserClient  
IntelAccelerator → IOAccelMemoryInfoUserClient  
IntelAccelerator → IOAccelCLContext  
IntelAccelerator → IOAccel1GContext
```

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 -_-

```
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', [ptr]);
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(classname);
61             service_ids[handle] = [userclient, type];
62         }
63     }
64 });
65 }
```

Diagram illustrating the hooking process:

- Library and function to hook**: Points to the `IOServiceOpen` function in the `IOKit` library.
- Memory Storage**: Points to the `classname` variable where the class name of the IO object is stored.
- Retrieve ptr to library function**: Points to the `IOObjectGetClass` function being retrieved from the `IOKit` module.
- Call library func**: Points to the call to `IOObjectGetClassFunc` to retrieve the class name.
- Dictionary storage for later use**: Points to the `service_ids` dictionary where the retrieved class name and type are stored for later use.
- Retrieve previous**: Points to the `userclient` variable which is part of the previously stored entry in the `service_ids` dictionary.

A |

```
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", \n  
          service, owningTask, type, connect, *connect, className);  
    return ret;  
}
```

Open Connection

Extract Service Name

```
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)  
{
```

IOConnectCall*
multiple variants

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

TALOS

Library and function
to hook

Grab arguments

Query dictionary for
proper type

Output information
with correct client

```
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   }  
}, {  
  onLeave: function(args) {}  
})
```

Putting it Together

```
[IOConnectCallMethod # 15] IntelAccelerator TYPE IGAccelDevice SELECTOR 11
[IOConnectCallMethod # 16] IntelAccelerator TYPE IGAccelDevice SELECTOR 10
[IOConnectCallMethod # 17] IntelAccelerator TYPE IGAccelSharedUserClient SE
[IOConnectCallMethod # 18] IntelAccelerator TYPE IGAccelDevice SELECTOR 12
[IOConnectCallMethod # 19] IntelAccelerator TYPE IGAccelDevice SELECTOR 12
[IOConnectCallMethod # 20] IntelAccelerator TYPE IGAccelDevice SELECTOR 12
[IOConnectCallMethod # 21] IntelAccelerator TYPE IGAccelDevice SELECTOR 12
[IOConnectCallMethod # 22] IntelAccelerator TYPE IGAccelDevice SELECTOR 12
```

Correct types!

Selector?

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

```
s for i in $( find . -perm -111 -type f)  
grep "IGAccelDevice" $i 2>/dev/null
```

```
Binary file ./AppleIntelHD4000GraphicsMTLDriver.bundle
```

```
Binary file ./AppleIntelBDWGraphicsMTLDriver.bundle/Contents/MacOS/AppleIntelBDWGraphicsMTLDriver
```

```
Binary file ./AppleIntelSKLGraphicsMTLDriver.bundle/Contents/MacOS/AppleIntelSKLGraphicsMTLDriver
```

```
Binary file ./AppleIntelHD5000GraphicsMTLDriver.bundle/Contents/MacOS/AppleIntelHD5000GraphicsM
```

```
Binary file ./AppleIntelSKLGraphics.kext/Contents/MacOS/AppleIntelSKLGraphics matches
```

```
Binary file ./AppleIntelKBLGraphics.kext/Contents/MacOS/AppleIntelKBLGraphics matches
```

```
Binary file ./AppleIntelKBLGraphicsMTLDriver.bundle/Contents/MacOS/AppleIntelKBLGraphicsMTLDrive
```

```
Binary file ./AppleIntelBDWGraphics.kext/Contents/MacOS/AppleIntelBDWGraphics matches
```

```
Binary file ./AppleIntelHD5000Graphics.kext/Contents/MacOS/AppleIntelHD5000Graphics matches
```

```
Binary file ./AppleIntelHD4000Graphics.kext/Contents/MacOS/AppleIntelHD4000Graphics matches
```

```
→ Extensions kextstat | grep AppleIntelHD5000Graphics
```

```
66 0xfffffff7f831cf000 0x00000000
```

```
99 0xfffffff7f83494000 0x4000
```

```
122 0xfffffff7f831cf000 0x167000 com.apple.driver.AppleIntelEnergyEfficiency
```

```
137 0xfffffff7f82993000 0x9e000 0x9e000 com.apple.driver.AppleIntelHD5000Graphics
```

Grep through all executable files for IGAccelDevice

Search through loaded extensions to locate proper extension

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     if (selector < (uint32_t) kNumberOfMethods) {  
232         dispatch = (IOExternalMethodDispatch *) &sMethods[selector];  
233  
234         if (!target) {  
235             if (selector == kMyScalarIScalarOMethod) {  
236                 target = TProvider;  
237             }  
238             else {  
239                 target = this;  
240             }  
241         }  
242     }  
243     return super::externalMethod(selector, arguments, dispatch, target, reference);  
244 }  
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,  
        0,  
        0  
    }  
};
```

Method Table & Entry

Descriptor of expected arguments

// 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:

RE

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

    *a2 = a1;
    if ( selector <= 9 )
        JUMPOUT(__CS__, `vtable for' IOAccelDevice2[298]);
    if ( selected == 1 )
        result = IOAccelDevice2::getTargetAndMethodForIndex(.
    else
        result = IOAccelDevice2::getTargetAndMethodForIndex(.
    return result;
}
```

Exercise for the viewer to locate
functions for IOAccelDevice2 :)

- Following references it becomes easy to locate function definitions

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

Values below the function determine arguments needed for each function call

```
an 0
dq offset IGAccelDevice::telemetryMapOABufferMemory(IntelDeviceMap
db 8 dup(0), 3, 7 dup(0), 4 dup(OFFh), 4 dup(0), 4 dup(OFFh)
db 0Ch dup(0)
dq offset IGAccelDevice::logStatus(IntelDeviceLogStatus *, IntelDev
db 8 dup(0), 3, 7 dup(0), 4 dup(OFFh), 4 dup(0), 4 dup(OFFh)
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
```

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

```
94
95 var IGAcelDevice = ["IOAccelDevice2::get_config(IOAccelDeviceConfigData *)",
96 "IOAccelDevice2::get_name(char *)",
97 "IOAccelDevice2::get_event_machine(IOAccelDeviceEventMachineData *)",
98 "IOAccelDevice2::get_surface_info(uint,IOAccelDeviceSurfaceData *)",
99 "IOAccelDevice2::set_stereo(uint,uint)",
100 "IOAccelDevice2::get_next_gid_group",
101 "IOAccelDevice2::get_current_trace_file",
102 "IOAccelDevice2::get_device_info(IOAccelDeviceInfo *)",
103 "IOAccelDevice2::get_next_gid_group",
104 "IOAccelDevice2::set_api_property(IOAccelDeviceProperty *)",
105 "get_wa_table(_WA_TABLE *,_WA_TABLE *,ulong long,ulong long *)",
106 "get_hw_caps(_IntelHwCapsInfo *,_IntelHwCapsInfo *,ulong long,ulong",
107 "debug_control(IntelDeviceDebugControlInOutng,ulong long *)",
108 "telemetryMapStatsMem(IntelDeviceMapStatsMemInOutg,ulong long *)",
109 "telemetryOperation(TelemetryOperation *,TelemetryOperation *)",
110 "telemetryInitOABuffer(MDAPIInitOABufferOp *,MDAPIInitOABufferOp *",
111 "telemetryReadOABuffer(MDAPIReadOABufferOpInng long *)",
112 "telemetryMapOABufferMemory(IntelDeviceMapStatsMemInOutg,ulong long *)",
113 "logStatus(IntelDeviceLogStatus *,IntelDeviceLogStatus *,ulong long *)",
114 "get_vram_size(ulong long *)",
115 "get_max_frequency(ulong long *)"]
116
117
118 clients["IGAccelDevice"] = IGAcelDevice
119
```

IOAccelDevice2
Methods from
Ida

IGAccelDevice
Methods from
Ida

EQ AIDA

```
147] IntelAccelerator TYPE IGAccelDevice SELECTOR IOAccelDevice2::set_api_property  
148] IntelAccelerator TYPE IGAccelDevice SELECTOR IOAccelDevice2::get_hw_caps(->machin  
149] IntelAccelerator TYPE IGAccelDevice SELECTOR IOAccelDevice2::get_config(IOAcc  
150] IntelAccelerator TYPE IGAccelDevice SELECTOR IOAccelDevice2::get_device_info(  
151] IntelAccelerator TYPE IGAccelSharedUserClient SELECTOR 9  
152] IntelAccelerator TYPE IGAccelSharedUserClient SELECTOR 10  
153] IntelAccelerator TYPE IGAccelDevice SELECTOR 11 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  
158] IntelAccelerator TYPE IGAccelDevice SELECTOR debug_control(IntelDeviceDebugCo
```

Client

Method

Updates



twitter.com/1blankwall1

github.com/blankwall/