iOS内核漏洞挖掘

-

Fuzz & 代码审计

PANGU TEAM

xKungfoo 2015

议程

- iOS内核漏洞
- Fuzz系统
- 代码审计
- 漏洞分析

iOS安全体系

- 安全机制
  - Code Sign
  - Sandbox
  - Read-only Root FS
  - …
- 功能多在内核层实现
  - 需要patch内核来defeat

iOS内核漏洞

- 传统桌面系统内核漏洞
  - 能在低权限帐户触发
  - 修改进程标志位 -> 提权为System进程
- iOS内核漏洞
  - Sandbox内/外触发
  - Mobile/Root触发
  - Patch内核代码

iOS内核溢出保护

- 溢出保护
  - KASLR
  - Kernel pointer obfuscation
  - Stack / Heap cookie
  - Kernel space isolation
  - Kernel code page is R-X
  - Kernel heap is not executable
  - …

议程

- iOS内核漏洞
- Fuzz系统
- 代码审计
- 漏洞分析

Fuzz系统

- Why Fuzz
  - 容易实现
  - 覆盖面广
  - Apple的代码质量不高
  - 低投入高产出

Fuzz系统

- Why NOT Fuzz
  - 分析困难（无法调试）
  - Panic多 / Exploitable少
  - 欠缺精度

Fuzz系统

- Where to Fuzz
  - ioctl
  - sysctl
  - IOKit
  - File system
  - Network
  - …

Best Target

- IOKit
  - 扩展设备大多基于IOKit
  - 调用接口简单 - IOConnectCallMethod
  - IOKit Extensions大多闭源
  - 部分设备可在sandbox内打开

IOKit Fuzz

- 被动Fuzz
  - 更容易实现
  - 测试面狭窄
  - 需要人工干预
  - PoC精简相对麻烦
- 主动Fuzz
  - 需要不少准备工作
  - 测试覆盖面更广
  - 自动化
  - PoC往往更简洁

被动 IOKit Fuzz

- 核心思想 – hook
  - Cydia Substrate
    - 需要依赖Cydia Substrate环境
    - 可以一次注入所有进程
  - Dyld interpose feature
    - 实现简单
    - 需要自己实现注入 – DYLD_INSERT_LIBRARIES

被动 IOKit Fuzz

- Hook IOConnectCallMethod

```
kern_return_t
IOConnectCallMethod(
    mach_port_t      connection,          // In
    uint32_t selector,          // In
    const uint64_t   *input,             // In
    uint32_t inputCnt,          // In
    const void   *inputStruct,           // In
    size_t           inputStructCnt,     // In
    uint64_t*output,            // Out
    uint32_t*outputCnt,              // In/Out
    void      *outputStruct,        // Out
    size_t           *outputStructCntP)      // In/Out
```

被动 IOKit Fuzz

- input
  - 填充随机uint64_t
- inputStruct
  - 任意填充随机数据
  - 解析结构后替换某些数据
    - XML
    - …

被动 IOKit Fuzz

- 注入进程选择
  - Mobile Safari
  - …

主动 IOKit Fuzz

- 枚举所有可打开的设备
- 枚举设备对应的扩展方法
  - selector
  - inputCnt / inputStructCnt
  - outputCnt / outputStructCnt
- 主动测试所有的接口

主动 IOKit Fuzz

- IOConnectCallMethod -> IOUserClient::externalMethod
  - if dispatch != NULL
    - Size check
    - (*dispatch->function)(target, reference, args);
  - else method = getTargetAndMethodForIndex(&object, selector)
    - Size check
    - method->func

主动 IOKit Fuzz

- Find structure

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

struct IOExternalMethod {
    IOService *       object;
    IOMethod          func;
    IOOptionBits flags;
    IOByteCount       count0;
    IOByteCount       count1;
};
```

主动 IOKit Fuzz

- Overriding methods
  - externalMethod
  - getTargetAndMethodForIndex
  - getExternalMethodForIndex

主动 IOKit Fuzz

- 动态获取
  - 枚举所有设备
    - ioreg
  - 枚举扩展方法
    - IOServiceOpen打开设备
    - mach_port_kobject获取内核地址（需要获取vm permutation）
    - 读取对应的vtable数据
    - 定位重载的几个函数

主动 IOKit Fuzz

- 静态获取 – IDA脚本
  - 枚举所有设备
    - OSMetaClass (*UserClient)
  - 定位*UserClient的vtable
  - 定位重载的几个函数
  - 定位IOExternalMethodDispatch / IOExternalMethod结构

主动 IOKit Fuzz

- Fuzz IOConnectCallMethod
  - 实际调用io_connect_method
- Fuzz io_connect_method

```
kern_return_t io_connect_method
(
    mach_port_t connection,
    uint32_t selector,
    io_scalar_inband64_t input,
    mach_msg_type_number_t inputCnt,
    io_struct_inband_t inband_input,
    mach_msg_type_number_t inband_inputCnt,
    mach_vm_address_t ool_input,
    mach_vm_size_t  ool_input_size __unused,
    io_scalar_inband64_t output,
    mach_msg_type_number_t *outputCnt,
    io_struct_inband_t inband_output,
    mach_msg_type_number_t *inband_outputCnt,
    mach_vm_address_t ool_output,
    mach_vm_size_t * ool_output_size __unused
)
```

  - inputStructCnt/outputStructCnt > 4096
    - 使用 ool_input / ool_output

议程

- iOS内核漏洞
- Fuzz系统
- 代码审计
- 漏洞分析

代码审计

- Open Source
  - XNU
  - IOHIDFamily
  - …

代码审计

- Heap Overflow
- Integer Overflow
- Type Confusion
- Use after Free
- Logical Error
- Kernel Information Leak
- …

XNU

- iokit/Kernel
  - iokit基础库
- bsd/hfs
  - 文件系统
- bsd/dev
  - 系统设备
- bsd/netinet
  - IPv4接口
- bsd/netinet6
  - IPv6接口
- …

IOHIDFamily

- 搜索扩展方法接口
  - “IOExternalMethodDispatch “
  - “IOExternalMethod “

More

- Check at Apple open source
  - http://opensource.apple.com/release/os-x-1010/

议程

- iOS内核漏洞
- Fuzz系统
- 代码审计
- 漏洞分析

injectStringGated

- 典型的heap overflow
- Bug in IOHIDFamily
- Discovered by Luca Todesco (@qwertyoruiop)
- PoC - http://github.com/kpwn/vpwn

injectStringGated

- IOHIDSecurePromptClient selector =12
  - injectStringMethod -> injectStringGated
  - 输入是任意长度的struct

```
IOExternalMethod *
IOHIDSecurePromptClient::getTargetAndMethodForIndex(IOService **
    targetP,  UInt32 index)
{
        ...
        // 12: kIOHIDSecurePromptClient_injectString
        { NULL, (IOMethod)&IOHIDSecurePromptClient::injectStringMethod,
    kIOUCStructIStructO, kIOUCVariableStructureSize, 0 },
        ...
};
```

injectStringGated

```
IOReturn
IOHIDSecurePromptClient::injectStringGated(void * p1, void * p2, void * p3 __unused,void * p4 __unused)
{
    IOReturn result = kIOReturnBadArgument;
    IOHIDSecurePromptClient_RawKeystrokeData * dummyRawData = NULL;
    UTF32Char *string = (UTF32Char*)p1;
    intptr_t length = (intptr_t)p2 / sizeof(UTF32Char);
    vm_size_t dummyDataSize = length * sizeof(IOHIDSecurePromptClient_RawKeystrokeData);

    ...

    // _reserved->rawKeystrokes是在IOServiceOpen的时候分配的堆内存，在这里拷贝的时候没有检查输入的length
    __InsertBytes(_reserved->rawKeystrokes, _reserved->insertionPoint, _reserved->stringLength, string, length,
        sizeof(UTF32Char));
    __InsertBytes(_reserved->unicode, _reserved->insertionPoint, _reserved->stringLength, dummyRawData,
        length,  sizeof(UTF32Char));
    __EraseMemory(string, length * sizeof(UTF32Char));
    _reserved->insertionPoint += length;
    result = kIOReturnSuccess;

    ...
}
```

injectStringGated

- _reserved->rawKeystrokes 初始化

```
IOReturn
IOHIDSecurePromptClient::ensureBufferSize(UInt32 size)
{
    ...
    // newSize = size = 32
    result = kIOReturnNoMemory;
    require(newSize < 1024, finished);

    newBufferSize = newSize * (sizeof(UTF32Char) + sizeof(IOHIDSecurePromptClient_RawKeystrokeData));
    // newBufferSize = 32*12 = 384 分配的堆在kalloc.512的zone里
    newBuffer = (UInt8*)IOMalloc(newBufferSize);
    require(newBuffer, finished);
    newKeystrokeOffset = newBuffer + newSize * sizeof(UTF32Char);
    memcpy(newBuffer, _reserved->unicode, _reserved->stringLength * sizeof(UTF32Char));
    memcpy(newKeystrokeOffset, _reserved->rawKeystrokes, _reserved->stringLength *
      sizeof(IOHIDSecurePromptClient_RawKeystrokeData));
    oldBuffer = (UInt8*)_reserved->unicode;
    oldBufferSize = _reserved->bufferLength * (sizeof(UTF32Char) +
      sizeof(IOHIDSecurePromptClient_RawKeystrokeData));
    _reserved->unicode = (UTF32Char*)newBuffer;
    // _reserved->rawKeystrokes 被设置为 newBuffer+32*4 的位置
    _reserved->rawKeystrokes = (IOHIDSecurePromptClient_RawKeystrokeData*)newKeystrokeOffset;
    _reserved->bufferLength = newSize;
    newBuffer = NULL;
    result = kIOReturnSuccess;
    ...
}
```

CVE-2014-4487

- Heap overflow / Free into wrong zone
- Bug in IOHIDFamily
- Used in TaiG jailbreak

CVE-2014-4487

- IOHIDLibUserClient selector = 15
  - _getElements -> getElements
  - 输入1个scalar input
  - 输出任意长度的struct

```
{ //    kIOHIDLibUserClientGetElements
    (IOExternalMethodAction) &IOHIDLibUserClient::_getElements,
    1, 0,
    0, kIOUCVariableStructureSize
    },
```

CVE-2014-4487

- 出参必须是structureOutputDescriptor
  - 调用io_connect_method触发
    - ool_output

```
IOReturn IOHIDLibUserClient::_getElements(IOHIDLibUserClient * target, void * reference
    __unused, IOExternalMethodArguments * arguments)
{
   if ( arguments->structureOutputDescriptor )
      return target->getElements((uint32_t)arguments->scalarInput[0], arguments-
    >structureOutputDescriptor, &(arguments->structureOutputDescriptorSize));
   else
      return target->getElements((uint32_t)arguments->scalarInput[0], arguments-
    >structureOutput, &(arguments->structureOutputSize));
}
```

CVE-2014-4487

```
IOReturn IOHIDLibUserClient::getElements (uint32_t elementType, void *elementBuffer, uint32_t *elementBufferSize)
{
...
        elementLength = mem->getLength();
        if ( elementLength )
        {
            // 根据用户指定的大小来分配内存
            elementData = IOMalloc( elementLength );

            if ( elementData )
            {
                bzero(elementData, elementLength);

                // 向分配的堆中填充数据，elementLength更新为实际填充的数据长度
                ret = getElements(elementType, elementData, &elementLength);

                if ( elementBufferSize )
                    *elementBufferSize = elementLength;

                mem->writeBytes( 0, elementData, elementLength );

                // 释放刚分配的堆
                IOFree( elementData, elementLength );
...
}
```

CVE-2014-4487

- 填充elements时没长度检查 – Heap overflow
- 返回实际写入数据的长度 – Free into wrong zone

```
IOReturn IOHIDLibUserClient::getElements (uint32_t elementType, void *elementBuffer, uint32_t *elementBufferSize)
{
...
    if ( elementType == kHIDElementType )
        array = fNub->_reserved->hierarchElements;
    else
        array = fNub->_reserved->inputInterruptElementArray;
...
    count = array->getCount();
    bi = 0;
    for ( i=0; i<count; i++ )
    {
        element = OSDynamicCast(IOHIDElementPrivate, array->getObject(i));

        if (!element) continue;

        // 填充数据之前根本没有对size进行检查
        // Passing elementBuffer=0 means we are just attempting to get the count;
        elementStruct = elementBuffer ? &(((IOHIDElementStruct *)elementBuffer)[bi]) : 0;

        if ( element->fillElementStruct(elementStruct) )
            bi++;
    }

    // 输出长度修改为真实写入的数据长度
    if (elementBufferSize)
        *elementBufferSize = bi * sizeof(IOHIDElementStruct);
}
```

CVE-2014-4461

- 映射内核对象到用户态
- Bug in xnu/iokit/Kernel
- Used in Pangu Jailbreak

CVE-2014-4461

- IOSharedDataQueue
  - 用于管理队列数据
  - 数据内存区域能被映射到用户态
  - 可以设置port获取数据变动通知

CVE-2014-4461

- initWithCapacity函数将notifyMsg成员放在数据内存区域的尾部

```
Boolean IOSharedDataQueue::initWithCapacity(UInt32 size)
{
    ...

    allocSize = round_page(size + DATA_QUEUE_MEMORY_HEADER_SIZE +
      DATA_QUEUE_MEMORY_APPENDIX_SIZE);
    if (allocSize < size) {
        return false;
    }
    // 分配足够大小的内存给dataQueue
    dataQueue = (IODataQueueMemory *)IOMallocAligned(allocSize, PAGE_SIZE);
    if (dataQueue == 0) {
        return false;
    }
    ...
    // notifyMsg成员被放在dataQueue的尾部
    appendix          = (IODataQueueAppendix *)((UInt8 *)dataQueue + size +
      DATA_QUEUE_MEMORY_HEADER_SIZE);
    appendix->version   = 0;
    notifyMsg         = &(appendix->msgh);
    setNotificationPort(MACH_PORT_NULL);

    return true;
}
```

CVE-2014-4461

- 映射到用户态时包含了notifyMsg

```
IOMemoryDescriptor *IOSharedDataQueue::getMemoryDescriptor()
{
    IOMemoryDescriptor *descriptor = 0;

    if (dataQueue != 0) {
        descriptor = IOMemoryDescriptor::withAddress(dataQueue, getQueueSize() +
     DATA_QUEUE_MEMORY_HEADER_SIZE + DATA_QUEUE_MEMORY_APPENDIX_SIZE,
     kIODirectionOutIn);
    }

    return descriptor;
}
```

CVE-2014-4461

- notifyMsg – 包含一个内核对象
  - 可以在用户态下修改port指针地址

```
void IODataQueue::setNotificationPort(mach_port_t port)
{
    mach_msg_header_t * msgh = (mach_msg_header_t *) notifyMsg;

    if (msgh) {
        bzero(msgh, sizeof(mach_msg_header_t));
        msgh->msgh_bits = MACH_MSGH_BITS(MACH_MSG_TYPE_COPY_SEND, 0);
        msgh->msgh_size = sizeof(mach_msg_header_t);
        msgh->msgh_remote_port = port;
    }
}
```

CVE-2014-4461

- 当队列数据变化时会向port发送一个msg
  - 转换成 write-what-where

```
void IODataQueue::sendDataAvailableNotification()
{
    kern_return_t       kr;
    mach_msg_header_t * msgh;

    msgh = (mach_msg_header_t *) notifyMsg;
    if (msgh && msgh->msgh_remote_port) {
        kr = mach_msg_send_from_kernel_with_options(msgh, msgh->msgh_size, MACH_SEND_TIMEOUT,
      MACH_MSG_TIMEOUT_NONE);
        switch(kr) {
            case MACH_SEND_TIMED_OUT:    // Notification already sent
            case MACH_MSG_SUCCESS:
            case MACH_SEND_NO_BUFFER:
                break;
            default:
                IOLog("%s: dataAvailableNotification failed - msg_send returned: %dn",
      /*getName()*/"IODataQueue", kr);
                break;
        }
    }
}
```

Q & A

PANGU TEAM