The Python Bites your Apple

Fuzzing and exploiting OSX Kernel bugs

Flanker

KeenLab Tencent

XKungfoo Shanghai, April 2016

科恩实验室
KEEN
security lab

Table of Contents

科恩实验室 KEEN security lab

About me

Senior Security Researcher at KeenLab, Tencent

- Pwn2Own 2016 OSX Category Winner
- BlackHat, CanSecWest, HITCON, QCon Speaker
- *nix platform sandbox bypass and kernel exploitation
- Google Android Security Top Researchers Hall of Fame

科恩实验室 KEEN security lab

About KeenLab

- Former KeenTeam with all researchers move to Tencent and form KeenLab
- 8 Pwn2Own Champions
- Universal Rooting
- We're hiring!

科恩实验室
KEEN
security lab

Objective of this talk

- Basic description of IOKit
- Kernel Zone Allocator and Fengshui Technique
- Introducing KitLib and distributed Fuzzer
- Introducing Kexthelper, a IDA plugin for OSX KEXT
- Case Studies

IOKit Security Background

What's IOKit

I/O Kit is a collection of system frameworks, libraries, tools, and other resources for creating device drivers in OS X

- Security researchers tend to refer it as Kernel drivers and frameworks written with IOKit and accessible via IOKit method calls

Why attacking IOKit

- IOKit drivers runs in Kernel space, some of them even reachable from browser sandbox for efficiency (Graphics).
- Huge number of drivers implemented
- Few access restrictions (compared to Android)

IOKit Security Background

IOService

Different services are exposed via IOKit. We can consult most of them in Hardware IO tools and ioreg.

- IOAccelerator
- IOHIDDevice
- IOPMrootDomain
- ...

IOKit Security Background

IOUserClient

External method calls are first routed via IOUserClient, triggered by mach_msg IPC

```
kern_return_t IOServiceOpen( io_service_t service, task_port_t owningTask, uint32_t type,
      io_connect_t *connect );
```

- When userspace openService is called, corresponding newUserClient at Kernel space is invoked
- Different types may map to different userClient
  - IOAccelerator
- May check caller's identity: is root? (OSX and iOS differ)
- IOServiceClose maps to clientClose in is_io_service_close (race condition here?)

IOKit Security Background

externalMethod

Method calls are dispatched through getTargetMethodForIndex and/or externalMethod

```
virtual IOExternalMethod * getTargetAndMethodForIndex(IOService ** targetP, UInt32 index );
virtual IOReturn     externalMethod(uint32_t selector, IOExternalMethodArguments *
      args,IOExternalMethodDispatch * dispatch, OSObject * target, void * reference)
```

- IOExternalMethodArguments is constructed in is_io_connect_method containing incoming parameter
- IOExternalMethod/IOExternalMethodDispatch specifies parameters constraint

IOKit Security Background

Calling conventions

- structureInput will be converted to descriptor if size >0x4000 in IOConnectCallMethod
- if size <0x4000 passes through inband buffer
- io_connect_method generate and send mach_msg
- of course we can directly call io_connect_method, bypass this constraint

libkern C++ Runtime

- Reduced set of C++
  - No Exception
  - No Multiple Inherience
  - No template
  - Custom implementation of RTTI
- OSMetaClass
  - OSMetaClass is defined by macros
  - Contains Name, size and father class for each corresponding class

IOKit Security Background

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
```

IOKit historical vulnerabilities

- Race condition (hottest) (CVE-2015-7084)
- Heap overflow
- UAF
- TOCTOU
- OOB write (Our P2O bug!)
- NULL dereference (not exploitable with SMAP and in sandbox)

Cons of traditional fuzzer

- written in C/C++, more time consuming
- error-prone, easy to make mistakes
- less supporting library
  - socket
  - logging
- not dynamically expandable due to language nature

Introducing Kitlib

```
io_connect_t t;
io_service_t svc =
      IOServiceGetMatchingService(kIOMasterPortDefault,
      IOServiceMatching("fuckliangchen"));
IOServiceOpen(svc, mach_task_self(), 0, &t);
uint32_t outputCnt = 0x100;
size_t outputStructCnt = 0x2000;
uint64_t* output = new uint64_t[outputCnt];
char* outputStruct = new
      char[outputStructCnt];

const uint32_t inputCnt = 0x100;
uint64_t input[inputCnt];
const size_t inputStructCnt = 0x2000;
char inputStruct[inputStructCnt];
IOConnectCallMethod(t, 0, input, inputCnt,
      inputStruct, inputStructCnt, output,
      &outputCnt, outputStruct,
      &outputStructCnt);
```

```
import kitlib
h = kitlib.openSvc('fuckliangchen', 0)
kitlib.callConnectMethod(h, [0L]*0x100,
      'a'*0x2000, 0x100, 0x2000)
```

What's Kitlib

- Kitlib is a Python wrapper for IOKit calls
- Internally written in C++ and Python, provides convenient functions for writing fuzzers, using SWIG and ctypes
- Performance cost is low

SWIG usage

- Define interfaces in header file (kitlib.h)
- C++ wrapper layer in cpp file (if needed) (kitlib.cpp)
- Writing wrapping code for argument convention from Python to C++ in glue file (kitlib.i)

Questions

- Memory management?
- Type conventions?

Header file

Defining basic types

- mach_port_t
- mach_vm_address_t
- io_object_t
- io_service_t
- io_iterator_t

Define interfaces

```
mach_port_t openSvc(const char* svc_name, uint32_t type);
mach_port_t* openMultiSvc(const char* svc_name, uint32_t* typearr);
size_t getSvcCntForName(const char* svc_name);
bool svcAva(const char* svc_name,uint32_t type);
```

Glue file

Wraps functions

```
size_t getSvcCntForName(const char* svc_name)
{
    io_iterator_t iter;
    kern_return_t kr;
    io_service_t device;
    size_t ret = 0;

    kr = IOServiceGetMatchingServices(kIOMasterPortDefault, IOServiceMatching(svc_name), &iter);
    while ((device = IOIteratorNext(iter)))
    {
        ++ret;
        IOObjectRelease(device);
    }
    IOObjectRelease(iter);
    return ret;
}
```

Glue file

Wraps functions

```
unsigned int callConnectMethod(
                    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    *outputStructCnt)
{    kern_return_t kt = IOConnectCallMethod((mach_port_t) connection, /* Connection  */
                                        selector,          /* Selector      */
                                        input, inputCnt,             /* input, inputCnt */
                                        inputStruct,       /* inputStruct   */
                                        inputStructCnt,            /* inputStructCnt */
                                        output, outputCnt, outputStruct, outputStructCnt); /*
                                              Output stuff */

    return kt;
}
```

Argument Wrapping

```
/*
 This function handles scalar input, translate the incoming python value, which is list,
 to native representation. Memory cleanup is needed afterwards
 */
%typemap(in) (const uint64_t *input, uint32_t inputCnt) {
    /* Check if is a list */
    if (PyList_Check($input)) {
        uint32_t size = PyList_Size($input);
        uint32_t i = 0;
        $2 = size;
        $1 = (uint64_t*) malloc(size * sizeof(uint64_t));
        for (i = 0; i < size; i++) {
            PyObject *o = PyList_GetItem($input,i);
            if (PyLong_Check(o)) {
                $1[i] = PyLong_AsUnsignedLongLong(o);
            }
            else {
                PyErr_SetString(PyExc_TypeError,"list must contain L numbers");
                free($1);
                return NULL;
            }
        }
    } else if ($input == Py_None) {
        $1 = NULL;
        $2 = 0;
    } else {
        PyErr_SetString(PyExc_TypeError,"not a list");
```

Argument wrapping

freeing memory

```
%typemap(freearg) (const uint64_t *input, uint32_t inputCnt) {
    free($1);
}
```

Call flow

- user Python code calls in SWIG auto-generated function
- SWIG auto-generated function calls user convention typemap code, mapping python types to C++ arguments $1 $2, etc
- SWIG auto-generated function passes $1 $2, etc into C++ glue code
- SWIG auto-generated function calls freearg code to free memory

Kitlib Implementation

Example source code

```
import kitlib
h = kitlib.openSvc('fuckliangchen', 0)
kitlib.callConnectMethod(h, [0L]*0x100, 'a'*0x2000, 0x100, 0x2000)
```

- inputScalar to list
- inputStruct to string/bytearray
- outputScalar/outputStruct maps to len
- output maps to tuple (retcode, outscalar, outstruct)

Problems for Kitlib1

- I'm lazy and I don't want to write wrapper for each interface
- Argument passing is immutable, cannot do TOCTOU fuzzing

Calling directly into library functions

ctypes

- For functions without need for wrapping we directly call ctypes
- Build-in mutable support (TOCTOU and race condition fuzzing)

Basic ctypes primitives

ctypes

- c_int, c_ulonglong, c_char_p
- create_string_buffer

Parsing KEXT for arguments

- static const IOExternalMethodDispath/IOExternalMethod array (parseable)
- dynamic routed via code (oops)

- for former we can automatically retrieve arguments via pattern matching
- Scan const-data section for matching
- Map class to arguments array

Example IOExternalMethodDispatch

```
__const:0000000000064BC0 ; IGAccelCLContext::attach(IOService *)::methodDescs
__const:0000000000064BC0 __ZZN16IGAccelCLContext6attachEP9IOServiceE11methodDescs db 0
__const:0000000000064BC0                                         ; DATA XREF:
      IGAccelCLContext::attach(IOService *)+16
__const:0000000000064BC1                 db    0
__const:0000000000064BC2                 db    0
__const:0000000000064BC3                 db    0
__const:0000000000064BC4                 db    0
__const:0000000000064BC5                 db    0
__const:0000000000064BC6                 db    0
__const:0000000000064BC7                 db    0
__const:0000000000064BC8                 dq offset
      __ZN16IGAccelCLContext15map_user_memoryEP22IntelCLMapUserMemoryInP23IntelCLMapUserMemoryOutyPy
      ; IGAccelCLContext::map_user_memory(IntelCLMapUserMemoryIn *,IntelCLMapUserMemoryOut
      *,ulong long,ulong long *)
__const:0000000000064BD0                 db    0
__const:0000000000064BD1                 db    0
__const:0000000000064BD2                 db    0
__const:0000000000064BD3                 db    0
__const:0000000000064BD4                 db    0
__const:0000000000064BD5                 db    0
__const:0000000000064BD6                 db    0
__const:0000000000064BD7                 db    0
__const:0000000000064BD8                 db    3
__const:0000000000064BD9                 db    0
__const:0000000000064BDA                 db    0
```

After parsing

```
_const:0000000000064BC0 ; IGAccelCLContext::attach(IOService *)::methodDescs
__const:0000000000064BC0 __ZZN16IGAccelCLContext6attachEP9IOServiceE11methodDescs
      IOExternalMethod <0, \
__const:0000000000064BC0                                        ; DATA XREF:
      IGAccelCLContext::attach(IOService *)+16
__const:0000000000064BC0                                  offset
      __ZN16IGAccelCLContext15map_user_memoryEP22IntelCLMapUserMemoryInP23IntelCLMapUserMemoryOutyPy,\
      ; IGAccelCLContext::map_user_memory(IntelCLMapUserMemoryIn *,IntelCLMapUserMemoryOut
      *,ulong long,ulong long *)
__const:0000000000064BC0                                  0, 3, 0FFFFFFFFh, 0FFFFFFFFh>
__const:0000000000064BF0                       IOExternalMethod <0, \ ;
      IGAccelCLContext::unmap_user_memory(IntelCLUnmapUserMemoryIn *,ulong long)
__const:0000000000064BF0                                  offset
      __ZN16IGAccelCLContext17unmap_user_memoryEP24IntelCLUnmapUserMemoryIny,\
__const:0000000000064BF0                                  0, 4, 0, 0FFFFFFFFh>
```

Example: IOExternalMethodDispatch matching

- +0 points to TEXT or zero
- +8, +16, +24 reasonable integers (NO TEXT pointing)

Example: Vtable matching

- Reference from constructor
- +8, +16, ... all points to TEXT section or EXTERN(UNDEF) section

Pseudo Algo

```
matchers = [IOExternalMethodDispatchMatcher, VtableMatcher, IOExternalMethodMatcher]
for matcher in matchers:
    if matcher.isSectionBegin(const_data_section):
        matcher.deflateToList(section, offset)
        break

map userclient with arguments
scan newUserClient for service-userclient mappings
add father's userclient to children service
scan constructor for service size
scan for offset access to determine field offset

Construct vtable as a structure S, set vt(offset 0, size 8) type to S*
```

Restoring object structure

- Scanning MetaClass function for object's size
- Create the object's whole vtable area as a struct
- Setting +0(8) field as vt and type to previous struct pointer

Restoring object structure(cont.)

```
//...
__text:0000000000048079          mov     r14, rdi
__text:000000000004807C          movzx   eax, word ptr [rbx+22h]
__text:0000000000048080          cmp     eax, 3
__text:0000000000048083          jnz     loc_48122
__text:0000000000048089          cmp     qword ptr [rbx], 0
__text:000000000004808D          jz      loc_48129
//...
__text:00000000000480A6          cmp     rax, rcx
__text:00000000000480A9          jnb     short loc_48129
__text:00000000000480AB          mov     rax, [r14]
__text:00000000000480AE          mov     rdi, r14
__text:00000000000480B1          mov     rsi, rbx
__text:00000000000480B4          call    qword ptr [rax+9E8h]
__text:00000000000480BA          mov     rdi, [r14+1030h]
__text:00000000000480C1          mov     rax, [rdi]
__text:00000000000480C4          mov     esi, [r15+8]
__text:00000000000480C8          shl     rsi, 6
__text:00000000000480CC          add     rsi, [r14+610h]
__text:00000000000480D3          call    qword ptr [rax+140h]
__text:00000000000480D9          inc     dword ptr [rbx+2Ch]
__text:00000000000480DC          cmp     byte ptr [rbx+30h], 0
__text:00000000000480E0          jz      short loc_480F8
__text:00000000000480E2          mov     eax, [r15+8]
__text:00000000000480E6          mov     rcx, [r14+608h]
```

科恩实验室 KEEN security lab

Restoring object structure(cont.)

- Scanning functions and perform forward-flow analysis on registers starting RDI
- Retriving MOV/LEA offset and do addStructMember

```
__int64 __fastcall IGAccelCLContext::process_token_SetFence(__int64 this, __int64 a2)
{
  __int64 v2; // r15@3
  __int64 result; // rax@6
  unsigned int v4; // esi@7

  if ( *(_WORD *)(a2 + 34) != 3 )
  {
    v4 = -5;
    return IOAccelContext2::setContextError((IOAccelContext2 *)this, v4);
  }
  if ( !*(_QWORD *)a2
    || (v2 = *(_QWORD *)(a2 + 24), (unsigned __int64)*(unsigned int *)(v2 + 8) << 6 >=
          *(unsigned int *)(this + 1600)) )
  {
    v4 = -2;
    return IOAccelContext2::setContextError((IOAccelContext2 *)this, v4);
  }
  (*(void (__cdecl **)(__int64))(*(_QWORD *)this + 2536LL))(this);
  (*(void (__fastcall **)(_QWORD, unsigned __int64))(**(_QWORD **)(this + 4144) + 320LL))(
    *(_QWORD *)(this + 4144),
    *(_QWORD *)(this + 1552) + ((unsigned __int64)*(unsigned int *)(v2 + 8) << 6));
  ++*(_DWORD *)(a2 + 44);
  if ( *(_BYTE *)(a2 + 48) )
    *(_DWORD *)(*(_QWORD *)(this + 1544) + 16LL * *(unsigned int *)(v2 + 8)) = 0;
  *(_BYTE *)(this + 4154) = 1;
  result = (*(__int64 (__fastcall **)(__int64, __int64))(*(_QWORD *)this + 2528LL))(this, a2);
  *(_BYTE *)(this + 4154) = 0;
  return result;
}
```

```
__int64 __fastcall IGAccelCLContext::process_token_SetFence(IGAccelCLContext *this, __int64 a2)
{
  __int64 v2; // r15@3
  __int64 result; // rax@6
  unsigned int v4; // esi@7

  if ( *(_WORD *)(a2 + 34) != 3 )
  {
    v4 = -5;
    return IOAccelContext2::setContextError((IOAccelContext2 *)this, v4);
  }
  if ( !*(_QWORD *)a2
    || (v2 = *(_QWORD *)(a2 + 24), (unsigned __int64)*(unsigned int *)(v2 + 8) << 6 >=
          LODWORD(this->field_640)) )
  {
    v4 = -2;
    return IOAccelContext2::setContextError((IOAccelContext2 *)this, v4);
  }
  ((void (__cdecl *)(IGAccelCLContext
        *))this->vt->__ZN16IGAccelCLContext16endCommandStreamER24IOAccelCommandStreamInfo)(this);
  (*(void (__fastcall **)(__int64, unsigned __int64))(*(_QWORD *)this->field_1030 + 320LL))(
    this->field_1030,
    *(_QWORD *)&this->gap610[0] + ((unsigned __int64)*(unsigned int *)(v2 + 8) << 6));
  ++*(_DWORD *)(a2 + 44);
  if ( *(_BYTE *)(a2 + 48) )
    *(_DWORD *)(this->field_608 + 16LL * *(unsigned int *)(v2 + 8)) = 0;
  this->field_103a = 1;
  result = ((__int64 (__fastcall *)(IGAccelCLContext *,
        __int64))this->vt->__ZN16IGAccelCLContext18beginCommandStreamER24IOAccelCommandStreamInfo)(
             this,
             a2);
  this->field_103a = 0;
```

Running fuzzer

- retriving metadata for all kexts and store using pickle
  - idc.Batch
- Set up multiple VM on fuzzing server
- add fuzzer as start-up item, load pickle and record progress

Fuzzing outcome

- Heap overflow in AppleXXX (CVE-2016-?)
- Race condition in IOXXX (CVE-2016-?)
- Double free in AppleXXX (CVE-2016-?)
- Integer overflow in IOXXX (CVE-2016-?)
- NULL pointer dereferences in IOXXX (CVE-2016-?)
- .....(more waiting disclosure)

科恩实验室 KEEN security lab

A hidden ignored attack surface in IOKit

oops, Apple hasn't fixed it yet, will disclose later.

Infoleak in AppleBDWGraphics/IntelHD5000 via racecondition

- IGAccelCLContext/IGAccelGLContext provides interface via externalMethod for mapping/unmapping user memory, passed in mach_vm_address_t
- Ian Beer and us both discovered a race condition in unmapping user memory, which lead to code execution
- Apple fixes this issue by adding a lock in un_map_usermemory (the delete operation), but its incomplete.

Operation Procedure

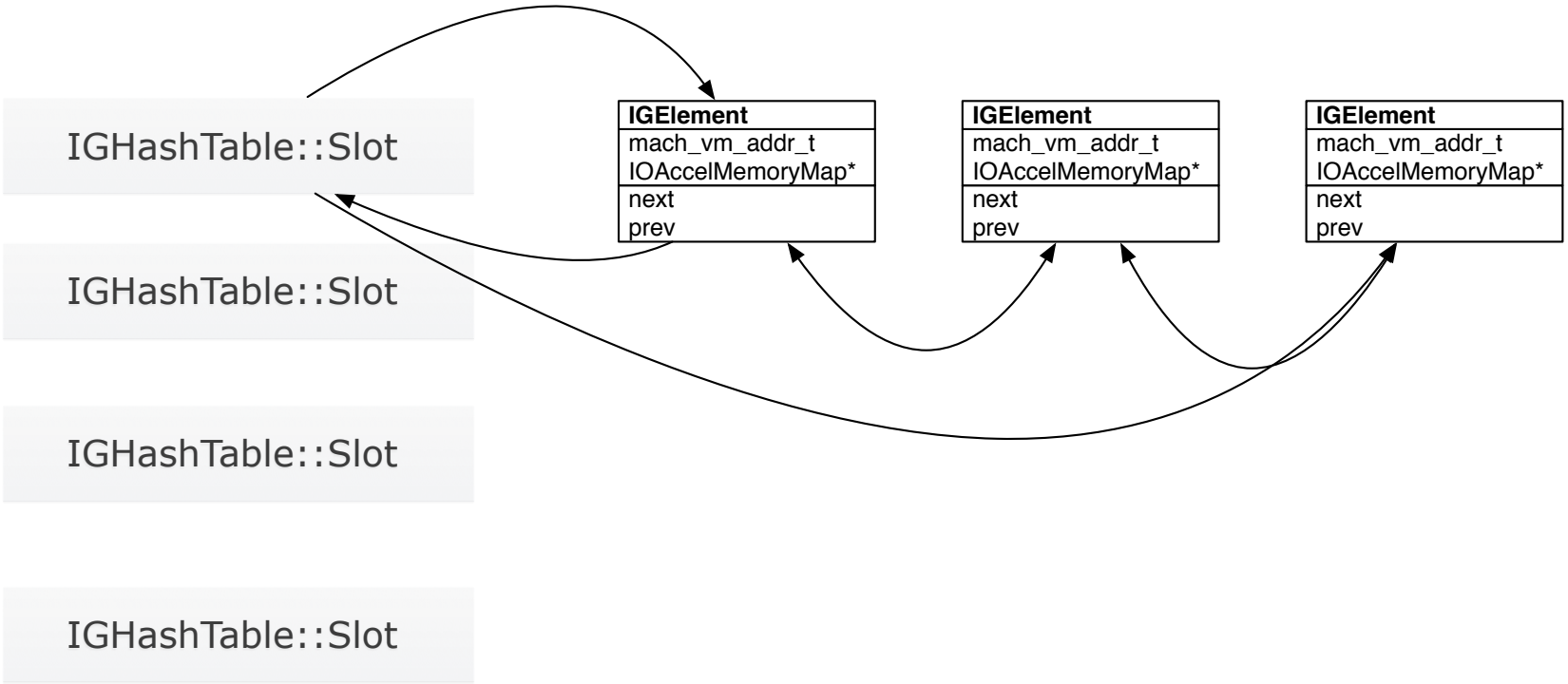
MAP_USER_MEMORY

- contains
  - index slot using hash function
  - iterate the list for matching
- add
  - Append item to corresponding slot list

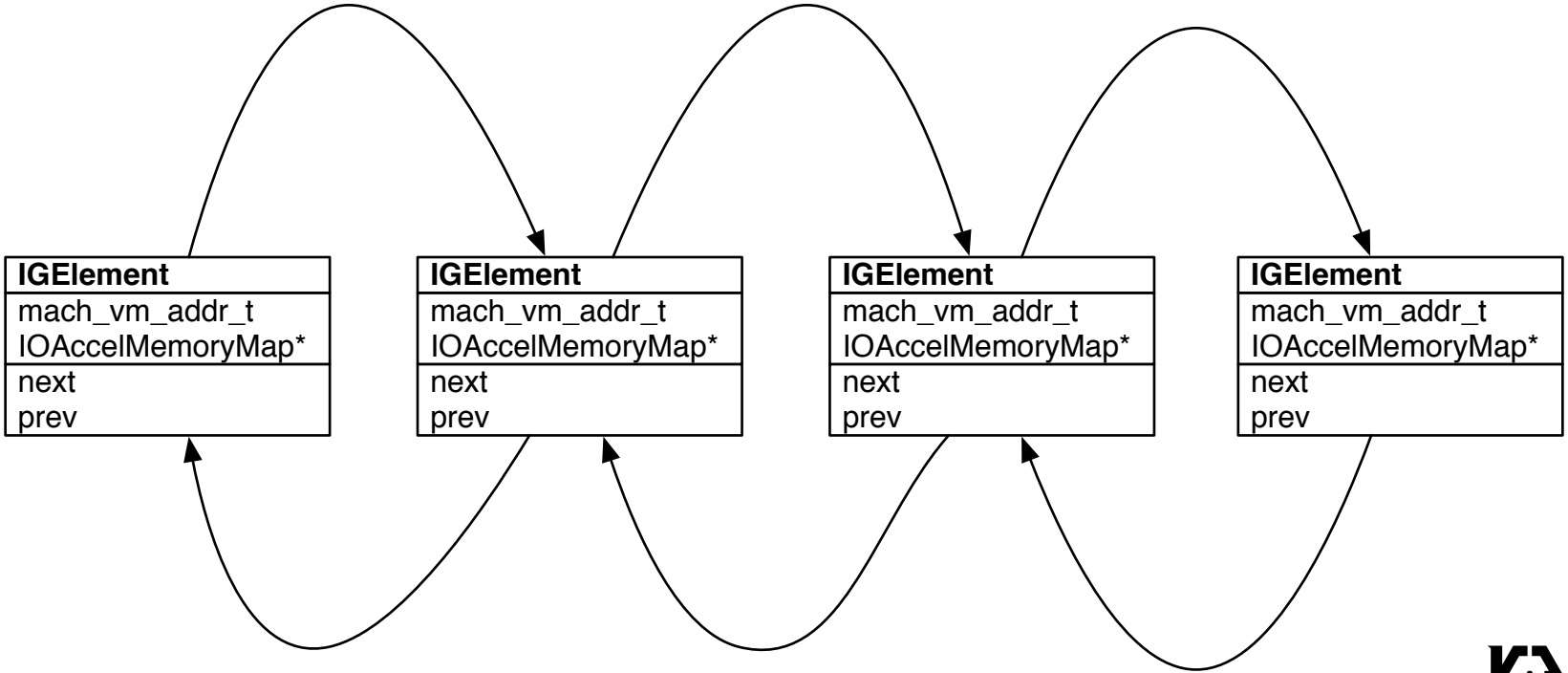
UNMAP_USER_MEMORY

- contains
- get
- remove (get a object ptr and call virtual function)
  - Update head and tail when appropriate
  - Update prev-¿next and next-¿prev

The IGHashTable structure



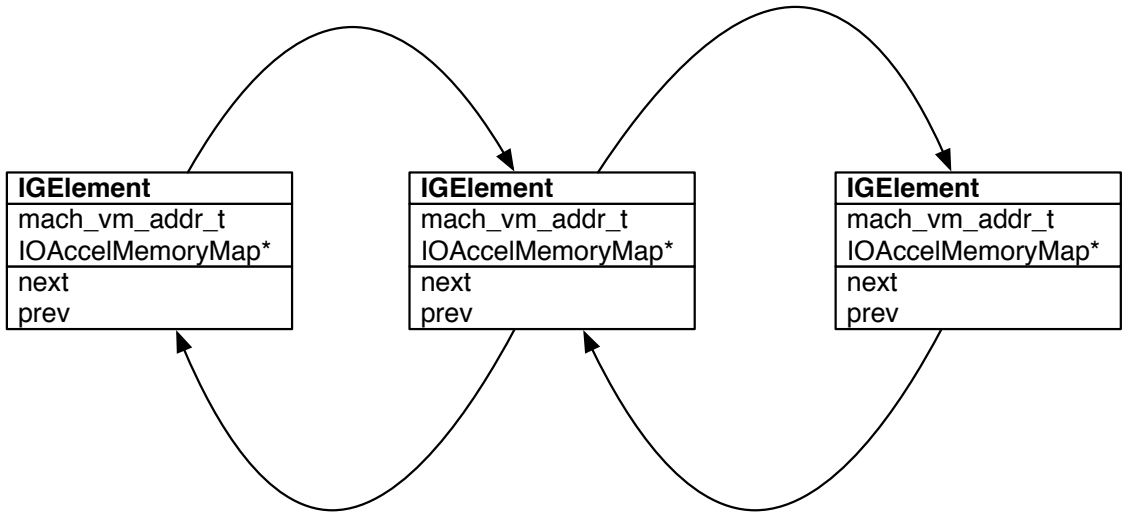
The LinkedList connecting elements with same hash values



科恩实验室
KEEN
security lab

The normal idea that failes (Ian Beer one)

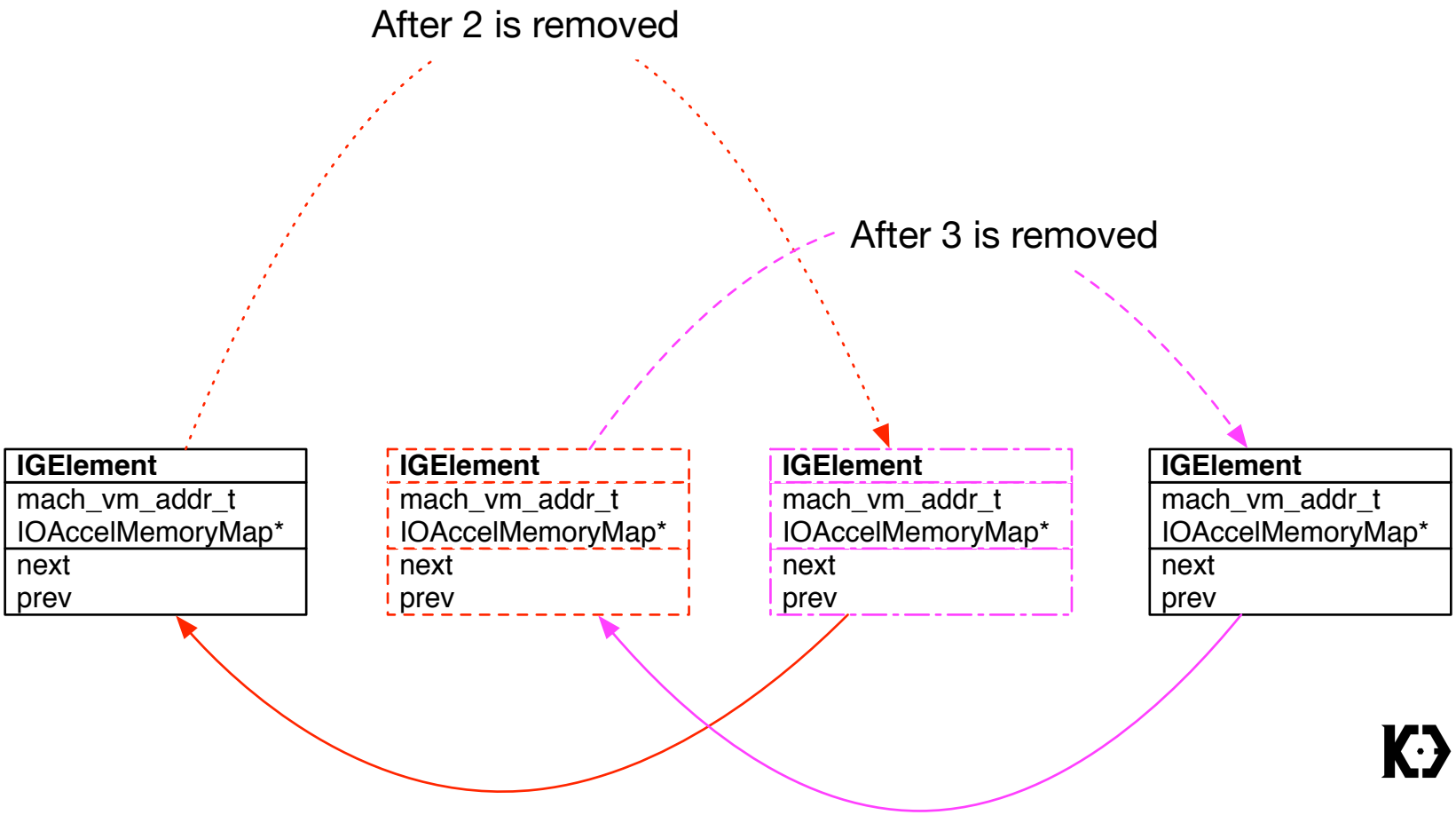
The ideal situation is both threads passes hash table::contains, and when one is retrieving IOAccelMemoryMap* after get returns valid pointer, the other frees it and we control the pointer



However in reality more frequently they do passes contains but thread 1 will remove it before thread 2 do get and thread 2 hit a null pointer dereference

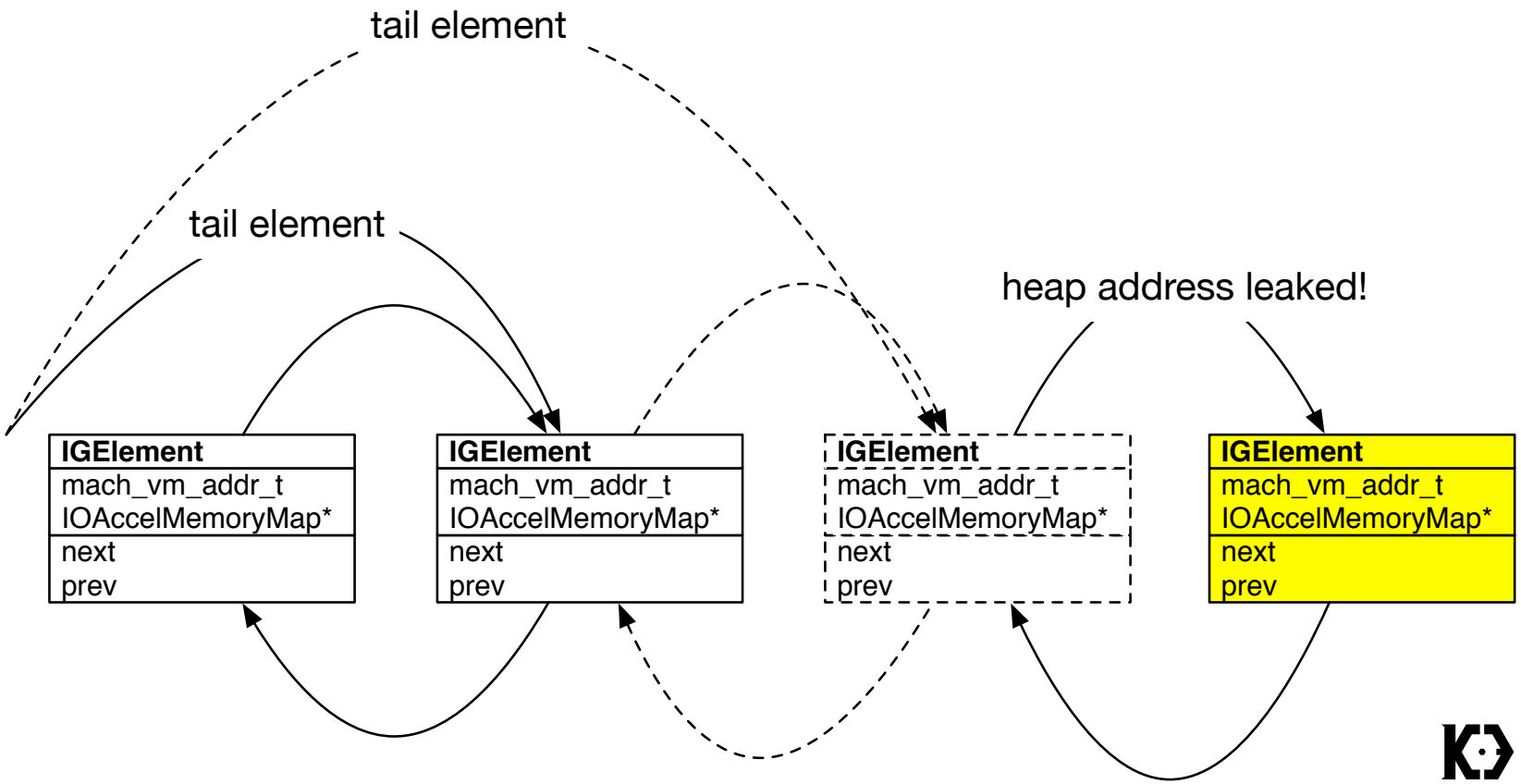
科恩实验室 KEEN security lab

The advanced racing by us



科恩实验室
KEEN
security lab

The new vulnerability



科恩实验室 KEEN security lab

Unmap frees the element while map is still traversing

```
*** Panic Report ***
panic(cpu 0 caller 0xffffff80165ce40a): Kernel trap at 0xffffff80165c4e90, type 13=general protection, registers:
CR0: 0x000000008001003b, CR2: 0xffffff81210ea000, CR3: 0x00000001b8fd30b7, CR4: 0x00000000003627e0
RAX: 0x0000000000000001, RBX: 0xdeadbeefdeadbeef, RCX: 0x0000000000000059, RDX: 0x0000000000000008
RSP: 0xffffff9122e3bae0, RBP: 0xffffff9122e3bae0, RSI: 0xffffff803197ee34, RDI: 0xdeadbeefdeadbeef
R8:  0x0000000000000004, R9:  0x0000000caca00d2a, R10: 0x8000000caca00d2a, R11: 0x000a0d200caca000
R12: 0xffffff803197ee34, R13: 0x00000000e00002c2, R14: 0xffffff803197ee34, R15: 0xffffff80323fd600
RFL: 0x0000000000010202, RIP: 0xffffff80165c4e90, CS:  0x0000000000000008, SS:  0x0000000000000010
Fault CR2: 0xffffff81210ea000, Error code: 0x0000000000000000, Fault CPU: 0x0, PL: 0

Backtrace (CPU 0), Frame : Return Address
0xffffff80f8808c50 : 0xffffff80164dab12 mach_kernel : _panic + 0xe2
0xffffff80f8808cd0 : 0xffffff80165ce40a mach_kernel : _kernel_trap + 0x91a
0xffffff80f8808eb0 : 0xffffff80165ec273 mach_kernel : _return_from_trap + 0xe3
0xffffff80f8808ed0 : 0xffffff80165c4e90 mach_kernel : _memcmp + 0x10
0xffffff9122e3bae0 : 0xffffff7f989e71d4 com.apple.driver.AppleIntelBDWGraphics :
__ZNK11IGHashTableIyP16IGAccelMemoryMap12IGHashTraitsIyE25IGIOMallocAllocatorPolicyE8containsERKy + 0x42
0xffffff9122e3bb00 : 0xffffff7f989e56cc com.apple.driver.AppleIntelBDWGraphics :
__ZN16IGAccelCLContext15map_user_memoryEP22IntelCLMapUserMemoryInP23IntelCLMapUserMemoryOutyPy + 0x66
0xffffff9122e3bb50 : 0xffffff8016ae1592 mach_kernel : _shim_io_connect_method_structureI_structureO + 0x122
0xffffff9122e3bb80 : 0xffffff8016ae220a mach_kernel :
__ZN12IOUserClient14externalMethodEjP25IOExternalMethodArgumentsP24IOExternalMethodDispatchP8OSObjectPv + 0x34a
0xffffff9122e3bbe0 : 0xffffff8016adf277 mach_kernel : _is_io_connect_method + 0x1e7
0xffffff9122e3bd20 : 0xffffff8016597cc0 mach_kernel : _iokit_server + 0x5bd0
0xffffff9122e3be30 : 0xffffff80164df283 mach_kernel : _ipc_kobject_server + 0x103
```

KEEN security lab

Overwriting free'd element's next pointer

```
Anonymous UUID:       D09DE92C-8710-4673-953D-BACF9F5B3C09

Thu Mar 24 01:34:03 2016

*** Panic Report ***
panic(cpu 2 caller 0xffffff800931f92b): "a freed zone element has been modified in zone kalloc.32: expected 0xdeadbeefdeadbeef but
found 0xffffff8029eb73a0, bits changed 0x2152416ff746cd4f, at offset 16 of 32 in element 0xffffff8029eb7440, cookies
0x3f0011330a841290 0x53521934cf94203"@/Library/Caches/com.apple.xbs/Sources/xnu/xnu-3248.40.184/osfmk/kern/zalloc.c:503
Backtrace (CPU 2), Frame : Return Address
0xffffff810b7a2a80 : 0xffffff80092dab12 mach_kernel : _panic + 0xe2
0xffffff810b7a2b00 : 0xffffff800931f92b mach_kernel : _zone_find_largest + 0x8fb
0xffffff810b7a2c30 : 0xffffff800983ca36 mach_kernel : __ZN6OSData16initWithCapacityEj + 0x66
0xffffff810b7a2c50 : 0xffffff800983cab0 mach_kernel : __ZN6OSData13initWithBytesEPKvj + 0x30
0xffffff810b7a2c80 : 0xffffff800983cc4e mach_kernel : __ZN6OSData9withBytesEPKvj + 0x6e
0xffffff810b7a2cb0 : 0xffffff800985d475 mach_kernel : __Z21OSUnserializeXMLparsePv + 0x13f5
0xffffff810b7a3d40 : 0xffffff800985db76 mach_kernel : __Z16OSUnserializeXMLPKcPP8OSString + 0xc6
0xffffff810b7a3d70 : 0xffffff80098de1da mach_kernel : _is_io_service_open_extended + 0xfa
0xffffff810b7a3de0 : 0xffffff80093977a1 mach_kernel : _iokit_server + 0x56b1
0xffffff810b7a3e30 : 0xffffff80092df283 mach_kernel : _ipc_kobject_server + 0x103
0xffffff810b7a3e60 : 0xffffff80092c28b8 mach_kernel : _ipc_kmsg_send + 0xb8
0xffffff810b7a3ea0 : 0xffffff80092d2665 mach_kernel : _mach_msg_overwrite_trap + 0xc5
0xffffff810b7a3f10 : 0xffffff80093b8bda mach_kernel : _mach_call_munger64 + 0x19a
0xffffff810b7a3fb0 : 0xffffff80093eca96 mach_kernel : _hndl_mach_scall64 + 0x16

BSD process name corresponding to current thread: fuckaddremovebdw
Boot args: keepsyms=1

Mac OS version:
15E65

Kernel version:
```

科恩实验室 KEEN security lab

Questions?

Credits

- Liang Chen
- Marco Grassi
- Wushi

Thanks!

科恩实验室
KEEN
security lab