OS X/iOS mach*ports*register函数中的多个内存问题

Reported by ianbeer@google.com, Jul 27 2016

mach*ports*register函数是一个内核任务端口的MIG方法。

它在MIG中定义类似如下所示：

```
routine mach_ports_register(
    target_task : task_t;
    init_port_set : mach_port_array_t =
     ^array[] of mach_port_t);
```

从生成的代码中，我们发现一些奇怪的东西。它实际发送的mach消息的结构体如下所示：

```
typedef struct {
  mach_msg_header_t Head;
  // 内核处理数据开始
  mach_msg_body_t msgh_body;
  mach_msg_ool_ports_descriptor_t init_port_set;
  // 内核处理数据结束
  NDR_record_t NDR;
  mach_msg_type_number_t init_port_setCnt;
} Request __attribute__((unused));
```

消息中包含一个OOL端口描述符的同时，却还有一个相对独立的**init*port*setCnt**，即使**ool*ports*descriptor_t**已经包含了描述符的长度。

内核在**ipc*kmsg*copyin*ool*ports_descriptor**处理OOL端口描述符时，它将为所有端口分配一个足够大的缓冲区。 它使用的是init*port*set.count的值，而不是init*port*setCnt。

调用**mach*ports*register**生成MIG代码的方法如下:

```
  OutP->RetCode = mach_ports_register(target_task, (mach_port_array_t)(In0P->init_
port_set.address), In0P->init_port_setCnt);
```

这里没有去验证**In0P-> init*port*setCnt**是否等于**init*port*set.count**。

这意味着当我们调用**mach*ports*register**时，将出现很多问题，如下面例子所示：

```
kern_return_t
mach_ports_register(
  task_t      task,
```

```
    mach_port_array_t memory,                           <-- 分配的buffer指针
    mach_msg_type_number_t  portsCnt)                   <-- 与buffer无关且可控
  {
    ipc_port_t ports[TASK_PORT_REGISTER_MAX];
    unsigned int i;

    if ((task == TASK_NULL) ||
        (portsCnt > TASK_PORT_REGISTER_MAX) ||
        (portsCnt && memory == NULL))
      return KERN_INVALID_ARGUMENT;                     <--  portsCnt必须 >=1 && <=3

    for (i = 0; i < portsCnt; i++)
      ports[i] = memory[i];                             <-- 如果我们发送1个OOL端口，但是po
rtsCnt>1，则会造成指针越界
    for (; i < TASK_PORT_REGISTER_MAX; i++)
      ports[i] = IP_NULL;

    itk_lock(task);
    if (task->itk_self == IP_NULL) {
      itk_unlock(task);
      return KERN_INVALID_ARGUMENT;
    }

    for (i = 0; i < TASK_PORT_REGISTER_MAX; i++) {
      ipc_port_t old;

      old = task->itk_registered[i];
      task->itk_registered[i] = ports[i];
      ports[i] = old;
    }

    itk_unlock(task);

    for (i = 0; i < TASK_PORT_REGISTER_MAX; i++)
      if (IP_VALID(ports[i]))
        ipc_port_release_send(ports[i]);            <-- this can decrement the ref
on a pointer which was read out of bounds if we call this function multiple times

        <--  我们可以通过调用多次来减少这个越界指针的引用

    if (portsCnt != 0)
      kfree(memory,
            (vm_size_t) (portsCnt * sizeof(mach_port_t)));   <-- 可以释放错误大小的bu
ffer

    return KERN_SUCCESS;
  }
```

我已经修改了这个Poc的MIG生成代码，使它每次只发送一个OOL mach端口，并且将**init*port*setCnt**设置为一个可控的值，你应该会发现因为无效指针的引用或其他类似的原因而导致的内核错误减少。

这个bug可以被很好地利用来引发**mach*port*t**的UaF，这可能有各种有趣的后果（例如获取另一个任务的任务端口）。

测试环境： OS X 10.11.6 (15G31) MacBookPro10,1

译者：莫云溪
原文链接：https://bugs.chromium.org/p/project-zero/issues/detail?id=882
原文作者：ianbeer@google.com