OS X IOBluetoothFamily.text 中的内核 UAF 漏洞

译者：布兜儿(看雪 ID：我有亲友团)

原文链接：https://bugs.chromium.org/p/project-zero/issues/detail?id=830

原文作者：ianbeer@google.com

OS X IOBluetoothFamily.text 中的内核 UAF 漏洞

当你从用户空间创建一个新的 IOKit 用户端时，将会调用：

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

参数 owningTask 端口通过 MIG 反序列化代码转化为一个任务结构指针，然后获得对任务的引用，调用 is_io_service_open_extended 时传入 task 结构体，在这之后丢弃这个引用。

is_io_service_open_extended 接着会调用该服务重写的 newUserClient 或者 initWithTask 的方法的实现。

如果这些服务想要保留指向 owningTask 的指针，那么就有必要实际持有一个引用。

实际上我们可以让任意一个任务端口作为 owningTask，也就是说，如果用户端没有持有引用，我们可以非常轻松的通过任务端口进行另一项任务，关闭这个任务(释放任务结构)，然后通过用户端来使用被释放的任务结构。

IOBluetoothHCIUserClient(IOBluetoothHCIController 的 type 0 用户端)可以在没有引用的前提下，由一个普通用户来实现，并在当前地址 + 0xe0 处存储一个原始的任务结构指针。

这个指针将在稍后被用于 IOBluetoothHCIUserClient::SimpleDispatchWL 中，来构建和操作 IOMemoryDescriptors。

此 POC 会 fork 一个子进程，并向父进程发送它的端口号，然后转向。父进程将会创建一个新的 IOBluetoothHCIUserClient，将子进程的任务端口作为 owningTask 参数来传递，然后杀掉子进程(释放它的任务结构)。接着父进程在用户端调用一个外部方法，导致 UAF。

IOMemoryDescriptor 的代码在处理任务结构方面做了一些奇怪的事情，并且内存映射释放它，使得这个 bug 成为一个简单的可利用的内存崩溃问题，并且可以被用来做一些更有趣的逻辑处理 。

注意：必须打开蓝牙，这个 POC 才有效。

build: clang -o bluetooth_uaf bluetooth_uaf.c -framework IOKit

你可以设置 gzalloc_min=1024，gzalloc_max=2048 或者类似于 UAF 的实际错误，否则可能会得到一些奇怪的崩溃 。

已在 MacBookAir5,2 ， OS X 10.11.5（ 15F34 ）上测试。

注：附上 [Exploit] (https://www.exploit-db.com/exploits/40652/)