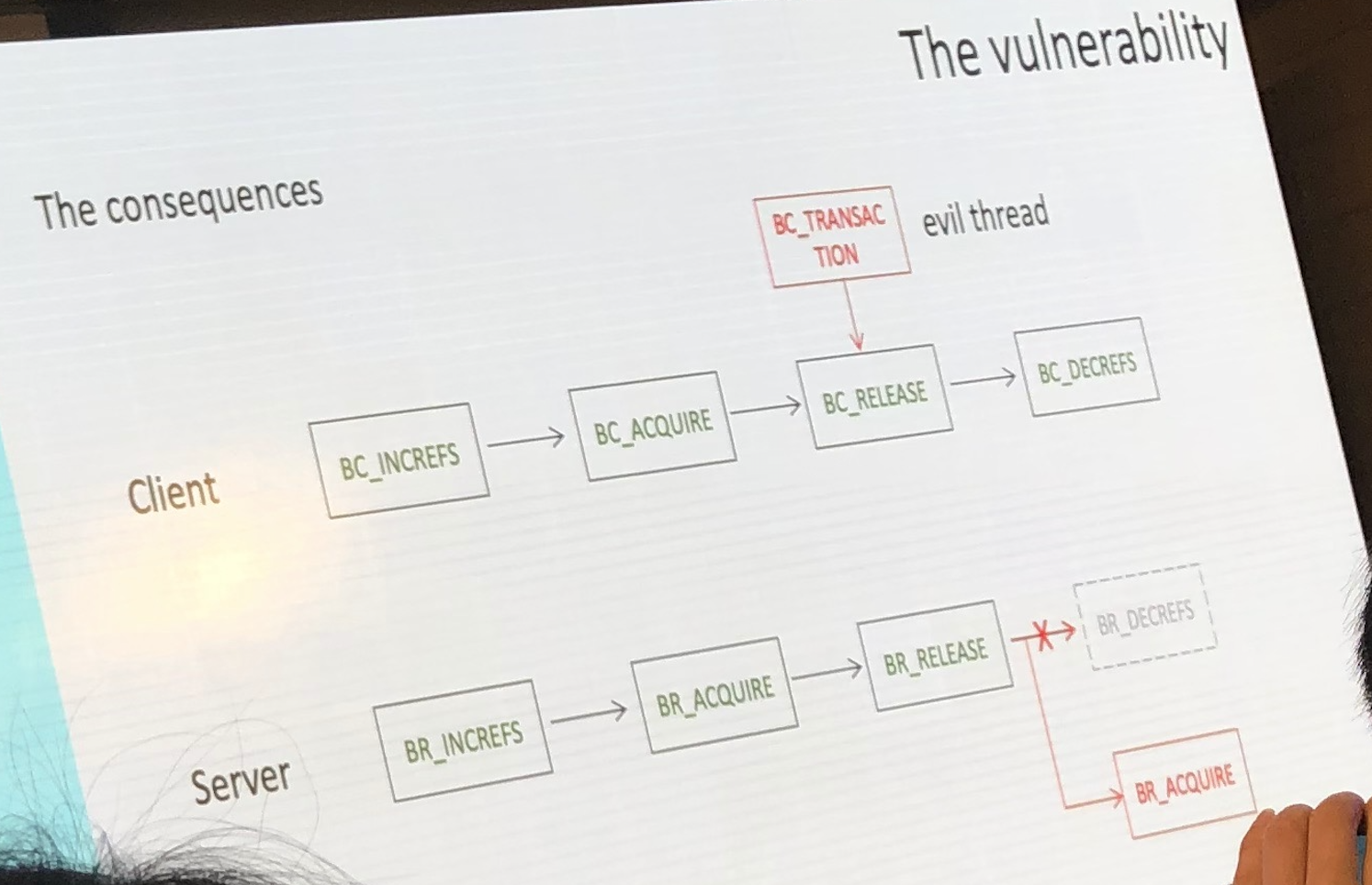
MOSEC day2

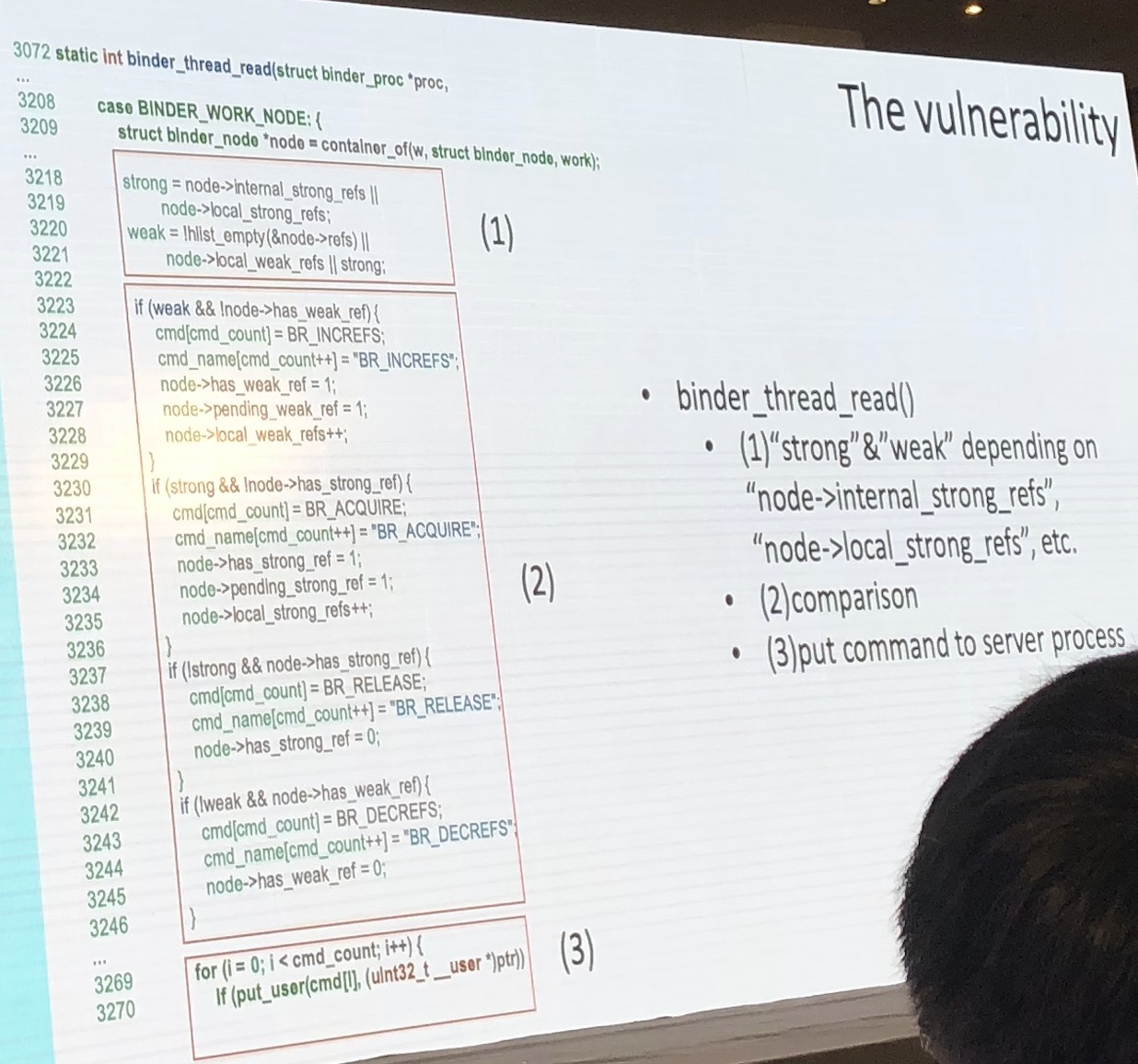
# 无处安放的shellcode

**主要内容：**

绕过用户态的防护措施来实现提取安卓权限，研究并实现了一种全新的用户态进程攻击方法，借助JavaVM解释器执行恶意Java字节码。由于Java字节码是以数据的形式存放，依靠Java的解释执行机制便不再需要可执行内存了，打破了目前SELinux防护策略的封堵。

首先分享了一个Android下Binder中的一个use-after-free漏洞以及利用技巧。





该漏洞影响系统上所有Binder相关的服务。利用时可以通过dalvik中的registerReceiver调用来进行堆喷射，并且构造对象就可以控制服务端的PC寄存器。

Heap spraying、string 用户态 堆喷、Memory layout、 得到一个可控的划窗 两次跳转 得到相对地址， 因为dalvik-free里地址相对固定，string metadata。mmap or mprotect shellcode with PORT+EXE，拿到权限。

但是Android N之后就不行了，从Android N开始，一系列的SELinux策略被引入，用来限制在用户态进程中创建可执行内存。这使得传统映射可执行内存来执行shellcode的方式变得困难。

SELinux策略

* system\_server should never use JIT functionality
* neverallow system\_server self:process execmem;
* neverallow system\_server ashmem\_device:chr\_file execute;
* neverallow system\_server{

data\_file\_type

-dalvikcache\_data\_file

} 对初始化的内存；（ 从 Android O 开始这个也被禁止映射可执行）

但是Android下的大部分系统服务包括system\_server都支持解析JVM生成的bytecode， 这带来的新思路就是**用Java编写shellcode所需要使用的功能，并且通过ROP构造Execute SwitchImpl函数的参数**来达到执行任意的Java代码的目的。

**ExecuteSwitchImpl<false, true>(self, code\_item, shadow\_frame, result\_register,false)**

**用户态利用的思路**

使用内容可控的数据+内容不可控的代码

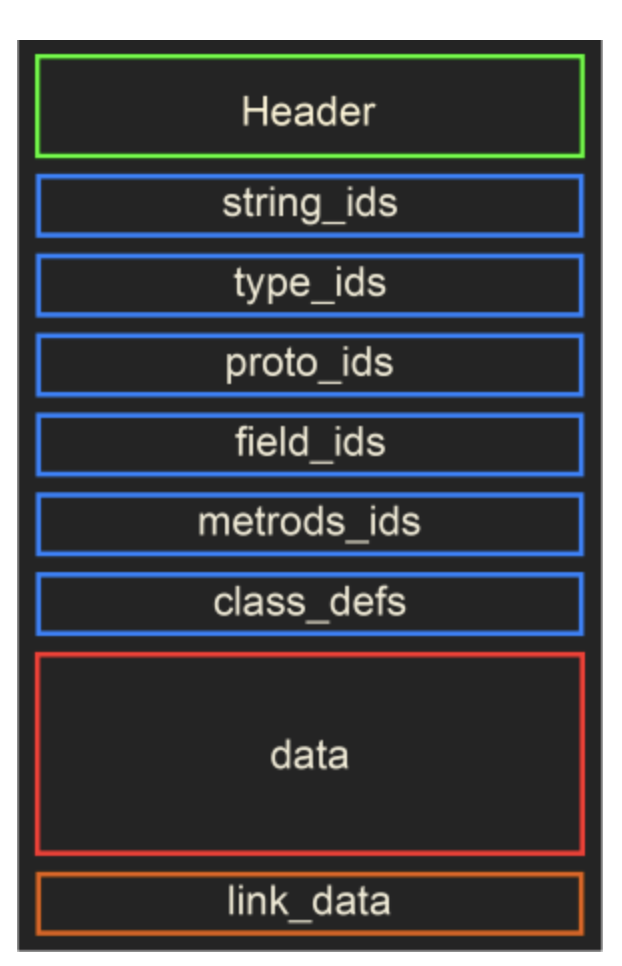
Java代码 + Dex

**得到相对位移地址**

了解DEX解析过程:dex索引查找方法( dex\_cache\_resolved\_methods 指向索引表（已经调用过的方法）、declaring\_class->找class 这样找Dex\_cache(所有有的))， dex查找string or type、 daclaring class dex\_cache。

构造PoC填充相应内容

|\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_|



**得到基地址**

在dalvik的 interpreter（拿指针的第三个参数）的 ShadowFrame的

artMethod\*method\_declaring\_class 是32bit 位置(所有地址在低地址空间)这个是 安卓自带信息泄漏 （ zygote space（加载过的的类，每个base地址都是一样））

ExecuteMterpImpl(self, code\_item, &shadow\_frame, &result\_register);

**局限于:** local application info leak

**=======基础知识=========**

Binder是基于OpenBinder实现的，是Android系统的重要组成部分。Binder构建于Linux内核中的Binder驱动之上，系统中涉及到Binder通信的部分，都需要通过与Binder驱动交互来实现数据传递。Binder驱动也是目前为数不多的几个可被沙箱中的进程访问的驱动之一，该漏洞可能被用于沙箱逃逸。

**=======基础知识=========**

**=======水滴漏洞=========**

当用户进程通过调用“IPCThreadState::transact()”向server进程请求数据时，server进程会调用”IPCThreadState::sendReply()”向用户进程回传数据以作为响应。在这个过程中，server进程会陷入内核，并调用“binder\_alloc\_new\_buf()”从用户进程（“target\_proc

->alloc”）对应的”alloc->free\_buffers.rb\_node”红黑树中申请一个”struct binder\_bu

ffer”类型的对象，被申请得到的对象会从”alloc->free\_buffers.rb\_node”红黑树中移除，并链入到”alloc->allocated\_buffers.rb\_node”红黑树中进行维护，此后将”t->buffer->

allow\_user\_free”赋值为0，以避免”t->buffer”在使用过程中被释放。

用户进程拥有对该”t->buffer”的释放权，不过要在用户进程收到了server进程发来的消息，结束本次交互并销毁所用的Parcel对象时。然而，我们可以试图通过主动向内核发送”BC\_FREE\_BUFFER”请求来提前释放该”t->buffer”对象，不过这需要满足内核对各项参数的校验。

上文中提到的“t->buffer”可以通过向内核传入对应的“data\_ptr”值，再通过调用“binder\_alloc\_prepare\_to\_free()”函数来找到，其返回值会赋值给”buffer”，紧接着内核会检查“buffer->allow\_user\_free”以及”buffer->transaction”的合法性，由于缺少互斥锁的保护，这里存在一个条件竞争问题，如果用户进程能在”t->buffer->allow\_user\_free”被赋值为0之前触发到“binder\_alloc\_free\_buf()->binder\_free\_buf\_locked()->rb\_erase()”，便有可能将其从”alloc->buffers”中移除。

之后便可再选择合适时机触发”kfree()”将其释放，而此时当server进程继续使用”t->buffer”时便触发了use-after-free问题。

**=======水滴漏洞=========**

**相关链接：**http://blogs.360.cn （binder水滴漏洞）

**演讲者：**

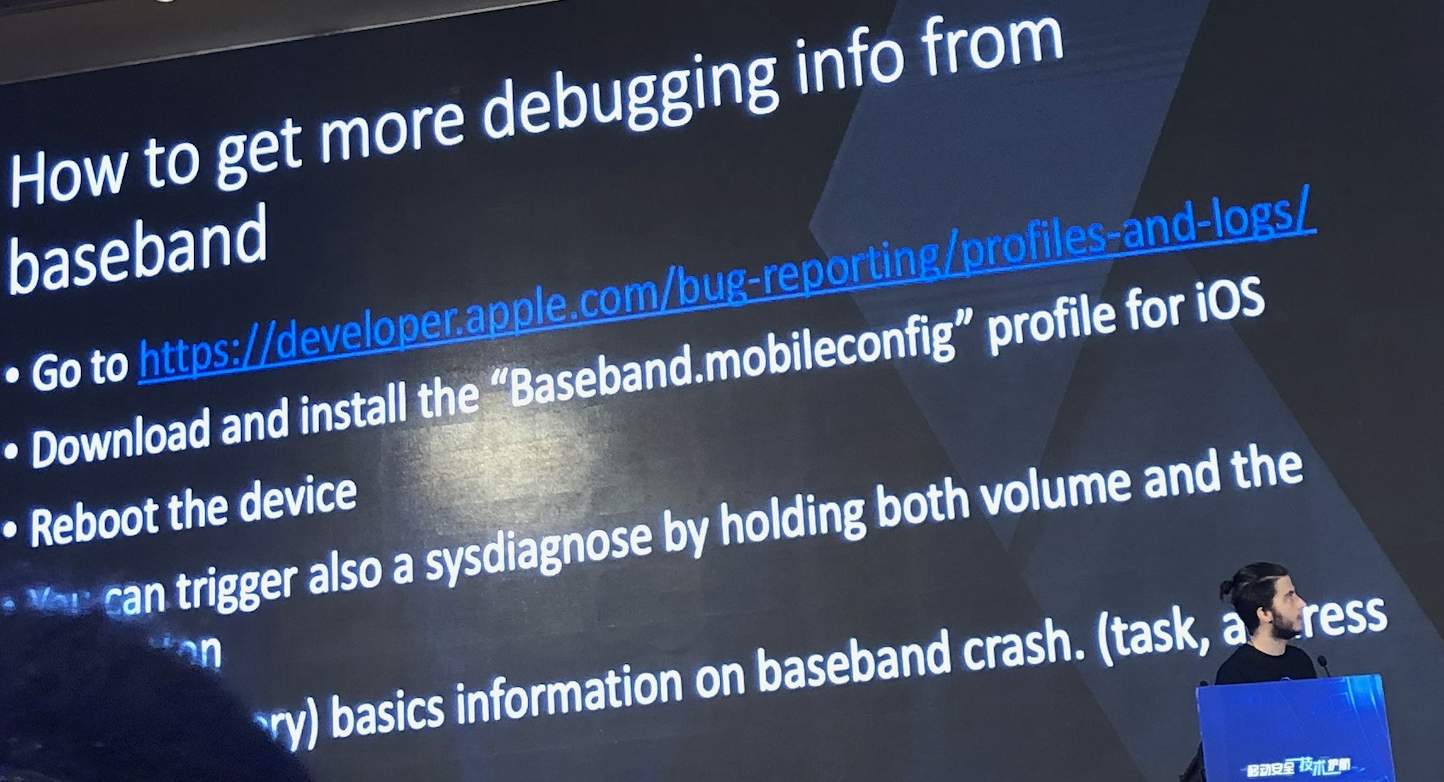
张弛 奇虎360 C0RE Team的一名安全研究员，主要从事安卓框架层的漏洞挖掘与利用。

韩洪立 奇虎360 C0RE Team的一名安全研究员，专注于安卓操作系统以及Linux内核的安全研究。在过去的几年里，他向谷歌、高通、英伟达、华为等厂商上报了数十个致命/高危漏洞，并得到了公开的致谢。

# 新一代移动网络的基带的崛起

**主要内容：**

第一部分分享了逆向最新iPhone中Intel架构的基带芯片的一些**心得体会**，通过逆向找到内存分配相关的代码再定位到进程和线程相关的处理函数，再通过创建线程相关的代码来定位和基带通讯处理的函数进行后续的代码审计。随后又介绍了一些iOS基带芯片相关的**攻击面**，**基带和内核交互的PCI-E接口代码、CommCenter和内核通讯的IOKit代码**。



Macro的第二部分分享主要是针对对Android基带芯片的一些研究，以华为P20为例，已经在基带芯片加入了诸如**Stack Cookies**之类的安全保护，相信在不久的未来，基带的芯片应该很快会引入**地址随机化保护（ASLR）**，并且在新的硬件架构下也有可能会引入类似**ARM下PaC/Intel CET之类的的硬件保护机制**。

**演讲者：**Marco Grassi 腾讯科恩实验室 高级研究员

# 钉枪：突破ARM特权隔离

**主要内容：**

首先分析了传统的调试模式下的物理的调试方式。但是在多处理器的模型下使得这种调试接口变得不太便利。因此，ARM引入了一个新的调试模型，在这种新的调试模型中，主机处理器能够暂停和调试同一芯片上的另一个目标处理器(处理器间调试)。这种调试模式**不依赖物理访问，也不需要通过JTAG接口**。但是ARM同时也引入了对应的调试验证机制，通过四种不同的信号来控制不同的调试模式。

**钉枪攻击的思想**是在多处理器的SoC下，利用这种处理器间调试能力，通过低权限处理器暂停和调试高权限的目标处理器。在调试模式下可以通过执行特权指令DCPS来提升处理器的权限，并且绕过ARM下的权限隔离机制。这种攻击可以读取系统的安全配置寄存器，甚至能够在trustzone中执行任意代码。在调试模式下可以直接把代码拷贝到trustzone的安全内存中，修改VBAR\_EL3寄存器中指向的异常向量表，达到在trustzone执行任意代码的目的。

通过这种方法，两位演讲者成功演示了提取存储在华为Mate 7安全内存中的指纹图像。

**演讲者：**张锋巍教授及宁振宇博士韦恩州立大学

# 畅游EL3:终极提权之旅

**主要内容：**

（以解锁的方式去root）

首先和大家分享了Android系统的trustzone模型，包括设备会被划分成四级特权(EL0-EL3)，以及可信和不可信区域(secure/non-secure)。通常操作系统内核权限是运行在不可信的NS-EL1区域，无法访问全部内存和外设，因为这是一个权限隔离的区域。

闻观行在接下来演示了逆向华为手机的trustzone，包括定位到某个smc调用在VBAR\_EL3中的异常向量表中的处理函数，并且通过逆向该处理函数**找到了一个可以在内核中触发的漏洞**，由于该处理函数中的某个虚函数指针是从内核和trustzone的共享内存中读取的，导致在内核权限下可以直接设置这个指针，然后在trustzone运行的EL3下执行任意代码。

在利用这个漏洞的时候，**他选择了跳转到trustzone的某个函数达成了任意内存读写的目的**，在执行shellcode时候他选择先通过ROP修改存放shellcode页表的属性，刷新TLB然后执行最终的代码，最后闻观行还演示了通过这个漏洞绕过了手机中的脸部识别功能。

**演讲者：**闻观行，盘古实验室的安全研究员，他目前关注的是安卓底层的漏洞分析和利用。

# 一些JSC的故事

**主要内容：**

最后的一个议题关于JavaScriptCore引擎的安全研究。JSC是WebKit浏览器中负责解释执行的Javascript的引擎，是苹果操作系统中默认的浏览器引擎。在这次的演讲中，Luca首先和大家分享了Webkit漏洞的历史变迁，包括早期DOM中的use-after-free类型的漏洞以及漏洞利用方式，包括ROP及改写JIT函数指针达到执行任意代码的目的。因此，苹果也增加了很多安全保护机制，包括对DOM对象的隔离及浏览器JIT内存的加固，内存牢笼及内存指针posion等等。

他以JIT DFG对StringSlice优化过程中存在的寄存器错误分配的问题为例子，并且成功的把这个漏洞转换成的Side Effects的问题，通过**回调toString把一个浮点数组混淆成变量数组，然后在该处构造虚假对象来达到任意内存读写的目的**，这也是目前浏览器中主流的利用方式。

**相关链接：**iokit.racing/jsctales.pdf （ppt）

**演讲者：**Luca（@qwertyoruiopz）是一位来自意大利的安全研究人员，研究iOS安全