SLAKE文章的工作

总体来说解决/回答了这样一个问题：哪些内核中的结构体可以用来做控制流劫持，以及如何分配这些结构体和如何触发控制流劫持（一个系统调用系列）。此外还讨论了如何控制内核堆布局的问题。

关于我之前考虑的将一个漏洞的特征总结为模式的想法，在文章中也有实现。文章中将这种模式称为漏洞的capacity，建模如下：漏洞capacity描述为一个元组列表A，每个元组$A[i]$记录可以写的字节起始偏移和终止偏移。同时对每个找到的可用的结构体建模，描述为一个列表C，列表中每个元素$C[i]$记录结构体中函数指针的偏移。这样建模的目的主要是为了做漏洞capacity和结构体的匹配。

文章的总体实现方法如下：静态分析寻找所有包含函数指针的内核结构体（或包含指向包含函数指针的结构体的指针），并识别这些结构体的分配点和其中函数指针解引用点（过滤不可被系统调用到达的点），然后做fuzz判断静态分析识别到的这些点实际能否被系统调用触发。

文章的结果：在内核的核心部分（allnoconfig）共找到20余个可用结构体，在包含网络子系统在内等（常用）子系统内共找到100余个可用结构体即其分配、控制流劫持系统调用序列。

关于文章的思考

从结果上来看，包含常用子系统在内，识别到的可用的结构体仅约100左右，与我直觉上的数字有些出入。且在SLAKE的github项目中作者并没有公布这些结构体及对应系统调用的具体信息。

从设计上来看，SLAKE只关注了结构体中的函数指针及其可能的利用，而完全没有考虑这些利用模式：1.由于结构体关键数据篡改导致的利用；2.由结构体中除函数指针外其他指针导致的任意内存读写或可能的控制流劫持等利用。诚然，劫持结构体中函数指针可以直接实现控制流劫持，是完成利用的最直接的方式，这样的利用模式也更容易实现和发现，但诸如任意内存读写等利用在堆相关的利用技术中也非常常见。

从问题范围上看，SLAKE没有解决以下两个问题：1.如何将一个漏洞capacity自动建模；2.实现控制流劫持后如何进一步完成利用。关于前者，SLAKE在工作展望中有提及，但是否有具体进展我没有进一步查找相关论文。关于第二点，SLAKE在其问题范围描述部分将其称为trivial，并提到了吴炜师兄的KEPLER，但KEPLER中主要关注kernel中类似one_shot的利用方式，是所有利用方式空间中的一个小子集。这与我此前考虑的并不是同一个问题，但显然这个问题并不trivial，仅实现控制流劫持并不能直接完成利用。

目前可以做或者可以进一步思考和关注的问题如下：

1. 考试结束后我打算向SLAKE作者询问一下关于SLAKE发现的100余个内核结构体及相应系统调用的具体信息，写一个匹配漏洞模式和内核结构体的小程序，这是我思考这个问题的初衷。如果这些结构体质量很高，对于今后利用内核漏洞很有帮助，同时在这个过程中可能会发现一些其他问题
2. 除了通过劫持内核结构体中函数指针的利用方式外，还有很多利用技术通过内核结构体中其他指针的劫持实现任意内存读写，或通过修改内核结构体关键数据，或通过一些更加巧妙的利用方式（如work_for_cpu_fn），这样的利用模式在内核中还存在多少？能否被自动发现？
3. 关于自动对漏洞capacity进行建模的工作现在进展如何？