Linux kernel vulnerabilities: State-of-the-art defenses and open problems

论文内容

论文首先指出，操作系统内核是可信计算基的组成部分。并且，论文提到，内核中的漏洞会导致严重的问题，包括内核保护绕过和root权限获取。然而，论文指出，目前已经有很多研究尝试减少内核中的漏洞，但内核漏洞及其利用仍然在Linux（一个常见的操作系统内核）中出现。本文通过分析和分类已被发现的Linux的漏洞，来发现目前研究的关注点。同时，本文还研究了防御手段和一些（未解决的）开放问题。此外，论文提到，Arnold等人指出，每一个内核的bug都应当被认为是安全关键的，并且应当尽快修复。Mokhov等人研究了内核开发者如何修复已被发现的漏洞。

本文的研究和发现主要有如下几点：

1. 以Linux内核中的漏洞为例子，论文研究了内核开发者编程时出现的错误以及这些错误对安全的影响。
2. 论文研究了已经提出的用于缓解内核漏洞的方法的有效性，这些方法包括代码完整性检查、软件错误隔离和用户态驱动程序等。
3. 论文发现，暂无单一方法可以阻止所有的内核漏洞；并且，用于缓解某个漏洞的特定利用方法的方法并不能缓解其他利用漏洞的方法。
4. 论文发现，某类漏洞根本没有得到解决，例如语义漏洞无法用state-of-the-art的技术来发现，因为这类技术目前主要关注于内存安全和代码完整性。

Linux内核漏洞

首先，论文列举了漏洞的可能的利用方式，包括内存损坏、策略违例、拒绝服务以及信息泄露等。

然后，论文详细介绍了Linux内核中的漏洞，漏洞主要分为如下几种，论文统计了它们出现的次数，以及可能的利用方法的次数。

缺少指针检查

内核没有检查来自用户的指针或数组下标是否真正指向用户有访问权的区域。此类漏洞可能导致任意地址内存读写，进而导致内存损坏、信息泄露、拒绝服务或权限提升等攻击。

缺少权限检查

内核在执行特权操作时，没有检查调用进程的权限。此类漏洞可能导致任意代码执行或权限提升等攻击。

缓冲区溢出

内核访问缓冲区时，错误地检查了边界；错误地分配了较小的缓冲区；使用了不安全的字符串处理函数；局部变量大小超过了栈大小。此类漏洞可能导致内存损坏、信息泄露或权限提升等攻击。

整型溢出

内核进行了错误的整数操作，导致出现上溢、下溢或者符号错。此类漏洞可能间接导致缓冲区溢出漏洞类似的那些攻击。

未初始化的数据

内核在复制缓冲区内容到用户空间时，没有对缓冲区中未使用的部分进行清零操作，从而将部分数据泄漏到用户空间。此类漏洞可能泄漏内核中的密钥或者内核的随机性。

错误的内存管理

包括内存泄漏、double-free漏洞以及use-after-free等。此类漏洞可能导致拒绝服务。

杂项

内核中的其他漏洞。

State-of-the-Art预防措施

论文按照运行时和编译期的分类，介绍了几种目前state-of-the-art的预防措施，也指出了他们的不足。

运行时工具

这类工具主要是在运行时动态检查。

软件错误隔离

BGI是一个工具，可以通过设置ACL来控制内核模块对内核内存数据的访问。不足是只能缓解内核模块内部的问题，而不能缓解内核本身的问题；同时，也不能缓解那些可以完全在内核模块中被利用的漏洞。

代码完整性检查

SecVisor用于检查内核代码完整性，拒绝执行攻击者注入的代码。然而，攻击者可以修改内核代码控制流，并且文献[23]指出，可以利用已有的内核代码片段，来进行任意运算。因此，SecVisor仅仅能够使得漏洞难以利用，而不能彻底防御。

用户态驱动程序

SUD使得驱动程序在用户态运行。一个不足之处为，用户态驱动程序并不能够防御拒绝服务攻击。

内存标签

内存标签系统可以检测出内核对于不可信输入的错误使用，可以防御一些代码注入漏洞，但不能防御其他大多数漏洞。

未初始化内存跟踪

Kmemcheck可以跟踪内存中每一个字节的初始化状态，但不能检测对于未初始化内存的读取。SD能够定期清零内核栈，但是不对动态分配的对象进行处理。这两个工具都不能保证它们能够发现和防御所有的信息泄露漏洞。

编译期工具

这类工具主要是静态检查。论文指出，已有很多工具用来查找和修复Linux内核中的各种安全问题。静态方法的一大问题是假阳性（误报）更多。

开放问题

论文在本节首先提到，用于缓解某个漏洞的特定利用方法的方法并不能缓解其他利用漏洞的方法。论文在本节还提到，Linux是用C语言写成的，而C语言不是一个类型安全的语言。虽然用类型安全的语言来开发操作系统内核是好事，但是若要复用已有的Linux内核代码，仍然需要使用C语言。此外，其他内核设计架构（如微内核），例如HiStar或Minix，可以减少内核中必须可信的代码的数量，但由于Linux是宏内核，这种模式无法应用到Linux。接

着，论文阐述了另外两种开放问题：

语义漏洞

论文中举了几个例子，均为在进行某些操作时缺少了权限检查，更一般地可描述为，没有实现接口定义的所有行为。我认为，接口定义本身可能也存在漏洞。

拒绝服务漏洞

论文还简要介绍了拒绝服务漏洞可以和其他漏洞组合利用来造成更严重的结果，同时介绍了拒绝服务漏洞防御的难点。难点主要在于，当内核拒绝服务被触发时，如何正确地恢复内核的状态（即保持内核应有的不变式）。

创新点、亮点或其他印象较深的部分

论文提到，每一个内核的bug都应当被认为是安全关键的，并且应当尽快修复。

论文发现，暂无单一方法可以阻止所有的内核漏洞；并且，用于缓解某个漏洞的特定利用方法的方法并不能缓解其他利用漏洞的方法。

文献[23]指出，可以利用已有的内核代码片段，来进行任意运算。

论文提到了语义漏洞，这类漏洞在之前听说得较少。

反思与讨论

论文提到，Arnold等人指出，每一个内核的bug都应当被认为是安全关键的，这一点十分有趣且引人深思，需要进一步阅读文献[3]来学习。

论文发现，暂无单一方法可以阻止所有的内核漏洞，这一点容易理解。但论文指出，用于缓解某个漏洞的特定利用方法的方法并不能缓解其他利用漏洞的方法，这一点需要进一步思考，为什么会这样。

关于”未初始化的数据“漏洞，此前在QQ群聊天时了解到，内核中对于结构体整体的赋值，大多使用memcpy或赋值操作符（可能也会被编译器编译为memcpy），由此，结构体中未被使用且未被初始化的paddings会被直接复制，很可能导致数据泄漏。网友表示此类缺陷在内核中时有发生。需要查找相关文献来进一步确认和了解。

文献[23]指出，可以利用已有的内核代码片段，来进行任意运算，这一点很有意思，需要进一步阅读。

论文提到了语义漏洞，而这类漏洞我在之前接触的远比其他漏洞少，需要进一步深入学习。

值得注意的是，本文是2011年发表的，而现在是2018年，因此可能出现了一些新类型的漏洞，需要去调查和学习。此外，Hennessy与Patterson指出软硬件结合是发展方向，那么软硬件结合的过程中是否会产生一些新的漏洞，或者操作系统内核能否缓解一些硬件的漏洞（比如几个月前的处理器的安全漏洞），都需要思考。

此外，论文中多次使用“... is based on our understanding of the techniques”，可以看出作者十分严谨，值得学习。

扩展阅读

[3] J. Arnold, T. Abbott, W. Daher, G. Price, N. Elhage, G. Thomas, and A. Kaseorg. Security impact ratings considered harmful. In *Proceedings of the 12th Workshop on Hot Topics in Operating Systems*, Monte Verita, Switzerland, May 2009.

[23] H. Shacham. The geometry of innocent flesh on the bone: Return-into-libc without function calls (on the x86). In *Proceedings of the 14th ACM Conference on Computer and Communications Security*, Alexandria, VA, October–November 2007.

语义漏洞的资料

结束语

总之，论文以Linux内核中的漏洞为例子，研究并阐述了内核开发者编程时出现的错误种类以及这些错误对安全的影响的种类。此外，论文研究并阐述了已经提出的用于缓解内核漏洞的各种方法的有效性，指出暂无单一方法可以阻止所有的内核漏洞；并且，用于缓解某个漏洞的特定利用方法的方法并不能缓解其他利用漏洞的方法。最后，阐述了几个开放的问题。我在阅读时对内核代码片段能够被利用来执行任意运算以及语义漏洞等几点子问题产生了兴趣，并对当下可能出现的新漏洞类型进行了简单思考。