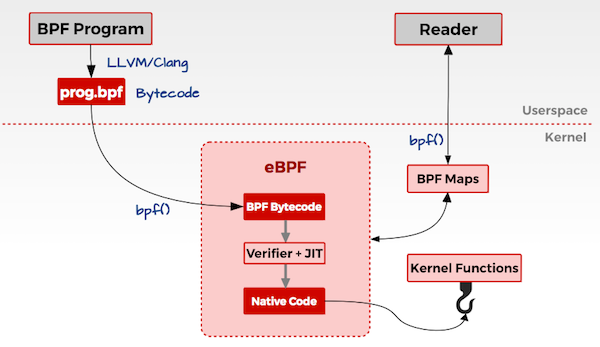
G.O.S.S.I.P 阅读推荐 2023-06-28

2023-06-28 20:29

知名的Linux新闻网站Linux Weekly News（也就是我们熟悉的lwn.net）的主编Jonathan Corbet在今年6月8日发表了一篇名为 Yet another memory allocator for executable code 的文章，讨论了Linux内核中那些动态分配、同时支持代码执行的内存页的管理问题。

为什么内核需要可执行的动态内存呢？过去几年中火爆的BPF/eBPF子系统就是一个典型的需要生成JIT代码执行的实例。另一个很诡异的需要使用可执行的动态内存的子系统是bcachefs这个基于Copy-on-Write（COW）的文件系统，它会为文件系统的每个B-tree node（因为这个文件系统本身就是基于bcache的 btree 实现演化而来）动态生成一个特定的unpack函数，具体的技术细节可以参考下面的链接：

https://lwn.net/ml/linux-kernel/ZFq7JhrhyrMTNfd%2F@moria.home.lan/



曾几何时，Linux内核简单粗暴地使用vmalloc()来分配可执行的内存块，作为安全研究人员，我们当然清楚这种简单的分配方式可能存在极高的安全风险——允许可写的内存页上的数据成为代码，这简直是攻击者最喜欢的肥肉。对这类内存的首要安全防护，就是防止在分配后被随意写入数据。

最近，开发者Mike Rapoport为Linux内核引入了一组新的API，用于替代之前另一个常用的module\_alloc()内存分配接口：

https://lkml.org/lkml/2023/6/1/442

这组API中，主要的内存管理接口包括

void \*jit\_text\_alloc(size\_t len);  
void jit\_free(void \*buf);

这两个接口看上去很简单，其关键之处在于，它只能分配出全零填充的内存块。要对分配的内存块进行写操作，还需要配合下面的API：

void jit\_update\_copy(void \*buf, void \*new\_buf, size\_t len);  
void jit\_update\_set(void \*addr, int c, size\_t len);

与此同时，分配器会根据一个新引入的jit\_address\_space结构体来管理分配的内存的属性，其中一个很重要的属性——pgprot 用来描述分配的内存页必须要对应的保护方式

从安全的角度来看，设计者认为，这组内存分配管理接口和内核的模块加载器是分离的，也就是说，模块加载器并不用负责去分配和释放动态加载的内核模块所需要的空间。但是在随后的相关讨论中，其他的开发者提出了许多意见和建议，比较主要的观点认为分配器应该和待分配的内存数据类型绑定，也就是说，为不同的数据类型开发不同的分配器（这个观点倒是比较的有趣）。当然，完美的设计往往是不存在（或者说不现实）的，大家可以持续关注这些改动，更多的技术细节也可以去lwn上阅读一下原文哦~

原文阅读：https://lwn.net/Articles/933867/