OWASP

Open Web Application Security Project

容器安全技术探索

——Kevin

虚拟机与容器的发展历程

• 虚拟机——物理硬件的虚拟化

- 1960s：IBM公司在1965年发布的IBM7044，实现了最早的虚拟化。它允许用户在一台主机上运行多个操作系统，让用户尽可能充分利用昂贵的大型机资源
- 1990s：VMware公司实现了X86平台的虚拟化，推出了虚拟化软件，使普通用户也能使用虚拟化技术
- 2000s：以XEN和KVM为代表的开源虚拟化技术兴起，以Linux为基础的虚拟化技术应用开始逐步普及
- 2005s: 芯片厂商开始介入虚拟化持术，开始用使用硬件辅助的方式来提升虚拟化的性能(以Intel为代表的VT系列技术)

• 容器——OS环境虚拟化

- 1978s：chroot——UNIX系统中最原始的容器（可能还不能算真正意义上的容器）
- 2007s：namespace,cgroup技术的出现，为Linux容器技术奠定了基础
- 2008s: LXC技术的出现，实现了第一套完整的容器管理方案，并集成到Linux内核中，无需额外的软件包
- 2013s：Docker的出现，开始引领容器技术的高速发展

问题思考：
1. 为什么虚拟化技术会得到发展？
2. 为什么有了成熟的虚拟化技术后，容器技术还会被高度关注以及广泛使用？

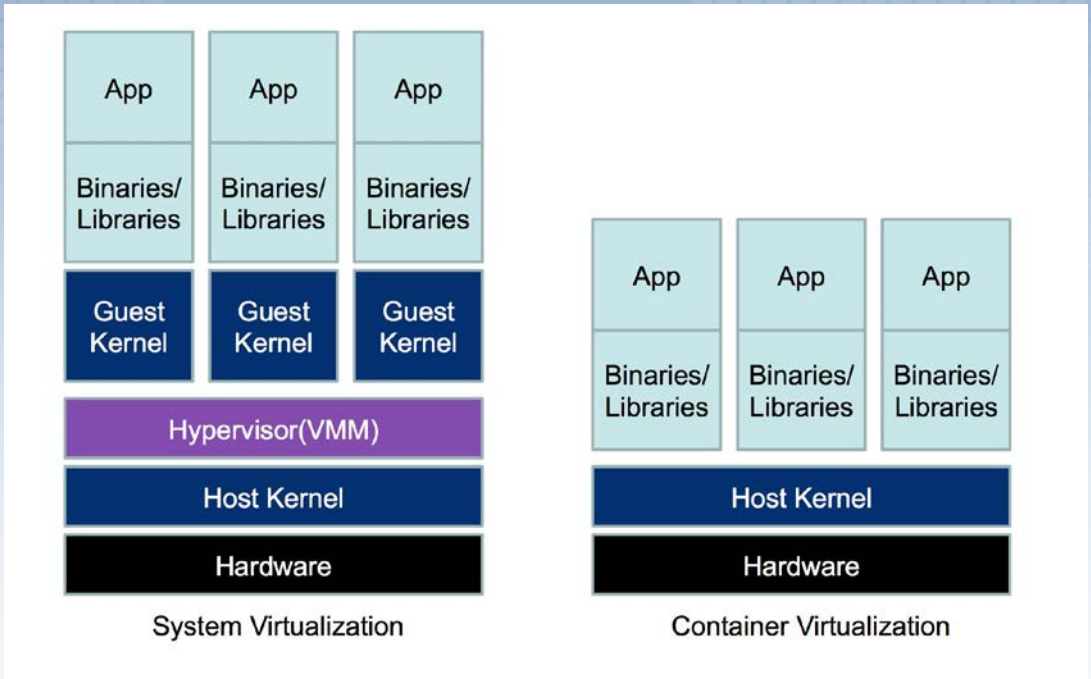
容器面临的安全挑战

- 容器比虚拟机带来了部署上的便利性，但同时容器在应用过程中也面临巨大的安全挑战

容器基础设施安全	软件供应链安全	容器运行时安全
容器的基础设施安全主要强调在容器设施范围内为开发者提供能够保障安全构建容器服务的能务及工具，容器基础设施安全主要由以下四个方面构成： • 身份认证 • 日志审计 • 秘钥的存储和使网 • 网络控制	确保容器镜像得到安全的部署，需要对镜像的生命周期进行全面的考虑，例如：需要确定image没有已知漏洞，部署之前没有被篡改过。避免将有已知漏洞的镜像引入到生产环境中	运行时安全是容器安全保障工作的最重要的环节；因为这涉及到工作负载是否能长期稳定运行。运行时的安全风险主要有： • 容器入侵 • 逃逸攻击 • 配置错误 • ……

- 从技术原理上看，容器的核心威胁来自于共享内核所带来的攻击面，下面主要研究与容器隔离相关的安全技术

虚拟化与容器技术对比



虚拟机特点：

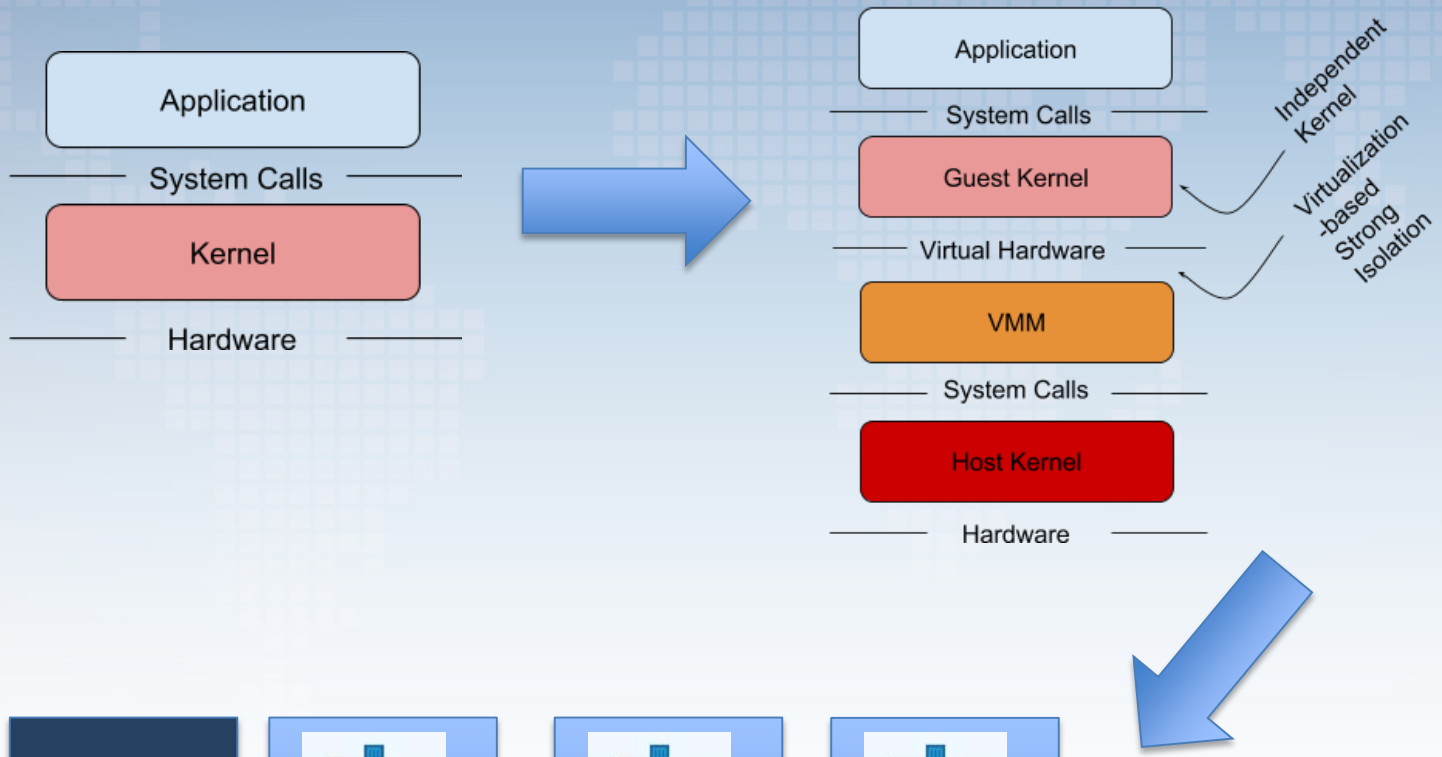
- 内核隔离
- 性能开销较大
- 分钟级启动
- 磁盘占用大（GB）

容器特点：

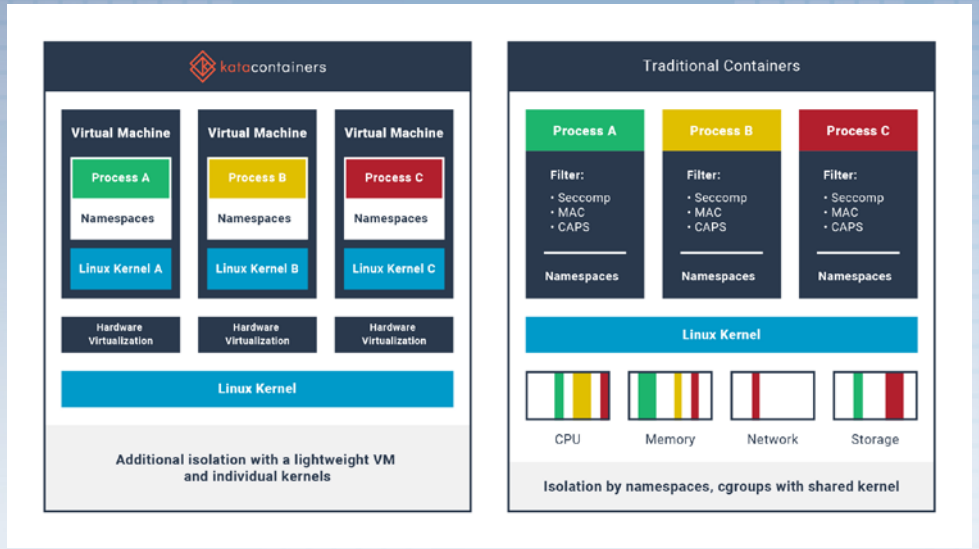
- 内核共享
- 性能开销较小
- 秒级启动
- 磁盘占用小(MB)

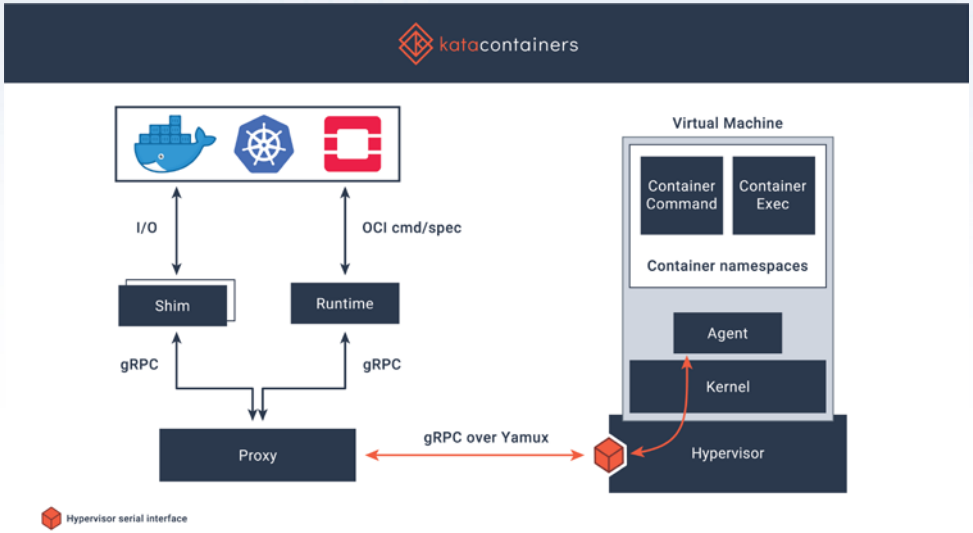
- 虚拟机的隔离强度高，安全性好，但性能开销大； 容器的隔离强度低，安全能弱，但性能好
- 为什么会有在虚拟机内再使用容器？

利用虚拟化技术增强容器安全性



虚拟化容器——Kata





- **Kata架构**

Kata最大的亮点就是通过虚拟化技术完全解决了传统容器共享内核的问题，让每个容器运行在一个轻量级的虚拟机中，使用单独的内核

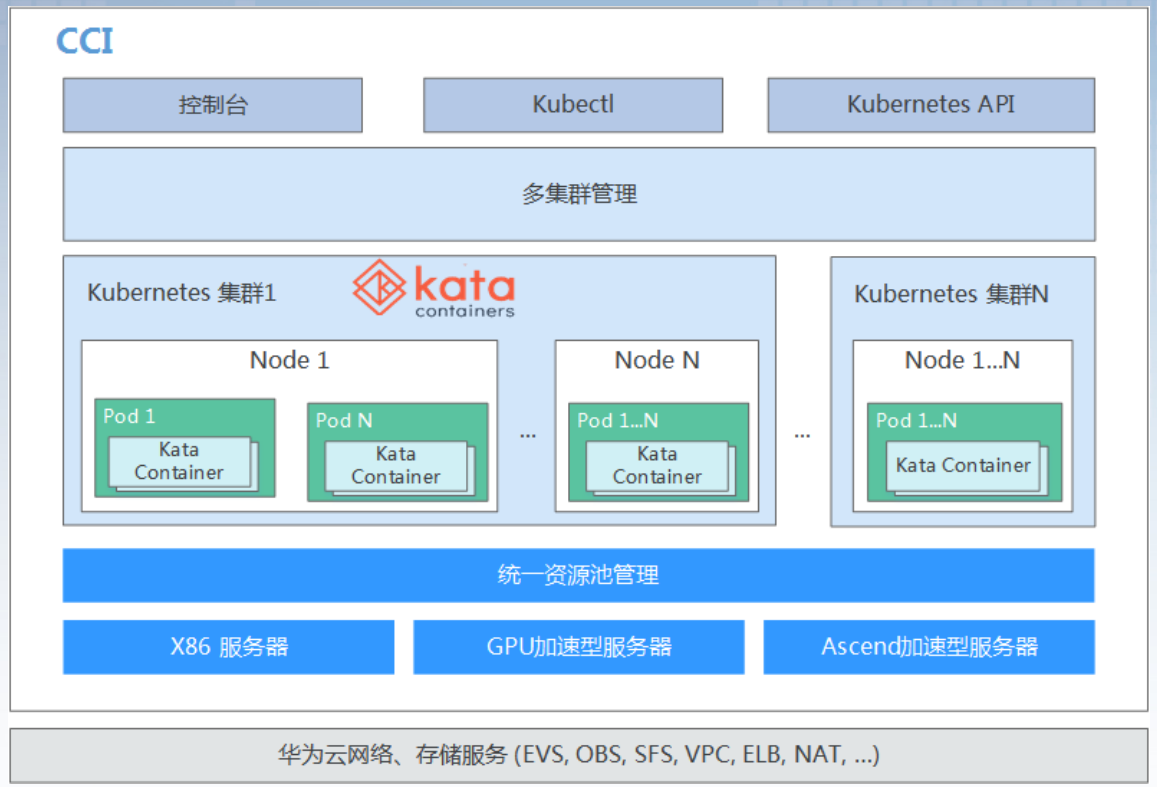
- **Kata的优点**

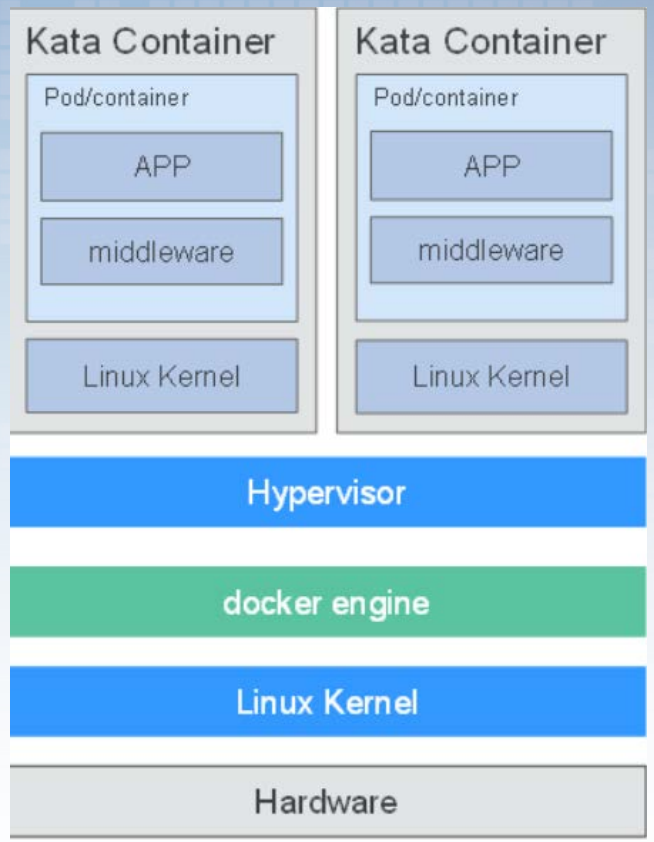
1. 安全性：基于Intel VT技术实现硬件级的安全隔离
2. 兼容器：兼容OCI和CRI接口；同时也兼容不同的架构和硬件平台的不同虚拟化环境
3. 高性能：Kata优化过的内核可以提供与传统容器一样的性能

- **Kata缺点：**

1. 启动速度与传统容器仍有差距
2. 因为已使用了虚拟化技术，不能在虚拟机内运行

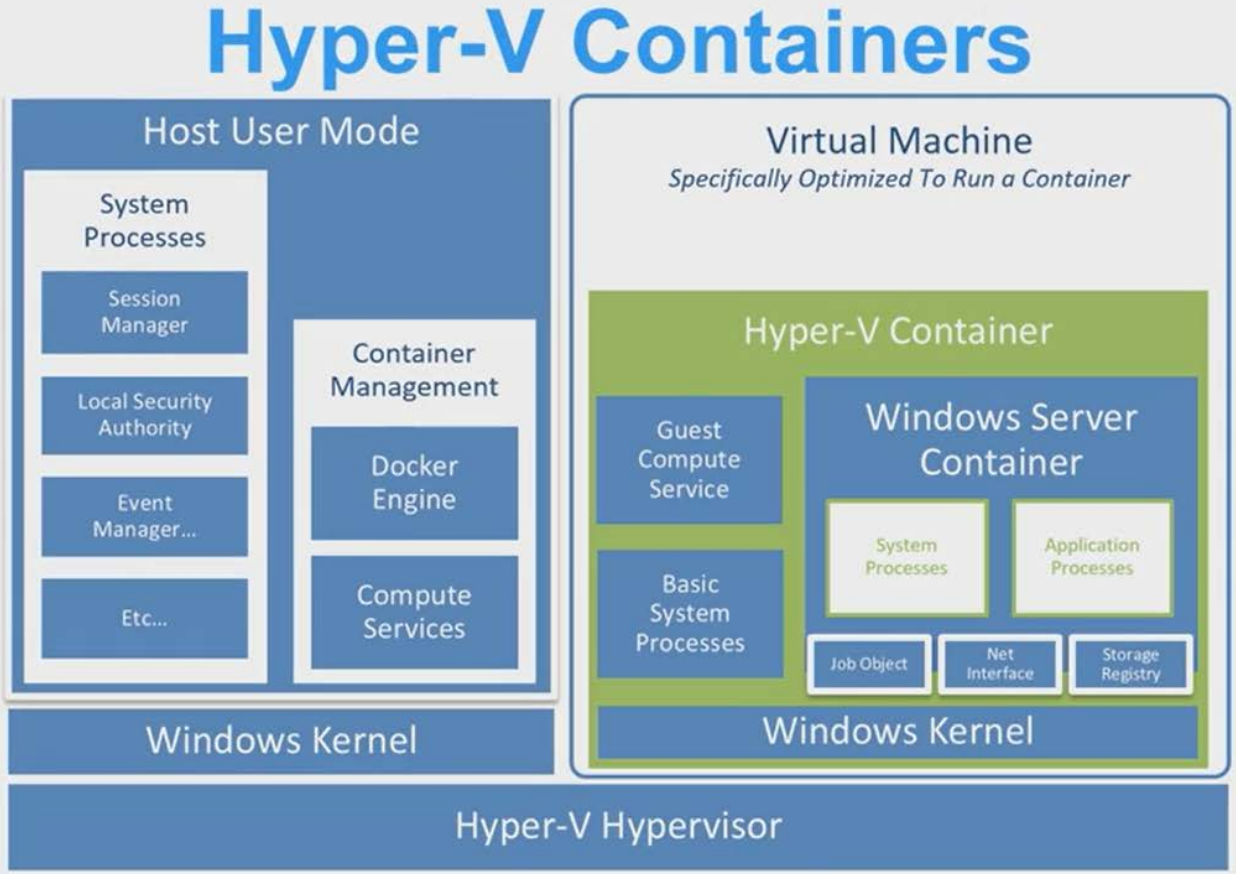
虚拟化容器——华为云OCI服务





- 使用Kata容器提供虚拟机级别的安全隔离，结合自有硬件虚拟化加速技术，提供高性能安全容器。

微软的虚拟化容器架构



•Hyper-V isolation: 多个容器运行在一个物理主机上，但每个容器都在一个单独的虚拟机内运行，提供不同容器之间的内核级隔离

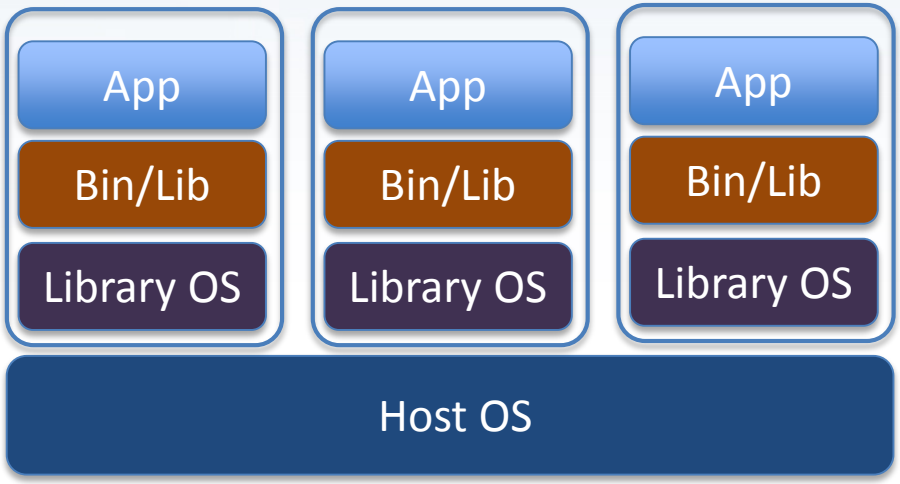
容器安全技术——Library OS

- **什么是Library OS**

A library operating system is one in which the services that a typical operating provides, such as networking, are provided in the form of libraries and composed with the application and configuration code to construct a **unikernel**: a specialized, single address space, machine image that can be deloyed to cloud or embedded environments.

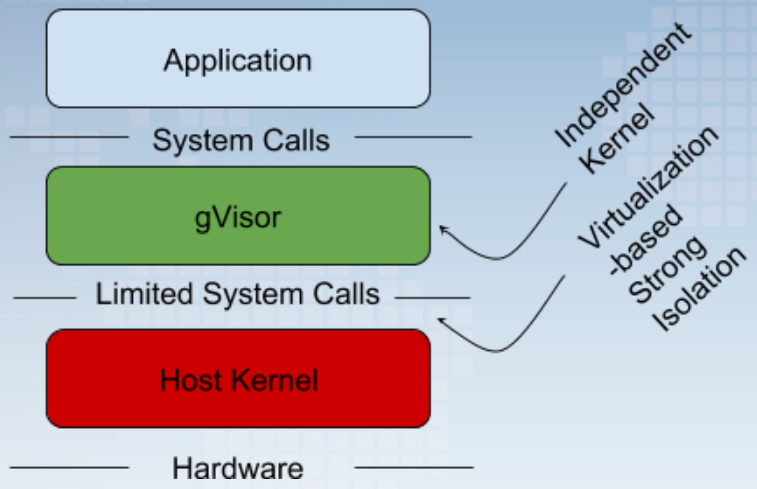
- **Library OS作用**

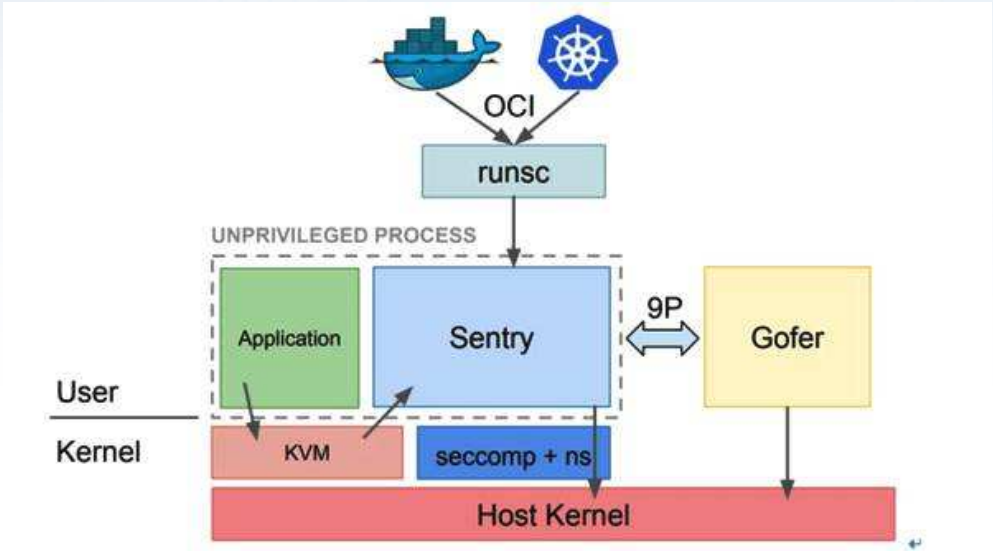
使用和Library OS实现了内核的部分系统调用，减少了内核与应用之间的交互，而每个应用都有一个独立的LibOS,从而在隔离机制上强于传统的容器，同时LibOS实现的系统调用越多，那么应用真正用到的Kernel的系统调用就越少，攻击面的就越少，安全性越好



- **Library OS特点**
  - 作为标准OS的一个库
  - 只需要提供应用所需求的体系调用，非常轻量级
  - 解决应用对Host OS的依赖问题

安全容器技术——Google gVisor





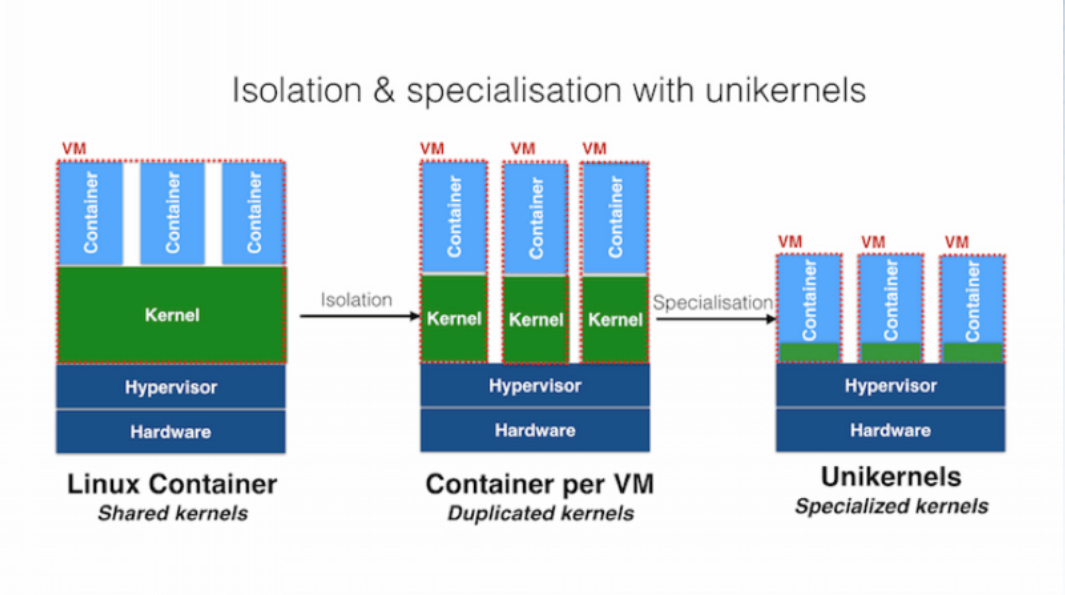
- **gVisor特点**

1. 类似于Library OS,提供了部分系统调用(目前Sentry实现了200多个系统调用)，google开源出来，也是希望通过生态来丰富系统调用
2. 支持KVM和ptrace两种模式
3. 容器内的所有进程，都以Runsc进程的线程存在，形式上Qemu
4. gVisor为每个应用进程都维护了一个Guest页表，Runsc进程自身也有一个Guest页表，从而隔离了各个进程与runsc之间的地址空间
5. gVisor提供两种网络通信方式，一种是通过Host OS的TCP/IP协议栈，二是通过gVisor实现的用户态TCP/IP协议栈
6. 兼容OCI

安全容器技术——unikernel

- **什么是unikernel?**

Unikernels are specialised, single-address-space machine images constructed by using library operating systems.



- 与虚拟化的区别： 直接运行在Hypervisor上，不依赖于其他的OS
- 相关项目:
  - ClickOS
  - Drawbridge
  - IncludeOS
  - ……

- **Improved security**

Unikernels reduce the amount of code deployed, which reduces the attack surface, improving security.

- **Small footprints**

Unikernel images are often orders of magnitude smaller than traditional OS deployments.

- **Highly optimised**

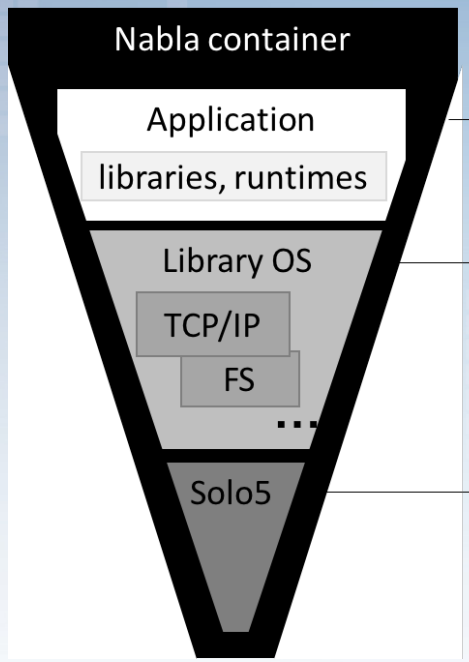
The unikernel compilation model enables whole-system optimisation across device drivers and application logic.

- **Fast Boot**

Unikernels can boot extremely quickly, with boot times measured in milliseconds.

http://unikernel.org/projects/

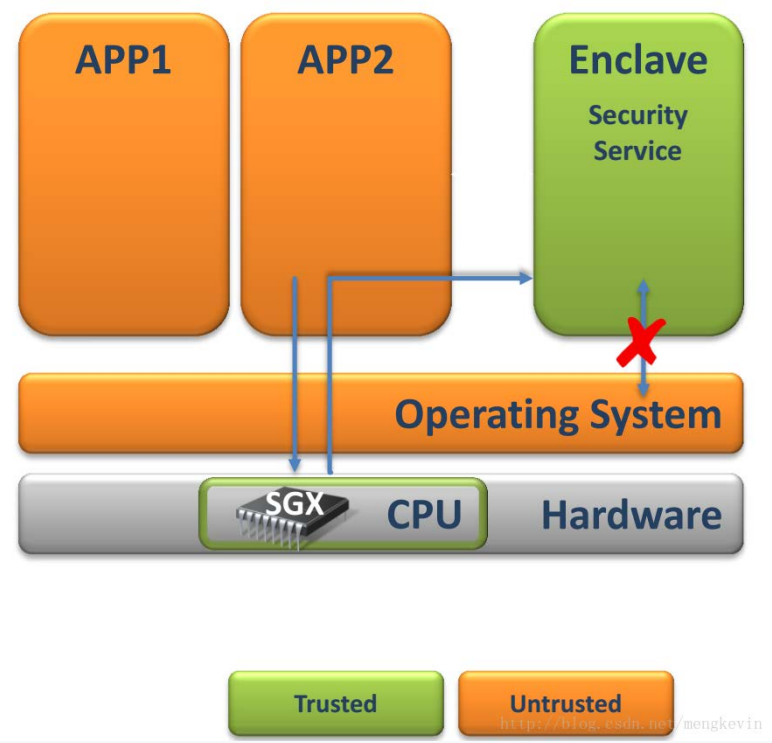
容器安全技术——IBM Nabla

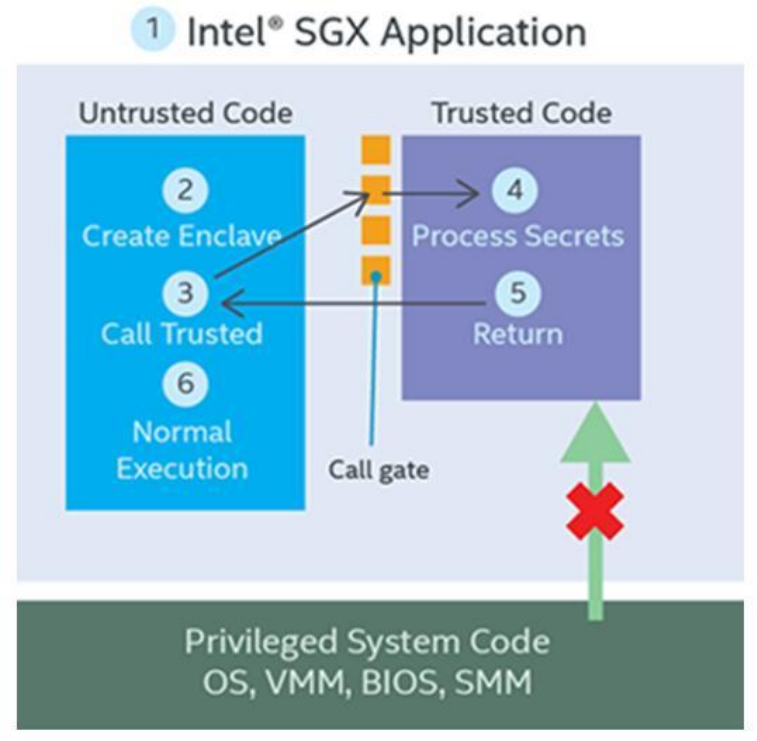


Solo5 originally started as a project by Dan Williams at IBM Research to port MirageOS to run on the Linux/KVM hypervisor. Since then, it has grown into a more general sandboxed execution environment, suitable for running applications built using various unikernels (a.k.a. library operating systems), targeting different sandboxing technologies on diverse host operating systems and hypervisors.

A containerized application can avoid making a Linux system call if it links to a library OS component that implements the system call functionality. Nabla containers use library OS (aka unikernel) techniques, specifically those from the Solo5 project, to avoid system calls and thereby reduce the attack surface. Nabla containers only use 7 system calls; all others are blocked via a Linux **seccomp** policy.

芯片级安全隔离技术——SGX



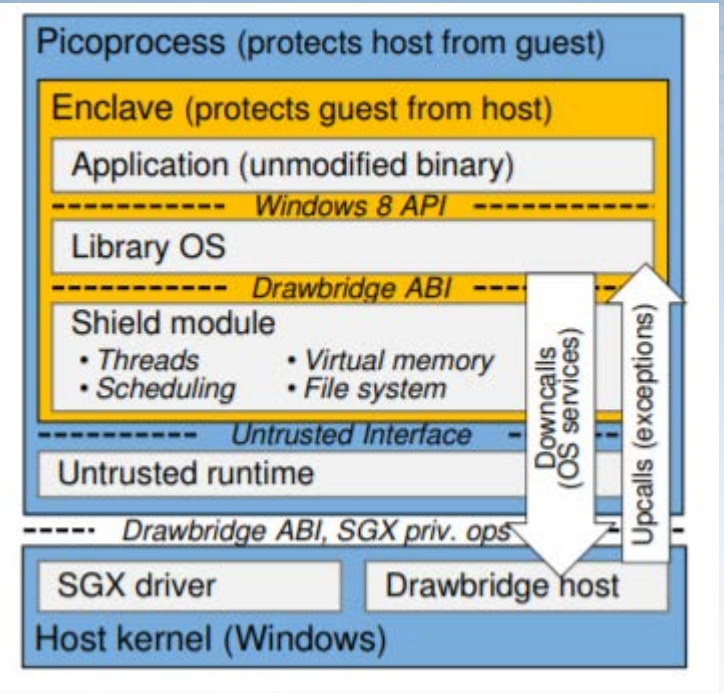


- SGX是一种CPU指令，使应用程序能创建安全区域； 安全区域的地址空间受保护，即使在存在特权恶意软件的环境下
- 应用程序包含两部分：Untrusted Code与Trusted Code,两部分都有自己的代码和数据
- SGX对Enclave区域提供完整性与机密性的保护

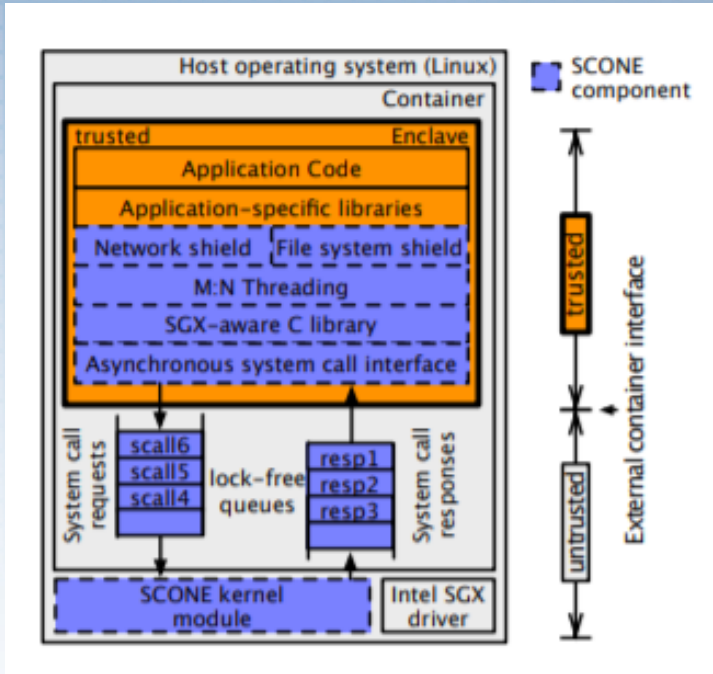
•缺点：
- •对已有的应用改造较大（Intel提供的SDK能力较弱）
- •SGX当前支持的内存比较小(128M)
- •CPU模式切换时开销比较大

OWASP
Open Web Application Security Project

基于SGX的安全容器



Haven components and interfaces



SCONE architecture

- 由于SGX技术本身的限制，基于SGX的容器基本还处于研究阶段

https://www.usenix.org/system/files/conference/osdi14/osdi14-paper-baumann.pdf
https://www.usenix.org/system/files/conference/osdi16/osdi16-arnautov.pdf

OWASP

Open Web Application Security Project

Thank You

Kevin
微信： Kevin501112