容器安全解决方案探讨与研究

胡俊，李漫
（武汉青藤时代网络科技有限公司，湖北武汉 430000）

摘 要：近年来，微服务架构的兴起，容器化部署已经成为时下最流行的生产方式，越来越多的公司将应用部署在基于容器的架构上。然而，在容器技术被广泛接受和使用的同时，容器以及容器运行环境的安全成为了亟待研究和解决的问题。为了进一步了解容器以及容器运行环境的安全威胁，论文将对容器安全问题进行分析总结，并结合容器安全的实际解决方法进行探讨，为使用容器的用户提供安全防护意见。

关键词：容器安全；解决方案；探讨
中图分类号：520.6099 **文献标识码：**B

Discussion and research on container security solutions

Jun Hu, Man Li
(*WuHan QingTeng Time Network Technology Co.,Ltd,HubeiWuhan 430000*)

Abstract: In recent years, the emergence of micro-service architecture, containerized deployment has become the most popular production method nowadays, more and more companies deploy applications on a container-based architecture. However, while container technology is widely accepted and used, the safety of containers and container operating environments has become an issue that needs to be studied and solved. In order to further understand the security threats of containers and container operating environment, this paper will analyze and summarize the container security issues, and discuss the practical solutions of container security in the market to provide security protection opinions for users who use containers.

Key words: container security; solution; explore

1 引言

近年来，云计算的模式逐渐被业界认可和接受。在国内，包括政府、金融、运营商、能源等众多行业以及中小企业，均将其业务进行了不同程度的云化。但简单地将主机、平台或应用转为虚拟化形态，并不能解决传统应用的升级缓慢、架构臃肿、无法快速迭代等问题，于是云原生(Cloud Native)的概念应运而生。云原生提倡应用的敏捷、可靠、高弹性、易扩展以及持续的更新。在云原生应用和服务平台构建过程中，近年兴起的容器技术凭借其弹性敏捷的特性和活跃强大的社区支持，成为了云原生等应用场景下的重要支撑技术。

容器是一种轻量级、可移植、自包含的软件打包技术，使应用程序可以在几乎任何地方以相同的方式运行。开发人员在自己笔记本上创建并测试好的容器，无需任何修改就能够在生产系统的虚拟机、物理服务器或公有云主机上运行。容器技术在操作系统层面实现了对计算

机系统资源的虚拟化，在操作系统中，通过对CPU、内存和文件系统等资源的隔离、划分和控制，实现进程之间透明的资源使用。

虽然容器技术备受追捧，在多个领域得到广泛使用，但其背后的安全问题不容忽视。Gartner 在 2017年6月发布了《2017年度最新最酷的信息安全技术》，其中就包括容器安全(Container Security)以及 DevOps 安全—— DevSecOps(Development -Security-Operations)。在 Gartner 发布的《2017年云安全技术成熟度曲线》 (Hype Cycle)中，容器安全目前处于新兴(Innovation Trigger)阶段，虽然尚未成熟，但技术进步较快。容器自身安全，以及容器生态环境中的安全防护，都是准备部署容器的企业中 CIO 和 CISO 们非常关心的安全问题。

因此，容器以及容器运行环境的安全问题亟待调研和分析，了解容器技术和基于容器技术的系统存在的风险、应对策略，将是构建安全的云原生环境非常重要的前提。

2 容器安全风险与挑战

作为容器技术的一种具体实现，Docker 近年来受到越来越多的关注，在某种程度上，Docker 已经成为了容器技术的代表。本文将从Docker着手，分析容器安全遇到的风险与挑战。

2.1 不安全的镜像源

开发者通常会在Docker 官方的Docker Hub仓库下载镜像，这些镜像一部分来源于开发镜像内相应软件的官方组织，还有大量镜像来自第三方组织甚至个人。在从这些镜像仓库中获取镜像的同时，也带来了潜在的安全风险。例如，下载镜像内软件本身是否就包含漏洞，下载的镜像是否被恶意的植入后门，镜像在传输过程中是否被篡改。

1）镜像使用有漏洞的软件

2017年4月，Federacy 的一名研究人员发布了一项报告，该报告分析了公开仓库中Docker 镜像的漏洞。这项研究扫描了 133 个公开 Docker 仓库中的 91 个。在扫描到的所有镜像中，其中的 24% 存在较为明显的漏洞，在这些漏洞中，11% 的风险等级为高，13% 的风险为中等，剩余的被视为潜在漏洞。扫描的 Linux 分发版本包括 Ubuntu、Debian 和 RHEL。基于 Ubuntu 的镜像在严重漏洞总数上的占比最高（27%），而 Debian 则是漏洞最少的分发版

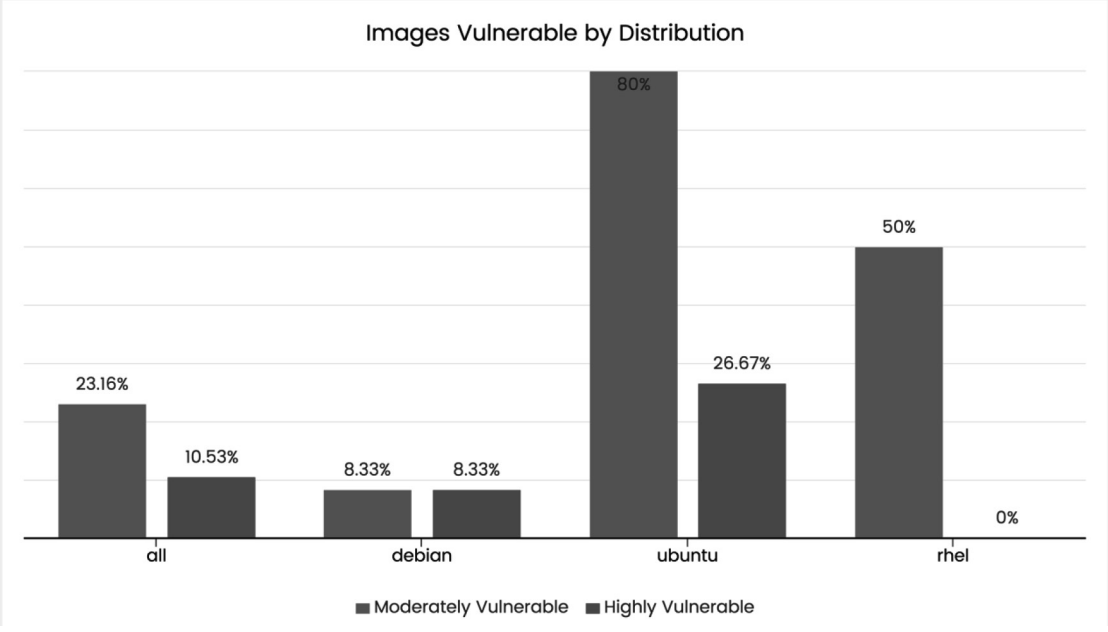

图1 公开仓库中Docker镜像的漏洞分布情况

（8%）。

2）攻击者上传恶意镜像

如果黑客在制作镜像时植入木马、后门等恶意软件，并将恶意镜像上传至 Docker Hub 等公共仓库，那么用户的容器环境从一开始就已经不安全了，后续更没有什么安全可言。

比如 Docker 对镜像不安全的处理管道，Docker 支持三种压缩算法，分别是 gzip、bzip2、xz。gzip 和 bzip2 使用 Go 的标准库，所以相对安全;xz 由于没有使用原生的 Go 去实现，又由于其使用由 C 编写的 XZ Utils 开源项目， 所以存在 C 程序恶意写入的可能，一旦被写入会导致执行任意代码漏洞，而只要有一个漏洞，执行Docker pull拉取镜像时就有可能导致整个系统沦陷。

2.2 容器入侵事件

由Docker本身的架构与机制可能产生的问题，这一攻击场景主要产生在黑客已经控制了宿主机上的一些容器（或者通过在公有云上建立容器的方式获得这个条件），然后对宿主机或其他容器发起攻击来产生影响。

1）容器逃逸攻击

容器与宿主机共享内核，容器可以执行一些系统调用。 如果宿主机的系统调用存在漏洞，容器便可以利用这些系统调用进行提权，进行虚拟机逃逸。

例如，脏牛漏洞（CVE-2016-5195）是linux内核的一个提权漏洞，该漏洞是Linux内核的内存子系统在处理写时拷贝（Copy-on-Write)时存在条件竞争漏洞，导致可以破坏私有只读内存映射。黑客可以获取低权限的本地用户后，利用此漏洞获取其他只读内存映射的写权限，进一步获取root权限。因此，攻击者入侵到容器中后，可以利用这个有漏洞的系统调用，获取宿主机的root权限。

2）容器拒绝服务攻击

容器实例共享宿主机的资源，也就是说底层的CPU、内存和磁盘等资源由主机操作系统进行统一的调度和分配。如果不对每个容器的可用资源进行有效的限制和管理，就会造成容器之间资源使用不均衡，严重时可能导致主机和集群资源耗尽，造成拒绝服务。

DOS攻击可针对任何资源，例如计算资源、存储资源、网络资源等。以存储资源为例，在容器技术的实现中，通过mount命名空间实现了文件系统的隔离，但是在存储空间方面却没有任何限制。因此，一个容器如果不断写文件，将会写满存储介质，其它容器将无法执行写操作，导致拒绝服务攻击。

3）容器网络攻击

目前Docker总共提供三种不同的网络驱动，分别是桥接网络（Bridge）、MacVlan,覆盖网络（Overlay network）、三种网络驱动都存在着安全风险。例如，桥接网络，是Docker预设的网络驱动，一个docker0的网桥将所有容器连接该网桥，docker0网桥扮演着路由和NAT的角色，容器间通信都会经过容器主机。如果各容器之间没有防火墙保护，攻击者就可以利用主机内部网络进行容器间相互攻击。

2.3 运行环境的安全

除了Docker 本身存在的问题以外，Docker的运行环境存在的问题同样会给Docker的使用带来风险。

1）Docker应用本身存在的安全问题

CVE官方记录Docker历史版本共有18项漏洞。主要有代码执行、权限提升、信息泄露。绕过这几类，高危漏洞1个，中危漏洞6个。基本出现在1.6.1及以前的版本中，目前Docker最新版本为1.10.2暂未被发现cve漏洞。因此Docker用户最好将Docker升级为最新版本，CVE详情如表1所示。

2）宿主机的安全问题

容器运行在宿主机上，若宿主机存在问题，同样会对容器的安全造成影响。例如，因为Docker与主机共用一个操作系统内核，因此如果主机内核存在横向越权或者提权漏洞，尽管容器是以非特权模式运行的，但容器中的攻击者还是可以利用内核漏洞逃逸到主机，进行恶意操作。同时，宿主机应遵循最小化安装原则，不安装不必要的软件以减少可被攻击面。

表1 Docker应用本身的漏洞

CVE编号	漏洞版本	漏洞名称
CVE-2018-15514	18.05.0 Win66	Docker for Windows 安全提权漏洞
CVE-2017-14992	17.09.0 CE	Docker输入验证漏洞
CVE-2016-8867	1.12.2	Dcoekr Engine 权限许可和访问控制
CVE-2015-3630	1.6.0	Docker Libcontainer 安全绕过漏洞
CVE-2015-3627	1.6.1	Libcontainer和Docker Engine 权限许可和访问控制漏洞
CVE-2015-3630	1.6.1	Docker Engine 安全绕过漏洞
CVE-2014-9358	1.3.3	Docker 目录遍历漏洞
CVE-2014-9357	1.3.2	Docker 权限许可和访问控制漏洞

3）容器配置不当引起的安全问题

Docker本身的配置，以及容器在运行时的配置不当，也会引发安全问题。例如若以root权限启动容器，则一旦攻击者入侵到容器中，即拥有了主机内核的所有功能，容器几乎可以做主机可以做的一切。因此，容器运行时需为容器配置运行用户。

3 容器安全防护

容器的安全防护应该从开始覆盖到容器的整个生命周期中，即容器的构建、分发、运行三个阶段，这样才能确保持续的安全性。如图2所示。

3.1 构建阶段

容器持续性安全的第一步，是构建安全的应用。这要求开发工程师具备一定的安全知识，从代码源头上减少可被攻击的风险。

进行代码集成和测试之前，利用代码审计工具发现代码中潜在的漏洞。

在构建镜像时，首先应确保基础镜像是从头开始编写的，或者是通过安全仓库下载的一个可信的基础镜像。然后，应通过去掉不必要的库和安装包，对镜像进行精简、加固，以减少可被攻击面。

镜像构建完成后，在镜像正式投入使用之前，应对镜像进行漏洞扫描以及时发现潜在的风险；同时，应该对Registry中的镜像进行周期性扫描，这些镜像中有可能存在近期新爆发的漏洞。

3.2 镜像分发阶段

在镜像传输阶段，进行适当的访问控制和镜像校验很有必要。

为保证镜像内容的可信，建议开启 Docker 的内容信任机制。Docker在1.8.0版本中增加了Content Trust功能，大大提高了Docker的安全性。内容信任机制为向远程镜像仓库发送和接收的数据提供了数字签名的功能，这些签名允许验证镜像标签的完整性和镜像发布者。

对Registry、编排工具等其他开发工具应设置统一的访问控制策略，集成到类似LDAP这类的平台中。

3.3 运行阶段

生产环境中运行容器的安全性分为运行时准备阶段和生产环境阶段两个步骤。

1）运行时准备阶段

当容器正式运行在生产环境中之前，应确保容器的运行时环境是安全的。这包括容器运行时的配置、宿主机的安全、Docker应用本身的安全配置、负载均衡等等其他网络或系统服务。

若容器运行时配置不当，会严重影响容器的安全运行。例如，容器运行时，需配置容器的运行用户，若不配置容器的运行用户，容器将会以ROOT权限运行。黑客一旦入侵到以ROOT权限运行的容器中，则拥有了主机内核的所有功能，黑客几乎可以做主机可以做的一切。因

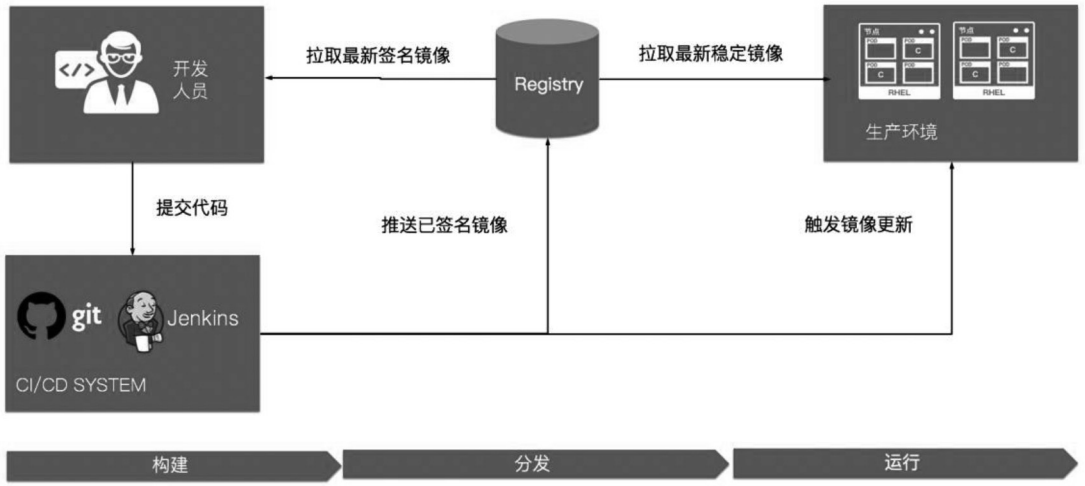

图2 容器环境安全工作流程

此，需使用容器编排工具，对容器的访问策略、运行时配置等进行统一管理，并实现基于CIS Benchmark的最佳操作实践配置，确保容器运行时配置得当。

容器与宿主机共享操作系统内核，若宿主机存在安全问题，则同样也会影响容器的安全运行。因此应定期检查宿主机的内核情况，确保内核的安全性；同时，宿主机应遵循最小化安装原则，不安装不必要的软件，以减少可被攻击面。

同时，Docker作为容器的运行平台，若配置不当将在一定程度上影响容器的安全运行。例如：Docker Daemon 中应配置限制默认网桥中的网络流量。因为默认情况下，默认网桥上同一主机上的容器之间允许所有的网络通信。因此，每个容器都有可能读取同一主机上容器网络上的所有数据包，这可能会导致不必要的信息泄露给其他容器。因此需要限制默认网桥上的容器的通信，只加入需要通信的容器；或者创建自定义网络，只加入需要与该自定义网络通信的容器。因此，应对宿主机上的Docker应用的配置采用基于CIS Benchmark的最佳操作实践，以减少容器的脆弱面。

2）生产环境阶段

容器的引入带来了新的入侵方式，同时使东西向流量的安全问题更加突出，因此，对生产环境进行持续性的安全防护和检测必不可少。

网络安全：容器的使用带来更频繁的东西向流量，而传统的安全产品对容器内的活动视而不见。因此容器安全产品需能可视化容器之间的访问关系，并监控容器之前的异常访问事件，对于异常、恶意的访问连接进行告警，同时监控基于网络的攻击事件如DDOS攻击、DNS攻击等。

入侵检测：传统安全产品对容器内的活动无法知晓，但黑客极有可能通过入侵容器的方式发起攻击。因此对容器内的入侵事件进行检测必不可少。例如检测容器内的隐藏的WebShell、监控容器内的恶意进程、提权行为等等。

运行时容器的漏洞扫描：虽然在容器使用镜像运行之前，会对镜像进行一次全方位的漏洞扫描。但一方面，容器运行起来后，可能会被黑客安装上有漏洞的应用加以利用；同时随着时间的推移，软件应用中更多的漏洞被发现了，这些有可能在正在运行的容器中被使用。因此需定期扫描运行中的容器，以确保运行态的容器中不存在新的漏洞。

3.3 容器的安全防护解决方案

综上所述，容器的安全解决方案按照容器的生命周期来分，分为以下几个方面。如图3所示。

4 容器安全产品防护能力

基于以上所述，容器的安全防护产品应具备以下能力。如图4所示。

1）主机、容器、编排工具等基本资产信息的获取能力

应具备宿主机、Docker、编排工具如k8s、容器的基本信息的获取能力。基本信息的获取，一方面有助于帮助企业消除资产盲点、梳理资产的基本情况，使容器资产清晰可视化；另一方面，有助于弥合运维团队与安全团队之前的差距，使安全人员也能第一时间获取资产变动的信息。同时，获取资产运行、配置情况，可帮助企业及时发现企业因配置不当引发的安全问题。

2）镜像的漏洞扫描能力

应具备能对处于各个阶段的镜像进行漏洞扫描的能力。在CI/CD阶段，镜像构建完成后即对镜像进行一次漏洞扫描，能从源头上降低镜像的风险；对Registry中的镜像进行周期性漏洞扫描，即在镜像投入使用前对镜像进行扫描，能及时发现新产生的漏洞对镜像的影响；在运行环境中对镜像进行周期性漏洞扫描，能再次验证使用中的镜像的安全性。因此，具备镜像的漏洞扫描能力必不可少。

3）容器内入侵事件监控的能力

传统的安全产品对容器内的活动无法感知，因此容器安全产品需具备能监控容器内入侵事件的能力。例如，对容器内进程的活动进行监控，及时发现挖矿等恶意进程；对容器内的反弹、提权行为进行监控，及时发现黑客入侵的痕迹；若为Web容器，则需对web目录下的文件进行实时监控，以及时发现黑客上传的WebShell等等。

4）容器网络安全防护的能力

容器的使用，使得东西向流量的交互变得更为频繁，东西向流量带来的安全问题也日益突出，而传统的安全产品对东西向流量的安全问题视而不见，因此容器安全产品应具备能对东西向流量的监控能力，以增加东西向的网络安全防护。一方面，需能识别、监控容器间的访问关系，并及时发现、阻断容器的恶意访问；另一方面，需对容器的网络访问进行系统调用、流量的监控，并及时发现异常的系统调用和由于黑客攻击引发的异常流量情况，例如DDOS攻击、DNS攻击等等。

5）容器事件响应 & 处理的能力

安全产品需要能发现容器的安全问题，同时，还应具备对容器事件的影响和处理的能力。一方面，对于存在高危漏洞的镜像、存在不安全配置的容器，应能对这些存在风险的容器进行预处理，即阻止这类镜像被使用或阻断这类存在潜在威胁的容器的运行；另一方面，对于发现容器入侵事件、网络安全问题的容器，应能及时对受感染容器进行隔离或停止运行，避免入侵事件的进一步泛滥。

图3 容器安全防护解决方案

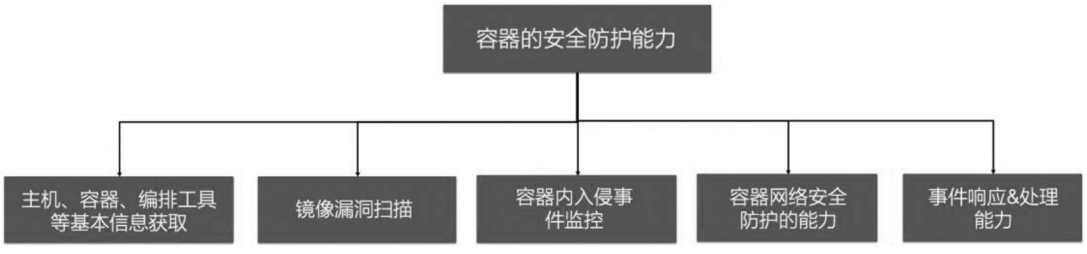

图4 容器安全产品防护能力

5 容器安全产品技术方案对比

结合容器安全的产品能力，总结出容器安全产品的核心技术应具备以下几点。

1）获取容器及主机上Docker的基本信息的能力

一方面，通过查找主机上Docker的配置文件获取Docker的配置信息；另一方面，通过Docker提供的API获取宿主机上的容器、镜像的基本信息。通过获取基本的资产信息，即可向安全人员清晰展示系统中跑了哪些容器。

2）监控容器内的进程、文件、系统调用行为的能力

对容器内的用户行为进行实时监控，以检测容器内的可能存在的黑客行为。例如对恶意进程的监控，可发现容器内的反弹行为、挖矿行为等；对容器内的相关文件进行监控，可对容器内文件完整性进行监控，同时也可对异常的后门文件进行识别，例如黑客上传的web后门、系统后门等；对容器内的系统调用进行检测，即可发现黑客利用内核漏洞或配置问题进行的恶意调用。

3）Registry中镜像信息的获取能力

通过获取Registry中镜像的信息实现对镜像的漏洞扫描能力。即根据Registry的API,获取Registry中镜像的基本信息、拉取镜像层，对镜像中的软件应用进行漏洞扫描，以发现其中存在的潜在风险。

目前，市场上涌现了一批容器安全产品安全厂商，如Neuvector、Twistlock、StackRox、Aqua等等。容器安全产品的技术方案上，一种是使用了平行容器的方式对宿主机上的容器进行安全防护，另一种则是采取了基于宿主机Agent的方式。

平行容器技术方案：利用容器的隔离性和良好的资源控制能力，在容器的宿主机中启动一个容器，该容器通过挂载宿主机的所有文件系统，而后在容器内部对这些文件系统进行实时监控和处理响应，以实现对容器进行防护的作用。

基于宿主机Agent的技术方案：即基于青藤Agent的主机防护能力，监控宿主机上容器相关的文件、进程、系统调用等等信息，增加其Agent中对于容器的清点、监控、防护能力，以实现一个Agent，实现宿主机安全、容器安全两种防护的效果。

针对以上两种技术方案，下文将从以下几个方面进行比较。

1）容器安全防护能力

从容器安全的防护产品能力上来看，如：镜像扫描、容器基线检查、容器内入侵事件的监控等，平行容器的技术方案和宿主机中安装Agent的方案均能实现这几个功能。但从上图中技术方案构图中可以看出，平行容器的方案，是在容器层，对其他容器进行检测。而宿主机安装Agent的方案，则是与Docker Engine同处于操作系统的应用层。相比平行容器的方案，Agent可以基于操作系统层面，对宿主机以及容器的底层调用进行更基础的信息收集和事件监控，同时也能对Docker Engine和上层的容器应用进行识别和监控。整体来看，宿主机安装Agent的技术方案具备的容器安全防护技术能力更强一些。

2）宿主机防护能力

宿主机的防护能力，是指对容器的运行环境进行防护，包括但不限于对宿主机上安装的资产进行识别，对宿主机中的软件进行漏洞扫描，监控宿主机中可能发生的反弹、提权、端口扫描等入侵行为。而因为平行容器方案是对容器层进行识别和监控，因此并不具备宿主机的防护能力；宿主机中安装Agent的方案，则本身即可对宿主机的操作系统、宿主机中的事件进行监控，因此天然具备宿主机的防护能力。

3）安装部署

从安装部署的便利性上来看，基于平行容器的技术方案，利用了镜像的可移植性特点，可以很好的利用企业管理的Registry、编排工具等集成方案，对平行容器进行平滑升级。而安装Agent的方案，则安装部署上，在需完全由厂商自行实现对应的安装部署管理，开发成本较大，但也能实现同镜像一样，统一下发Agent更新的能力。

4）稳定性

从产品的稳定性上来看，平行容器的性能控

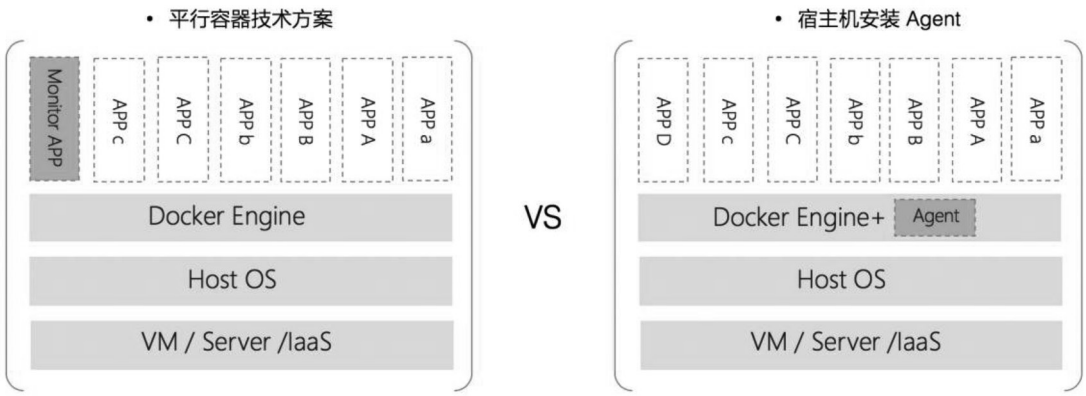

图5 容器安全产品技术方案架构

制，采用的是容器自带linux的Cgroup机制，能很好的限制住容器使用的CPU、内核等性能，以保持产品的稳定性。而宿主机安装Agent的方案，则需要厂商自行研究保持稳定性的技术手段，例如，可通过监控宿主机的性能占用情况，对超过一定性能指标的情况，自行对Agent进行降级处理，以释放更多的资源给业务需求。

5）执行效率&性能

经过调研实验发现，基于平行容器的解决方案，对于主机以及Registry中的镜像的扫描执行效率比较低，而基于Agent的扫描机制，则能大大提升扫描速度，降低性能消耗。

在对于宿主机中的本地镜像扫描的场景中，平行容器的方案是从Docker引擎中拉取镜像层到内存中，再进行解压和漏洞扫描，需处理的文件比较多，该过程中，主要是拷贝和解压的过程比较耗时。而Agent的扫描方案，则是可直接访问镜像对应目录，直接解析本地文件并进行扫描，减少了拷贝文件和解压的过程，能将扫描速度提升50倍以上。

在对Registry中的镜像扫描的场景中，平行容器的方案，需要首先从Registry中拉取所有镜像层的信息，然后在本地对所有层进行解压的扫描。而基于Agent的扫描机制，则根据镜像层，逐层下载镜像层，不对重复的镜像层进行下载和解压，扫描镜像层的结果同时会进行缓存，供下次漏洞扫描时使用，可避免重复扫描。经统计，Agent的扫描方式能提高2-3倍的扫描速度，同时下载流量约能节省50%左右。

6）开发效率

在开发效率上，基于平行容器的技术方案目前存在开源方案比较多，相关技术社区也支持的比较多，比如镜像扫描的开源工具clair,进行容器监控的sysdig等，因此相应的产品迭代会比较快。而Agent的方案，则暂无开源方案的支持，需靠厂商投入大量人力进行自研，迭代周期会相对较长。

5 结束语

容器技术提供了一种轻量的虚拟化方式，在DevOps、微服务等云原生应用的开发过程中，提供了极大的便利，经过近几年的发展，显示了强大的优势。容器最初作为开发者工具而流行，可以使用它来做隔离的开发测试环境和持续集成环境，这些都源于容器轻量、易于配置和使 用的优势。

与此同时，云原生应用(Cloud Native Application)因其敏捷、易扩展、高可用等优势，被大家广泛接受。尤其是由谷歌组织成立的云原生计算基金会(CNCF)，更是得到了众多厂 商与开源社区的关注。从简单的应用容器化，到云原生应用的开发，容器技术成为了其最基础也是最核心的支撑技术。新技术带来便捷与利益的同时，其安全性也需要引起足够的重视。

近年来，由于容器以及容器应用环境引发的安全风险与安全事件不断的被曝出，容器网络、 容器镜像、暴露的 API、容器的隔离等问题成为了容器使用时需要着重考虑的问题。

本文从容器安全风险入手，分别从软件脆弱性、安全威胁、应用安全威胁等方面，系统的介

绍了容器以及容器应用中所面临的安全问题。针对这些安全问题，从主机安全、镜像安全、 网络安全、应用安全等多个角度，提出了相应的检测与防护建议，并针对市场上出现的安全产品技术方案进行了对比和研究，期望能对企业未来在容器安全产品的选择上有所帮助。

参考文献

[1] Docker, https://github.com/docker.

[2] 国家信息安全漏洞库，http://www.cnnvd.org.cn/web/vulnerability/queryLds.tag?pageno=3&repairLd= [9] https://docs.docker.com/enterprise/17.06/#17062-ee-6-2017-11-27.

[3] 《Docker 容器与容器云 第 2 版》.

[4] Docker Hub, https://hub.docker.com/.

[5] Docker images commandline reference, https://docs.docker.com/engine/reference/commandline/images/.

[6] Kubernetes, https://kubernetes.io/.

[7] Azure Kubernetes Service https://azure.microsoft.com/en-us/services/container-service/.

[8] Docker Documentation, https://docs.docker.com/.

[9] 国家信息安全漏洞库，http://www.cnnvd.org.cn/index.html.

[10] CIS Benchmarks, https://www.cisecurity.org/cis-benchmarks/.

[11] Docker Registry, https://docs.docker.com/registry/.

[12] Dirty Cow, https://dirtycow.ninja/ .

[13] Aqua Security, kube-bench, https://github.com/aquasecurity/kube-bench.

[14] NeuVector, https://neuvector.com/.

[15] Twistlock, https://www.twistlock.com.

[16] StackRox, https://www.stackrox.com.

作者简介：

胡俊（1984-），男，汉族，湖北武汉人，华中科技大学,软件工程硕士，武汉青藤时代网络科技有限公司联合创始人兼产品负责人；主要研究方向和关注领域：主机安全、HIDS、容器安全。

李漫（1991-），女，汉族，湖北武汉人，武汉理工大学,电子商务学士，武汉青藤时代网络科技有限公司容器安全产品负责人；主要研究方向和关注领域：主机风险、容器安全。