**文献综述**

学院：计算机学院 班级：硕1708 学号：M201773154 姓名：汪润泽

**摘要：**近年来，以Docker为代表的容器技术得到迅猛发展。虽然容器技术具有高度的敏捷性，可重用性和可移植性等优点，但是在大规模运用场景上，因为安全问题，容器技术也存在着巨大的安全隐患。现如今，越来越多的国内外企业进行了容器技术的开发与应用，其在云计算基础设施和服务中的部署量也是逐年增加，可以说已经引发了新一轮的信息技术革命。所以，容器的安全隐患问题也得到了越来越多的重视。

**关键词：**容器；安全隐患；Docker；

引言

随着以Docker为代表的容器技术广泛的被人们认识与应用，随之而来的安全性问题也被人们所担忧，但这同时也促进了容器技术的完善和发展。本文基于对多篇文献的阅读和理解，介绍了有关容器以及容器安全隐患问题的一些现状以及思考。

1. 容器技术

目前来说，虚拟化主要有两种模式，分别是基于虚拟机监控程序平台和基于容器技术的轻量级虚拟化平台。

基于 Hypervisor（a），（b）与基于容器(c)虚拟化架构的不同（如图1-1所示）。

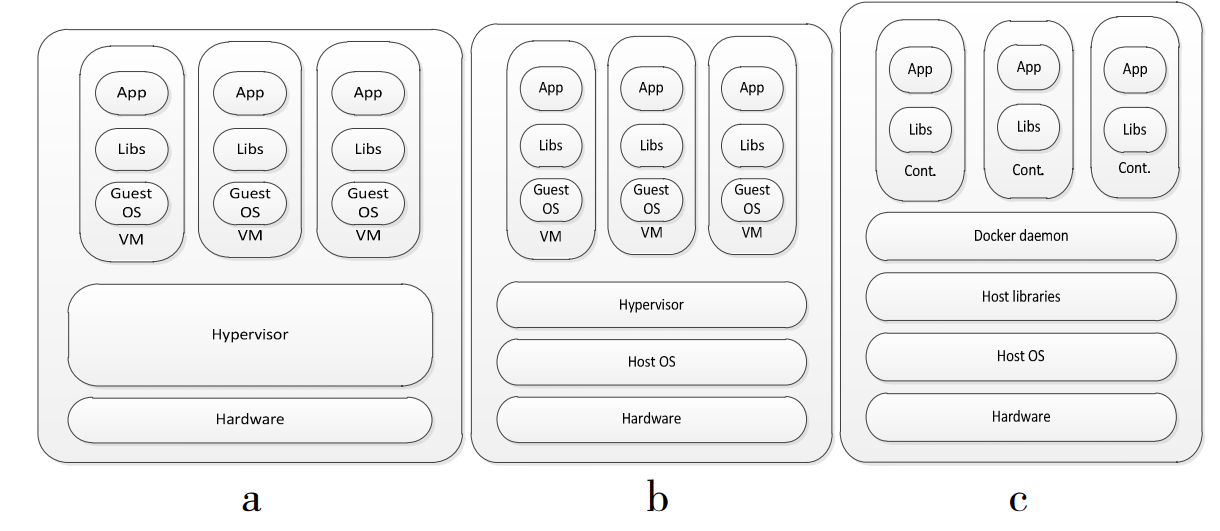


图 1-1

容器并不是一个完全新兴的概念，最早的容器技术可以追溯到1982年Unix系列操系统上的chroot工具。而最近基于Linux的容器则是依靠内核的支持，根据内核提供的系统调为接口。有两种基于内核的方式实现的容器：Linux container（LXC）使用cgroups 和 namespaces，以及 OpenVZ 分支。在LXC的基础上，Docker进一步优化了容器的使用体验，并且提供了各种容器管理工具，让用户无需关注底层的操作，可以简单明了地管理和使用容器。在Linux上，Docker使用 cgroups来实现对进程资源（比如 CPU，内存，网络）的限制，同时使用内核的namespace特性来实现对不同进程的隔离。

1. 容器技术优势

2.1 系统架构简单高效

容器技术与传统硬件虚拟化系统架构比较（如图2-1所示）。



图 2-1

由图2-1可以看出，使用容器技术的主机比使用硬件虚拟化技术的主机具有更少的系统层次且具有唯一的操作系统内核，因此应用程序指令到机器指令的转化层级更少，统一的系统缓存和进程调动效率更高。

2.2 存储和内存需求小

容器技术并不需要在用其中构建一个完整的操作系统，只需要应用程序文件及所需的运行时库文件就可正常工作。而在基础镜像和写时复制技术的帮助下，容器甚至不需保存应用程序本身，因此与传统虚拟机GB的存储需求相比，容器以MB计算的存储需求要小得多。同样的，由于已不再需要加载完整操作系统内核所需的内存和交换空间，因此在运行时容器所占用的内存资源也比较少，几乎不占用多余的内存。

2.3 部署启动速度快

部署一个容器可以从基础镜像直接构建,避免了传统虚拟机部署时克隆磁盘或安装操作系统的步骤,大大缩短了部署时间。由于避免了传统虚拟机加载内核完成系统初始化的繁琐程序,因此在启动速度上也具有先天的优势。

2.4 运行性能好

根据研究,容器技术在计算效率、内存访问、磁盘I/O、 网络I/O方面的性能普遍好于传统硬件虚拟化,高负载下的扩展性能也存在优势。

3 Docker 虚拟化的应用

大部分人都将Docker直接与虚拟机做比较，因为两个技术在设计上是等效的。但其实轻量级容器虚拟化的设计目标是完全不同于虚拟机的。大体上由有以下三种适合Docker的应用。

1）开发者使用微服务模式来使用Docker —— 一个容器必须托管单个服务，在其中运行着单个的进程或者守护进程的子程序。因此，一个容器不应该视为一个虚拟机：没有包的管理，没有init进程，没有ssh连接。所有的管理操作（容器的停止，重启，备份，更新，创建等）都应该通过主机来进行操作，这就需要root权限。

2）使用Docker来可重用和自动部署应用程序。Docker镜像通过一个通用的构建文件（dockerfiles）随时随地的构建，它指定了从基本镜像构建镜像的步骤。这种构建镜像的通用方法使得进程和镜像构建过程几乎不受主机影响，只取决于内核，而不是依赖库。

3）使用Docker搭建PPAS平台。现在主流的PaaS平台都已经集成了Docker，比如 Amazon Web Services和Google Container Engine，创建虚拟机的集群，然后通过编排工具来管理各个虚拟机中的容器。Docker作为一个容器引擎向PaaS提供了基础的资源隔离和标准化打包部署能力，使基于其上的PaaS平台构建简单高效。

4 云计算环境使用容器技术的信息安全风险分析

云计算基础设施的构建是以传统IT基础设施为基础, 因此云计算安全技术体系离不开传统安全技术的支持。但云计算中虚拟化技术的大量使用要求在传统安全技术的基础上应用适应虚拟化工作方式的云计算特有的安全技术（如图4-1所示）。



图 4-1

目前来说，传统安全技术仍然在云计算安全技术体系中居于主导地位，是实现云计算信息安全的必要田间。而云计算特有的安全技术是当前技术发展的热点，现有云计算安全技术主要包括以下方面。

4.1 宿主机安全

由于宿主机对运行其上的虚拟机具有绝对的操作权限,一旦宿主机被攻破,所有虚拟机都将面临危险,因此保护宿主机安全是整个云计算信息安全的关键环节。

目前云计算宿主机主要有两种部署形式,一种是安装全功能的操作系统(如Windows Server、Rad Hat Enterprise Linux等),宿主机在承载虚拟机运行的同时自己也可以执行应用程序，此时可将宿主机视为传统性网络服务器，传统服务器安全技术（如防火墙、入侵检测、访问控制等）仍然是安全可靠的。另一种部署方式是安装虚拟主机专用操作系统（如VMware ESXI、Windows Hyper-V Server等），此时宿主机只用于承载虚拟机，自己并不具备运行应用程序的条件，因此像防病毒软件等很多传统安全方法都无法使用，但由于此种宿主机功能单一所以漏洞很少，在合理配置的前提下配合防火墙等其他传统安全技术也能保证安全。

4.2 虚拟机隔离

同传统服务器不同，同一台宿主机上的虚拟机共享本地资源，并不具备天然的隔离性。虚拟机的安全隔离主要为内存隔离和I/O隔离，此功能主要在Hypervisor中实现。

4.3 虚拟机加固

虚拟机加固包括安全更新，反病毒扫描等。虽然单一虚拟机的加固和传统服务器加固并无不同，但在虚拟化环境下大量虚拟机同时进行安全更新或同时进行反病毒扫描很容易引起宿主机系统资源耗尽而宕机，同时长时间不在线的虚拟机可能因无法及时获取安全更新而暴露系统漏洞，因此在虚拟化环境中部署专用的安全更新和反病毒虚拟设备是十分必要的。

4.4 虚拟机安全监控与审计

目前虚拟机安全监控与审计主要有内部监控和外部监控两种方式。内部监控是指在被监控的目标虚拟机中家在监控代理，将目标虚拟机的状态与系统操作截获，并发送到安全虚拟机或管理域中的安全监控程序，再根据安全策略进行不同处理，从而实现对目标虚拟机的安全监控与审计。外部监控是利用Hypervisor的功能截获目标虚拟机的CPU、内存页面、I/O访问等信息与系统操作，发送到安全虚拟机或管理域中的安全监控程序，从而实现对目标虚拟机的安全监控与审计。

4.5 虚拟网络安全防护与入侵检测

同一宿主机上的虚拟机之间网络通信一般只通过虚拟交换机进行，并不会出现在物理网络中，此时传统的网络安全防护与入侵检测方法将不再发挥作用，需要使用虚拟网络安全防护与入侵检测技术加强虚拟机信息安全。

目前对虚拟网络流量进行防护与检测主要有两种方法。一种方法是通过专门的虚拟机配合虚拟交换机进行虚拟网络流量的防护与检测。另一种方法是通过重定向虚拟网络流量到外部网络交换机上，并使用传统安全设备对这部分流量进行防护与检测，以节省主机计算资源。

5 容器技术环境下信息安全风险分析

由于现有的云计算信息安全技术主要针对的是传统硬件虚拟化，而容器技术与传统硬件虚拟化又存在本质上的区别，因此在云计算环境大量应用容器技术将会带来新型的信息安全风险。

5.1 容器提权风险

在容器技术环境下，容器内部的进程实际上就是容器主机的进程，一旦容器内部进程通过某种方式获得超级用户权限，此时该进程实际上就获得了整个容器主机的超级用户权限，可以对容器主机和同主机的其他容器进行任意操作。

5.2 容器逃逸风险

容器内部的文件系统实际上是容器主机文件系统的一部分，容器理论上是可以便利容器主机文件系统的。实际上已经有研究表明，可以利用容器漏洞遍历并读取容器主机文件，获取敏感信息。

5.3 单一系统内核的风险

在传统云环境下可以通过Hypervisor监控具体虚拟机行为，而在容器技术环境下，由于容器和内核之间没有中间层，所以只靠现有信息安全技术无论是针对云环境还是针对传统主机都无法对具体容器行为进行安全监控与审计。

5.4 容器间通信的风险

逻辑上传统虚拟机之间进行通信仍然通过网络进行，需要虚拟交换机进行网络交换。但容器之间通信除使用网络之外，同一主机上的容器间还可能通过其他方式进行通信（例如，Docker就可以将多个容器连接在一起使用，此时容器间的通信不通过网络进行），此种情况下现有信息安全技术将无法对容器间通信进行防护与检测，形成信息安全风险。

5.5 容器基础镜像风险

现代容器技术普遍使用只读基础镜像构造容器，统一基础镜像可能被大量容器共享使用。虽然一般上认为基础镜像是安全的，但是这一点并不能得到保证。及时寄出镜像确实是安全的，也并不能防止被篡改的可能。目前尚无对寄出镜像安全与否的判断或检测标准，也缺乏相应的技术产品，一旦容器的基础镜像中被植入恶意代码，会给以其为基础的容器带来巨大的信息安全风险。

6结束语

与传统的硬件虚拟化技术比较，以Docker为代表的高级容器技术在管理和运维方面存在巨大的优势，并且其在云计算领域广为流行已经是不争的事实。然而由于信息安全技术的滞后性，目前在云计算领域使用容器技术仍然拥有信息安全风险。需要接下来进一步的研究和发展，并且现在应当在效益和安全之间做好平衡。随着容器技术发展的日益成熟，相信容器信息安全技术也会越来越成熟，最终使用容器技术所面临的安全风险最终是能够达到可控程度的。

**参考文献**

1. Docker Bench for Security. <https://github.com/docker/docker-bench-security>.
2. Managing configuration drift with Salt and Snapper. https://www.suse.com/communities/blog/managing-configuration-drift-salt-snapper/.
3. Maximizing IT Security with Configuration Management. <https://puppet.com/resources/whitepaper/maximizing-it-security-configurationmanagement>.
4. Poorly managed SSH keys pose serious risks for most companies. <http://www.computerworld.com/> article/2488012/ malware-vulnerabilities/poorlymanaged-ssh-keys-pose-serious-risks-for-most-companies.html.
5. Securing your SSH Server. <https://blog.rackspace.com/securing-your-ssh-server>.
6. Security Guide: How to Protect Your Infrastructure Against the Basic Attacker. http://blog.mailgun.com/security-guide-basicinfrastructure-security/.
7. Security Notice: OpenSSH Passwords. <http://authy.com/blog/openssh-passwords-vulnerable/>.
8. Someone said that 30vulnerabilities.https://jpetazzo.github.io/2015/05/27/docker-images-vulnerabilities/.
9. Understanding Docker Security And Best Practices. <https://blog.docker.com/2015/05/understanding-docker-security-and-best-practices/>.
10. AMAZON EC2 CONTAINER SERVICE. <https://aws.amazon.com/ecs/>.
11. AMAZON INSPECTOR. <https://aws.amazon.com/inspector/>.
12. AQUA. <https://www.aquasec.com/>.
13. AZURE CONTAINER SERVICE. <https://azure.microsoft.com/en-us/services/container-service/>.
14. CALLANAN, M., AND SPILLANE, A. Devops: Making it easy to do the right thing. *IEEE Software 33*, 3 (May 2016), 53–59.
15. COMMON VULNERABILITIES AND EXPOSURES. https://cve.mitre.org/.
16. CONTAINER ENGINE. <https://cloud.google.com/container-engine/>.
17. COREOS CLAIR. <https://coreos.com/blog/vulnerability-analysis-for-containers.html>.
18. DOCKER SECURITY SCANNING. <https://docs.docker.com/docker-cloud/builds/image-scan/>.
19. GUMMARAJU, JAYANTH AND DESIKAN,TARUNAND TURNER, YOSHIO. Over 30% of Official Images in Docker Hub Contain High Priority Security Vulnerabilities. <https://www.banyanops.com/blog/analyzing-docker-hub/>.
20. IBM BLUEMIX. <https://console.ng.bluemix.net/>.
21. IBM VULNERABILITY ADVISOR. <https://www.ibm.com/blogs/bluemix/tag/ibm-vulnerability-advisor/>.
22. KOLLER, R., ISCI, C., SUNEJA, S., ANDDE LARA, E. Unified monitoring and analytics in the cloud. In *7th USENIX Workshop on Hot Topics in Cloud Computing (Hot Cloud 15)* (Santa Clara, CA, 2015), USENIX Association.
23. LUK, C.-K., COHN, R., MUTH, R., PATIL, H.,KLAUSER, A., LOWNEY, G., WALLACE, S.,REDDI, V. J., AND HAZELWOOD, K. Pin: Building customized program analysis tools with dynamic instrumentation. In *Proceedings of the 2005 ACM SIGPLAN Conference on Programming Language Design and Implementation* (New York, NY, USA, 2005), PLDI ’05, ACM, pp. 190–200.
24. NATIONAL VULNERABILITY DATABASE. https:/nvd.nist.gov/.
25. RAFAEL BENEVIDES. 10 things to avoid in docker containers. <https://developers.redhat.com/blog/2016/02/24/10-things-to-avoid-in-docker-containers/>.
26. ROCHE, J. Adopting devops practices in quality assurance. *Common. ACM 56*, 11 (Nov. 2013), 38–43.
27. SHERMAN, A., LISIECKI, P. A., BERKHEIMER,A., AND WEIN, J. Acms: The akamai configuration management system. In *Proceedings of the 2Nd Conference on Symposium on Networked Systems Design & Implementation - Volume 2* (Berkeley, CA, USA, 2005), NSDI’05, USENIX Association, pp. 245–258.