

Docker 安全性研究

鲁涛,陈杰,史军

(江南计算技术研究所,江苏无锡 214083)

摘要:在云计算应用日益广泛的今天,虚拟化作为其中的关键技术发展迅速,特别是 Docker 容器技术以轻便性和高性能优势而成为研究者关注的热点,与其相关的安全性研究也愈受重视。在对 Docker 架构及组件构成简要介绍的基础上,先从网络通信、镜像构建、容器运行、仓库存储、内核支持和软件本身六个角度对现阶段 Docker 存在的安全威胁进行全面分析归纳,再从安全支撑、内核安全和数据中心安全三大层面对 Docker 安全防护技术进行全方位的梳理论述,阐明了每种安全防护技术的优势及需要完善改进的不足之处,并总结了日志审计、威胁检测和安全模型等其他安全防护技术。不仅呈现出当前 Docker 安全研究的概貌,同时明确指出了 Docker 安全领域亟需解决的一些关键问题,探讨与展望了 Docker 安全研究的发展趋势。

关键词:Docker;容器;威胁;安全防护

中图分类号:TN915.08

文献标识码:A

文章编号:1673-629X(2018)06-0115-06

doi:10.3969/j.issn.1673-629X.2018.06.026

Research of Docker Security

LU Tao, CHEN Jie, SHI Jun

(Jiangnan Institute of Computing Technology, Wuxi 214083, China)

Abstract: With the increasing use of cloud computing, virtualization has developed rapidly as the key technology. Especially, Docker container technology has become the focus of researchers because of its advantages of easy deployment and high performance. Consequently, the research of Docker security has become more and more important. Based on the brief introduction of Docker architecture and components, the security threats of Docker are analyzed and summarized from six perspectives including network communication, mirror construction, container operation, image storage, kernel support and software itself. Then we discuss the Docker security protection technology from three aspects including security support, kernel security and data center security, and point out the advantages and shortcomings of each security technology. At last, we summarize other security technologies such as log audit, threat detection and security model. We not only show the current Docker safety research profile, but also clearly point out the key issues that Docker security areas need to solve, and explore and look forward to the development trend of Docker security research.

Key words: Docker; container; threat; security protection

0 引言

云计算因为能够提供灵活的弹性服务,具备快速部署能力及便携性特点而在当今计算领域占据着大部分市场份额^[1],其中资源虚拟化技术在实现云计算基础设施服务按需分配中起到了至关重要作用^[2]。Vmware、Xen、KVM 和 Microsoft Hyper-v 等传统硬件级虚拟化由于要虚拟一个完整操作系统层,降低了资源利用率,对大规模的集群应用来说是不容忽视的缺陷。为寻求更高效的隔离化技术替代方案,Docker 公司将一些现有技术进行包装整合后,于 2013 年 3 月推出了

开源的容器项目 Docker^[3]。也得益于赶上了一场 IT 开发行业从 SOA 架构向微服务架构的变革^[4],Docker 迅速取得了成功,广泛应用于各种场景。应用的广泛性使得对其安全性的研究工作也更具紧迫性和现实意义。

鉴于此,文中将从 Docker 安全的视角出发,针对性梳理了现阶段 Docker 安全方面的研究成果,总结归纳了 Docker 面临的安全威胁和现有的安全防护技术方法,为 Docker 安全的后续深入研究提供了有价值的参考。

收稿日期:2017-07-06

修回日期:2017-11-08

网络出版时间:2018-02-24

基金项目:国家自然科学基金(91430214)

作者简介:鲁涛(1986-),男,硕士研究生,研究方向为云安全。

网络出版地址: <http://www.cnki.net/kcms/detail/61.1450.TP.20180224.1521.064.html>

1 Docker 架构及组件

Docker 是基于客户端服务器模式设计的,其主要组成可分为 Docker Client、Docker Daemon、Docker Regeister 三部分。Client 通过 tcp 或 unix 套接字向 Daemon 发送请求命令获取服务。Daemon 运行于宿主机中为客户端请求提供服务,并在需要时向 Regeister 发送请求,进行提取或推送镜像等操作。Regeister 主要任务是处理客户端请求,进行用户认证,并管理存储镜像,对请求服务进行响应。三者间的具体关系如图 1 所示。

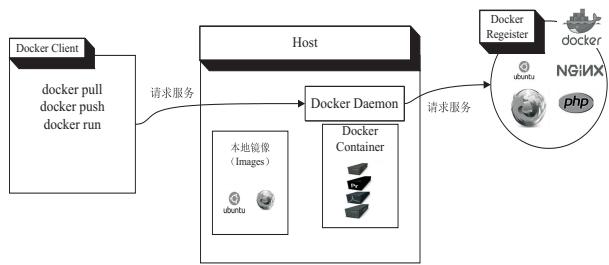


图 1 Docker 主要组件间关系

Docker Client 是 Docker 为用户提供的操作接口,可通过该接口输入命令完成对镜像、容器、仓库的操作。它可与 Daemon 位于同一宿主机中,也可以是位于远程主机中的客户端。其主要任务是和 Daemon 建立安全的连接,解析并处理命令行中的部分信息,使其能被 Docker Daemon 识别,辅助收集 Docker 客户端部分配置信息^[5]。

Docker Daemon 是常驻后台的系统进程,其主要作用是接收并处理 Docker Client 请求,管理所有 Docker 容器。默认情形下,其仅监听本地 Unix socket,这种情况下宿主机以外的客户端将无法进行远程访问,可以通过对配置文件的修改启用 tcp 监听,对远程访问提供支持^[6]。

Docker Regeister 又称为 Docker 仓库,其主要功能是存储镜像。根据使用范围又可以分为私有仓库和公共仓库。私有仓库一般是公司或个人自己搭建的只供内部用户使用的仓库。公共仓库一般是指可在外网上提供公共服务仓库,用户通过认证后可对镜像进行拉取、上传等系列操作。

2 Docker 安全威胁

根据 Docker 不同组成部分,从网络、镜像、容器、仓库、Docker 所依赖的 Linux 内核、Docker 软件本身等六个角度逐类分析了其中存在的安全威胁。

2.1 Docker 网络安全威胁

Docker 容器网络默认使用桥接方式进行连接,其在宿主机上创建一个虚拟的网桥 Docker0,它本身扮演了传统交换机的角色,在各个网络接口间自动地进

行包转发,每创建一个新的容器,就为其增加一个虚拟网络接口,并将该网络接口连接到网桥 Docker0。默认情况下,虚拟网桥并没有对转发的数据包进行任何过滤,因此这种连接方式容易遭受 ARP 欺骗和 MAC 泛洪攻击,位于同一宿主机中的容器也更易受到临近被感染容器的网络攻击。通过 docker run - p 命令指定对外服务端口时,操作不当可能致使暴露端口被映射到宿主机的每个网卡上监听外界连接,一些情况下会带来难以预料的安全风险^[7]。

2.2 Docker 镜像安全威胁

开发者在构建镜像时可能由于疏忽大意将包括数据库认证密码在内的敏感信息添加到镜像中,为生产环境中部署的应用埋下安全隐患;镜像在方便进行传输和便于部署的同时,也为含病毒后门的镜像恶意传播提供了便利。

在构建镜像过程中,现在使用的基础镜像都比较大,如基于 ubuntu 和 Debain 构建的镜像,由于包含与实际用途不相关的库和依赖项,可能会引入额外漏洞。Rui Shu 等在其《Docker Hub 安全漏洞分析》一文中给出了一份详细的统计数据^[8]。该数据如表 1 所示,由数据可以看出,无论是社区镜像或官方镜像都有较多的漏洞。在使用镜像前,必须要检测镜像中是否含有漏洞,在制作镜像过程中要尽量减少引入不相关的依赖库。在镜像的漏洞扫描方面尽管有 Docker Security Scanning、Clair 等安全扫描软件,但目前这些软件仍无法给出一种通用的解决方案以识别所有类型镜像中都安装了哪些软件依赖项。

表 1 镜像中所含漏洞数量统计

镜像类型	总镜像数	平均漏洞数	最大漏洞数
社区镜像	352 416	199	1 779
社区镜像 latest 版	75 533	196	1 779
官方镜像	3 820	185	791
官方镜像 latest 版	93	76	392

2.3 Docker 容器安全威胁

容器运行时有大量的状态向量和进程信息与操作系统相关联,捕获并转移这些信息相比于虚拟机更难操作,所以目前还不能像虚拟机一样支持热迁移。而 Docker 这种基于 Daemon 为中心运行的容器,在 Daemon 出现运行故障或者需要升级时只能中断运行。这对需要保持持续运行的业务将是灾难性的事故。

容器运行时存在一类配置不当引起的安全问题。当容器以 Privileged 特权运行时,容器内用户能获得宿主机绝大部分资源支配权;配置容器重启策略时,如不限制容器重启次数,反复的重启可耗尽计算资源,引发拒绝服务攻击;将敏感文件挂载到容器,可导致难以想

象的后果,若将/var/run/目录挂载到容器后,导致远程用户可不加认证对宿主机进行任意访问的后果。

2.4 Docker 仓库安全威胁

为确保镜像的传输安全,Daemon 和 Registry 通信过程启用了 TLS 安全连接,为防止镜像被篡改,Docker 1.8 之后版本中提供了镜像签名和校验功能^[9]。但这中间仍然存在一些问题,如在本地私有网络中搭建的 Registry,为了便利会给 Daemon 配置-insecure-registry 参数启用不安全通信连接,并通过传递-disable-content-trust 参数禁用 Daemon 签名校验功能;在 Docker Hub 中有成千上万的开发者,每个开发者都可以利用自己的私钥对镜像进行签名,签名校验机制并不能从源头上确保开发者可信问题^[10]。

Docker Hub 为方便用户升级镜像,实现了镜像自动构建(automated builds)功能。该功能允许 Docker Hub 跟踪指定的 GitHub 或 BitBucket 项目,一旦项目发生新提交,则自动重新构建镜像,并将镜像部署到实际生产环境中。在该过程中,会自动从第三方资源平台下载依赖项,而第三方资源平台可能是不安全的链接,传输过程中依赖项极有可能被恶意篡改^[11],这对 Docker 也是一项不安全因素。

2.5 Linux 内核支持安全威胁

大批研究者致力于发现可被用来执行任意代码或能进行本地提权攻击的内核漏洞,由此发现的大量内核漏洞对基于和宿主机共享内核机制构建的 Docker 容器也是非常大的威胁。另外 Linux 内核层面提供的支撑技术还不完全成熟,从隔离性上来看,尽管有 Namespaces 提供隔离防护,但是仍有很多内容并没被加入其中来保证完全隔离,这其中包括伪文件系统、系统时钟和 LSMs 中的一些特征等^[12]。系统运行所必须的关键目录若没有实现完全的隔离是非常危险的,例如/proc、/sys 下的子目录中含有系统运行的关键信息,直接暴露给容器会造成信息的泄露,并为恶意用户发起攻击创造了有利条件。Gao X 等论述了如何利用/proc 目录中泄露的信息,选择最佳时机,启动多个容器,以更小代价对数据中心发起电力攻击^[13]。

2.6 Docker 软件安全威胁

在 Docker 1.10 之前的版本中,Daemon 在创建容器时需执行很多特权操作,包括挂载文件系统、配置网络等,必须由 root 用户启动。这在 Daemon 被突破后,宿主机也将跟着沦陷^[9]。虽然在新版本中,Daemon 被置于新建的 Docker 用户组下运行,并利用安全可靠子进程来代理执行需要特权权限的操作。但在现实实践中,Docker 用户组并不能很好地保证宿主机安全。

Docker Daemon 对镜像、Dockerfile 文件解析过程中,如不能对这些输入数据进行很好的过滤,可能导致

严重的安全问题。历史上就出现过通过构造特殊 Dockerfile 压缩文件,在编译时触发漏洞获取执行任意代码权限的严重事件。早期版本中也出现过本地用户通过在 Docker Image 中构造特殊的软链接进行提权的问题。

3 Docker 安全防护

3.1 Docker 安全防护支撑技术

Docker 提供的隔离特性大多都是通过集成先前已有技术实现的,这包括 NameSpaces、Cgroups(control groups)和联合文件系统(UnionFS)特性等技术,这三项技术为 Docker 的安全隔离提供了基础架构支撑,具体关系如图 2 所示。

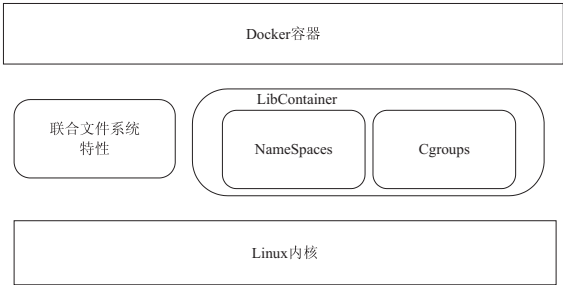


图 2 Docker 安全防护关键支撑技术

(1)NameSpaces 由一组 NameSpace 组成,主要作用是确保处于同一 NameSpaces 下的进程相互之间是可见的,不同 NameSpaces 的进程之间则相互隔离。Docker 环境中目前可支持六种不同的 NameSpace,它们分别对系统资源的不同方面进行了隔离,具体资源隔离情况如表 2 所示。

表 2 NameSpaces 资源隔离

NameSpaces	隔离系统资源
Pid	进程编号
Net	网络设备、网络栈、防火墙规则等
IPC	信号量消息队列和共享内存
Mnt	挂载点(文件系统)
Uts	主机名或域名
User	用户或用户组

NameSpaces 提供的隔离机制为容器提供了相对安全的环境。但正如前文所提到的这种隔离并不彻底,且从今后发展看,对 NameSpaces 的完善工作会使内核过于复杂,能否为 Linux 内核主线所接收也是未知数^[14],如何在不增加内核冗杂性的基础上提供更好的隔离方案,是一类亟需研究的安全防护问题。

(2)Cgroups 最大特点是可通过使用其各子系统对组内进程资源供给进行灵活组合限制^[15],解决了多租户容器平台中差别化的系统资源分配问题,减少了资源耗尽型 Dos 攻击的攻击面。其由很多子系统组

成,每个子系统实现对不同类型资源的控制^[16],具体对照关系如表 3 所示。

表 3 Cgroups 子系统功能对照表

子系统	功能
cpuset	管理组中进程的可用 cpu 和内存节点
debug	用于 Cgroups 自身的调试
cpu	控制进程可获得的 cpu 时间
cpuacct	统计进程已经使用的 cpu 时间
mem	内存管理和统计功能
devices	控制组中进程开发访问哪些设备
freezer	用于暂停和恢复组中进程
net_cls	对进程的套接字赋予标签
blkio	控制块设备输入输出限制
perf	对进程的性能监测
net_prio	设置进程网络优先级
hugetlb	设置 hugeTLB 相关参数

Cgroups 存在的缺陷同样是资源限制上不能涵盖全部类型系统资源,尽管 Linux 提供的另外两种资源限制机制 rlimit(resource limit)和配额(quota)机制可弥补 Cgroups 的部分不足之处,但是对 Cgroups 的丰富和完善也是今后安全研究工作的重要方面。

(3)联合文件系统是一种分层、轻量级的高性能文件系统,它支持对文件系统的修改作为一次提交并层层叠加,并将不同目录挂载到同一个虚拟文件系统中为用户提供一个统一的视图^[17]。

早期 Docker 产品中,镜像构建和容器运行所需层级结构、写时复制技术都依赖于联合文件系统提供的底层支持。随着 Docker 的发展,后添加的 Device Mapper 等驱动存储虽不属联合文件系统,但也可对层级结构、写时复制技术提供支持,统称这些驱动存储技术具有联合文件系统特性。Docker 的这种层级结构和写时复制特性确保了运行中容器不能直接对底层镜像文件进行修改,保证了镜像的稳定性,同时建立于其上的内容寻址保证了镜像中的每个层级都有一个唯一的校验 ID,从而确保镜像的完整性^[18]。

3.2 Docker 内核安全防护

确保内核安全和利用内核提供的安全技术进行自身安全加强都是 Docker 安全的重要方面。Docker 在这方面也做了很多整合工作,主要是通过采用能力机制(capabilities)、Seccomp(secure computing mode)安全特性、强制访问控制(mandatory access control,MAC)对容器的权能和系统调用进行限制;通过利用内核相关安全模块 Iptables、Traffic controller 来加强自身网络安全,以及通过利用内核相关完整性保护技术确保镜像和系统数据的完整性等;通过已有安全框架 Grse-

curity/PAX 等对内核进行安全强化。

(1)能力机制将 Linux 权限进行了更细粒度的划分,改变了过去 root 用户拥有全部特权现象。能力机制将特权细分为 37 种不同的能力^[14],涵盖了对文件、进程、网络、系统、设备等各方面的操作权限。Docker 默认情况下为容器提供了有限能力权限,用户也可根据实际需要在运行时添加或减少能力特权。能力机制方面仍存在的问题是,权能的划分在一些方面仍不够细致,不能很好地满足对容器的限制要求,CAP_SYS_ADMIN 就覆盖了很多的特权,需要进一步细分。为容器分配权限时,很难知道容器内应用需要哪些特权。为兼顾此问题,Docker 提供的默认能力权限列表包含了较宽泛的范围,从安全的角度考虑则并不是个好策略。

(2)Seccomp 可以限制用户进程的系统调用,对系统调用参数进行筛选。系统调用是用户态和内核态间最重要的接口,通过限定它可在很大程度上确保进程在安全可控范围内运行。Docker 默认基于白名单机制使用 Seccomp 配置文件规定允许进行哪些系统调用,有 50 多种系统将会被阻止,部分系统调用在特定参数的情况下也会被阻止。该配置文件下,大部分容器都可正常运行。

(3)强制访问控制相对于自主访问控制(discretionary access control,DAC)可提供更严格的保护,用户由于不能改变他们的安全级别和对象的安全属性,只能由 MAC 根据事先定义好的用户及数据安全等级匹配结果做出客观的具体决策,这样就能屏蔽掉用户主观因素而更好地保障系统安全。在 Linux 系统中,强制访问控制一般会 and DAC 配合使用,并在 LSM(linux security module)框架下实现具体安全模块^[19]。已经实现并合并到内核主线的安全模块有 SELinux、AppArmor、Yama、SMACK、Tomoyo,这些安全模块都可为 Docker 容器提供增强的安全防护。各模块间除 Yama 外,同时使用会存在互斥现象,一般会根据 Linux 发行版本的不同而选择不同的安全模块。Debian 系列一般使用自带的 AppArmor,Redhat 系列发行版本则选择自带的 SELinux 安全模块。

其中 SELinux 的问题主要是操作过于复杂,开发者和实际生产环节对启用 SELinux 都持抵制态度,而且在使用上和 BTRFS 驱动也不兼容。AppArmor 的独特之处在于它并不关注全系统的安全,只会为特别标明进程提供强制访问控制,其他进程都工作在不受控制的状态。Mattetti M 等介绍了一种 LiCSHield 安全框架用于加固 Linux 容器,该框架通过跟踪分析容器运行所需权限,有针对性地自动生成一个 AppArmor 配置文件,来确保容器后续运行的安全可控^[20]。除这些

安全模块外,用户也可根据 LSM 框架设计更适合自己的安全模块,李平等做了相关方面研究^[21]。

(4)在 Linux 内核中有一些网络框架可用来解决 Docker 网络安全问题,这包括 Netfilter 框架和 TC (traffic controller)框架。Netfilter 侧重于防火墙数据过滤,而 TC 则侧重流量带宽控制。Docker 可通过修改配置文件启用 Iptable 防火墙,配置防火墙策略来控制容器间的通信^[22],可通过 TC 限制容器在宿主机上的虚拟网络接口流量而达到限制容器带宽的目的^[18],或者结合 Cgroups 子系统标记网络数据包,利用 TC 中的分类队列限制容器的通信流量,有效防止独占网络带宽类型拒绝服务攻击。

(5)Linux 内核中的完整性保护相关技术可很好地应用到 Docker 中,确保镜像和容器运行过程中的完整性。涉及到的技术包括可信计算开放标准(trusted computing group,TCG)在 Linux 具体实施的部分模块及 Device Mapper 下实现完整性校验功能的 dm-verity 子模块。这些技术都可以用来实现 Docker 镜像的完整性保护,其中王鹄等利用 TCG 中的 TPM 做基础支撑,实现了镜像的完整性度量^[23],Hosseinzadeh S 等也提出了两种基于 vTPM 实现容器可信度量的方法^[24],而目前利用 dm-verity 来实现镜像的完整性校验的研究则较少。

3.3 Docker 数据中心安全防护

Docker 数据中心产品中两个重要组成部分: DTR(Docker trusted registry)、UCP(Docker universal control plane)。为确保数据中心的安全性,它们增加了许多新的安全特性,在启用 DCT(Docker content trust)框架的情况下可很大程度上保障数据中心的安全。

(1)Docker content trust 的主要功能是对从 Docker registry 传送进来的 Docker 镜像进行完整性和发布者真实性校验,为要推送到 Docker registry 中的镜像进行数字签名。DCT 在功能实现上采用了 Docker notary 及其封装的 TUF(the update framework)框架。在配置了 Docker notary 服务的 Docker registry 环境下, DCT 还可以实现更为高级的功能,比如门限签名(threshold signing)功能。

(2)Docker trusted registry 是 Docker 容器云的核心组件,它允许企业在本地或私有云中存储及管理它们的镜像资源。相比 Docker registry,除了提供更好的易用性外也增加了额外的安全特性,包括镜像签名、基于角色的访问控制(role based access control)、LDAP/AD 的身份认证机制、镜像扫描功能^[25]。

(3)和 DTR 相比,UCP 的主要功能则是侧重于本地及私有云环境中容器集群的管理和部署,而不是镜

像存储。在安全方面同样集成了基于角色的访问控制、LDAP/AD 的身份认证机制,除此之外,还提供了增强的 TLS 安全机制、Secrets 机制及日志记录集中管控功能。

3.4 其他相关安全防护

(1)Docker 尽管在日志的可视化审计方面有比较系统的实施方案^[7],但在大规模日志数据深入挖掘,自动化分析方面,仍有待进一步研究^[26];在运行环境审计方面,可参考 Docker 公司和 CIS(center for Internet security)合作制定的一份安全性评估文档,其相应的具体评估实施工具为 Docker bench for security。

(2)在针对 Docker 的安全威胁检测方面,尽管可采用传统基于主机的威胁检测与基于数据流的威胁检测技术,但如何结合 Docker 自身的特点进行针对化应用还有待进一步加强。Abed A S 等详细介绍了如何通过检测 Docker 中进程的系统调用,对定义的调用频率进行聚类分析,识别出异常入侵行为^[27]。该方法就属于将传统基于主机威胁检测技术应用于 Docker 具体环境中的一个实例。

(3)从不同的视角出发,对 Docker 的安全问题进行研究的内容还有很多。比如,为提供更安全的容器运行环境,先后出现了一批致力于改造出更适合容器运行的精简操作系统项目,包括 CoreOS、Ubuntu Core 等;一些研究者则从剔除容器对宿主内核依赖性的角度出发,提出了一种新型的更轻量级架构方案,该方案将借助定制的内核使应用能够直接在裸机上运行,相互之间不再共享内核系统,代表性的解决方案有 Unikernels 等;也有研究者致力于容器系统安全模型标准界定的研究,Catuogno 对 CCRA(common criteria recognition arrangement)界定的容器安全标准模型和 Reshetova 等提出的一种安全模型在 Docker 环境下进行了对照比较,并指出了两种安全模型实质上的等价性^[28];Sharath N 等则介绍了如何利用斯坦克尔伯格模型(Stackelberg model)通过线性规划方法寻求一个可同时兼顾 Docker 安全性和确保运行效率的最佳结合点^[29]。

4 结束语

随着容器应用的日益普及,将有越来越多的敏感应用迁移到容器中,在提供便利部署、高资源利用率的同时,如何加强容器的安全性也成为日益引人关注的问题。文中首先简要介绍了 Docker 的主要架构;接下来对 Docker 面临的安全威胁进行了梳理,指出了其存在的脆弱性;然后对 Docker 的安全防护技术进行了归纳,汇总了现阶段的研究方向,指出了一些亟待研究和解决的关键问题。与传统虚拟技术相比,Docker

容器的安全性还有差距,但两者并不是相对立的,在同样起到虚拟隔离效果的同时也存在优势互补的作用,尽力补足 Docker 安全性上的短板将能更好地发挥其优势。

参考文献:

- [1] 王 斌. 云计算 IaaS 体系架构面向中小企业的商业模式研究[D]. 北京:北京邮电大学,2014.
- [2] 罗军舟,金嘉晖,宋爱波,等. 云计算:体系架构与关键技术[J]. 通信学报,2011,32(7):3-21.
- [3] 李在弘,武传海. Docker 基础与实战[M]. 北京:人民邮电出版社,2016.
- [4] 王 磊. 微服务架构与实践[M]. 北京:电子工业出版社,2016.
- [5] 孙宏亮. Docker 源码分析[M]. 北京:机械工业出版社,2015.
- [6] 张 涛. Docker 全攻略[M]. 北京:电子工业出版社,2016.
- [7] GOASGUEN S. Docker cookbook[M]. [s. l.]: O'Reilly Media, Inc., 2015.
- [8] SHU Rui, GU Xiaohui, ENCK W. A study of security vulnerabilities on Docker Hub[C]//Proceedings of the seventh ACM on conference on data and application security and privacy. Scottsdale, Arizona, USA: ACM, 2017:269-280.
- [9] 华为 Docker 实践小组. Docker 进阶与实战[M]. 北京:机械工业出版社,2016:313-354.
- [10] COMBE T, MARTIN A, PIETRO R D. Containers: vulnerability analysis[R]. [s. l.]: [s. n.], 2015.
- [11] COMBE T, MARTIN A, PIETRO R D. To Docker or not to Docker: a security perspective[J]. IEEE Cloud Computing, 2016, 3(5):54-62.
- [12] GRATTAFFIORI A. Understanding and hardening Linux containers[M]. [s. l.]: NCC Group, 2016.
- [13] GAO Xing, GU Zhongshu, KAYAALP M, et al. Container-Leaks: emerging security threats of information leakages in container clouds[C]//Proceedings of the 47th IEEE/IFIP international conference on dependable systems and networks. Denver, CO, USA: IEEE, 2017:3-15.
- [14] 李 志. Linux 内核安全模块深入剖析[M]. 北京:机械工业出版社,2016.
- [15] MATTHIAS K, KANE S P. Docker: up & running[M]. [s. l.]: O'Reilly Media, Inc., 2015.
- [16] 汪 恺, 张功萱, 周秀敏. 基于容器虚拟化技术研究[J]. 计算机技术与发展, 2015, 25(8):138-141.
- [17] 杨保华, 戴王剑, 曹亚仑. Docker 技术入门与实战[M]. 北京:机械工业出版社, 2017.
- [18] 浙江大学 SEL 实验室. Docker 容器与容器云[M]. 北京:人民邮电出版社, 2016.
- [19] 刘威鹏, 胡 俊, 吕辉军, 等. LSM 框架下可执行程序的限制访问控制机制[J]. 计算机工程, 2008, 34(7):160-162.
- [20] MATTETTI M, SHULMAN-PELEG A, ALLOUCHE Y, et al. Automatic security hardening and protection of Linux containers[C]//Workshop on security and privacy in the cloud. Florence, Italy: Curran Associates, 2015.
- [21] 李平平, 陈莉君. 基于 LSM 的 Docker 访问控制机制研究[J]. 信息技术, 2016, 40(11):134-138.
- [22] DUA R, KOHLI V, KONDURI S K. Learning Docker networking[M]. [s. l.]: Packet Publishing Ltd, 2016.
- [23] 王 鹄, 胡 威, 张雨菡, 等. 基于 Docker 的可信容器[J]. 武汉大学学报:理学版, 2017, 63(2):102-108.
- [24] HOSSEINZADEH S, LAURÉN S, LEPPÄNEN V. Security in container-based virtualization through vTPM[C]//Proceedings of the 9th international conference on utility and cloud computing. [s. l.]: [s. n.], 2016:214-219.
- [25] GALLAGHER S. Securing Docker[M]. [s. l.]: Packet Publishing Ltd, 2016.
- [26] 褚瓦金. 日志管理与分析权威指南[M]. 北京:机械工业出版社, 2014.
- [27] ABED A S, CLANCY C, LEVY D S. Intrusion detection system for applications using Linux containers[C]//International workshop on security and trust management. [s. l.]: [s. n.], 2015:123-135.
- [28] CATUOGNO L, GALDI C. On the evaluation of security properties of containerized systems[C]//International conference on ubiquitous computing and communications and 2016 international symposium on cyberspace and security. Granada, Spain: IEEE, 2016:69-76.
- [29] SHARATH N, KUMAR V, CHANDRASEKARAN K. Solving security issues in Docker using Stackelberg games[J]. IJCTA, 2016, 9(16):8275-8285.