Computer Science and Application 计算机科学与应用, 2019, 9(5), 926-933
Published Online May 2019 in Hans. http://www.hanspub.org/journal/csa
https://doi.org/10.12677/csa.2019.95105

Hans汉斯

A Review of Docker Security Research

Jinjian Yu, Chuan Jia, Dong Pu

Chengdu University of Information and Technology, Chengdu Sichuan
Email: 825149459@qq.com, 137756755@qq.com, puudong@foxmail.com

Received: May 7th, 2019; accepted: May 20th, 2019; published: May 27th, 2019

Abstract

Under the wave of the era of big data, cloud computing platforms have been widely used. Virtualization technology is rapidly developing as the core technology in the field of cloud computing. As the current most widely used container technology, Docker emerges from virtualization technology with its lightweight, fast and efficient advantages. Docker has received a lot of attention and become a hot technology for companies and a focus topic for researchers. This article gives a brief introduction to Docker technology and its development, and then analyzes the various issues faced by Docker in terms of security, and proposes some solutions. Then it puts forward some security guidelines to help prevent security vulnerabilities of Docker. Finally, it summarizes the problems that Docker needs to solve and looks forward to the development trend of Docker research.

Keywords

Virtualization, Container, Docker, Security

Docker安全性研究综述

余金键，贾　川，蒲　东

成都信息工程大学，四川　成都
Email: 825149459@qq.com, 137756755@qq.com, puudong@foxmail.com

收稿日期：2019年5月7日；录用日期：2019年5月20日；发布日期：2019年5月27日

摘　要

在大数据时代的浪潮下，云计算平台得到了广泛应用。虚拟化技术作为云计算领域的核心技术飞速发展，Docker作为当前应用最为广泛的容器技术，凭借其轻量、快速和高效的优势从虚拟化技术中脱颖而出，获得了大量关注，成为了企业追捧的热门技术和研究者探讨的焦点话题。本文首先对Docker技术及其发展进行了简要介绍，再对现今Docker在安全性方面面临的各种问题进行了着重分析，并提出了一些解决

文章引用: 余金键, 贾川, 蒲东. Docker 安全性研究综述[J]. 计算机科学与应用, 2019, 9(5): 926-933.
DOI: 10.12677/csa.2019.95105

方法，接着再提出了一些安全指南帮助预防Docker在安全方面的漏洞。最后进行总结，指出了Docker亟需解决的问题，展望了Docker研究的发展趋势。

关键词

虚拟化，容器，Docker，安全

1. 引言

随着当今大数据时代数据呈爆炸式增长，各种云计算平台开始得到了广泛地使用，其中虚拟化技术作为云计算的关键技术，在过去十年开始飞速发展[1]。在大规模的计算环境中采用传统的虚拟机技术将会占用了很大的系统资源，使得系统主机运行的较为缓慢，Docker 作为一种新的虚拟化技术应运而生。其相较于传统的虚拟机技术更加快速，它的优势主要体现在进程/容器级上，Docker 在启动速度、信息隔离和资源消耗方面的优势十分明显[2]，因此 Docker 成为了当前人们研究虚拟化技术的热点。Docker 提供了一个生态系统，为容器内的应用程序打包，分发和管理提供平台。但是 Docker 平台尚未成熟，特别是在安全性方面仍存在一些问题。目前，与虚拟机(VM)和大多数其他云技术相比，Docker 的安全性较低[3]。本文对 Docker 进行了简要概述，同时对 Docker 存在的一些安全性问题进行了总结，并介绍了 Docker 安全部署指南，最后对 Docker 的发展进行了总结和展望。

2. Docker 技术

2008 年第一个比较完善的容器技术--LXC 发布，其建立在 cgroup 和 namespace 的基础之上。为了追求更加优化的容器技术，2013 年 3 月，Paas 服务提供商 dotcloud 公司(后改名为 Docker Inc)在整合已有的技术后，基于 LXC 技术推出了一款开源的应用容器引擎 Docker [4]。随着互联网公司的广泛应用及各大社区积极支持，加上自身体系架构及相关组件和工具的不断完善，目前 Docker 已成为容器技术领域最重要的技术。Docker 核心是基于谷歌 Golang 语言开发的 LXC，其通过 LXC 对网络信息封装从而实现数据处理的可移植性与规范化，Docker 源代码托管于 Github，并遵从 Apache2.0 协议[5]。Docker 通过内核虚拟化和分层镜像，可以让开发者把开发环境、应用及其所需的依赖包封装到镜像，由于 Docker 不受底层操作系统的制约，因此其可以发布到安装有 Docker 的电脑上，不受电脑使用的操作系统限制。Docker 能够为应用提供快速、高效、灵活的部署方式，同时由于云计算平台正逐渐改变当前 Paas 以及 Iaas 的架构模式[6]，各大云服务供给商都相继推出了基于 Docker 的容器云服务。云计算行业中的各大厂商如 Amazon、IBM、VMware 都纷纷加入了参与到了这个行列中。

传统的虚拟化技术如 VMware、Xen 等属于硬件级的虚拟化，需要模拟一个完整的 OS 层，硬件资源利用低；Docker 是进程级的虚拟化技术，相比于前者，没有了硬件模拟层开销，因此更轻量、快速和高效。在容器内运行应用程序时，与虚拟机技术不同，Docker 可以直接调用系统内核执行命令，不需要经过中间层翻译，因此资源开销更小，启动速度更快，因而可以很容易地在一台服务器上部署成百上千个容器并且快速启动。虽然 Docker 有着诸多优势，但是与其他虚拟化技术相比，安全方面却存在着很大的挑战。

3. Docker 安全性问题

虚拟环境为运行安全性很高的服务提出了很大的挑战，尤其是在多租户云系统中[7]。相较于 Docker，传统的虚拟机技术具有更高的安全性。其关键原因在于虚拟机技术添加了主机和应用程序中间的额外间隔层。

基于虚拟机技术的应用程序只允许与虚拟机内核通信，而不允许与主机内核通信。所以，如果攻击者想要攻击主机内核就必须绕过虚拟机内核与管理程序。同时，容器模型提供给了应用程序直接访问权限并能与主机内核进行通信。此时恶意者便可直接对主机内核进行攻击。与虚拟机技术相比，这是容器技术更易发生安全性问题的关键方面之一。目前，Docker 是排名第一的基于容器的虚拟化平台。Docker 容器也同样存在安全隐患。

3.1. 内核漏洞

内核管理着所有容器的操作和进程。在发生内核级漏洞的情况下，容器内运行的应用程序将处于被攻击者利用的威胁中。所有容器共享相同的内核体系结构。在这种情况下，如果某些应用程序被劫持并获得内核的某些特权，那么这时可能会导致所有正在运行的容器以及主机平台受到威胁。同样地，两个容器不可能使用相同内核模块的不同版本。

Docker 目前非常重视安全性问题，并试图提供更加坚实有效的方案来处理安全性问题。为了处理内核级漏洞，如运行 Docker Engine 时执行 AppArmor 或 SELinux。

要避免此类风险，必须将容器文件系统设置为只读。同时通过关闭容器间通信，也可避免这种攻击。避免在容器中安装不必要的包也是保持这种危险的好方法。

3.2. 拒绝服务攻击

拒绝服务(DoS)是最着名的网络资源攻击之一[8]。在这种攻击中，一个进程或一组进程试图消耗系统的全部资源，从而破坏正常的处理或操作。

在基于容器的系统架构中，所有容器共享内核资源。当一个容器利用对资源的访问权限时，就会发生 DoS 条件，在这种情况下，它会饿死所有其他容器。

如果遇到此类攻击，需通过以下的方案来解决：文件系统隔离，进程隔离，IPC 隔离，设备隔离，网络隔离和控制资源分配。因此，通过控制每个容器的资源分配，可以防止这种攻击。Docker 将 Cgroups(控制组)作为处理此类问题的关键工具。Cgroups 提供了一种计算方法来限制了每个容器的资源，例如 Docker 容器可用的 CPU 时间、磁盘 I/O 和存储空间，它们保证每个容器都能公平地获取资源，避免某个容器垄断所有资源。此外，Cgroup 准许 Docker 对每个容器的资源分配约束进行设置。

3.3. 容器突破

在这种攻击中，攻击者突破容器，然后便能够访问主机和其他容器。在获得访问权限后，恶意者将能对容器外的文件进行访问。

open_by_handle_at()允许进程通过 file_handle 结构访问已挂载文件系统上的文件。file_handle 结构利用 inode 编号来区分文件。要调用此函数，需要 CAP_DAC_READ_SEARCH 功能。默认情况下，容器内的超级用户具有此功能。这允许攻击者绕过 simfs 约束，访问主文件系统上的所有文件，主文件系统包括驻留在类似文件系统上的其他 VE 文件。

从 Docker 的官网可知，容器突破问题只出现在 Docker 版本 0.11 之前。这个漏洞在 Docker 0.12 中已被修复。

为了缓解此类安全漏洞，需要将容器文件系统设置为只读。使用特权标志运行容器很有可能会导致此类安全攻击。将容器卷设置为只读是阻止容器突破的有效方法。

3.4. 镜像中毒

容器镜像有时可能被某些病毒感染。如果有人运行过时的、已知易受攻击的软件版本，也会出现镜像中毒问题。Docker 下载的镜像由系统验证[9]。此验证完全基于签名清单的存在。虽然 Docker 从未对清单中的下载镜像进行校验。因此攻击者可以将任一镜像与签名清单一起传输。这种安全问题可能导致许多严重的漏洞。在 Docker 中，镜像从 HTTPS 服务器下载，这些镜像会通过 Docker 守护程序中不安全的流处理管道：[decompress] -> [tarsum] -> [unpack]。在 Docker 内部，这条管道很有效，但是不受保护。因此，需要在确认其数字签名之前评估未经认证的输入流。

Docker 用户需要警惕用于下载镜像的代码。用户应该只下载经过身份验证的受信任镜像。管理此类安全问题的另一个更好的选择是在本地阻止 index.Docker.io。通过这种方式，用户可以在通过 Docker 加载导入 Docker 平台之前进行手动下载和验证镜像。

3.5. 安全威胁

安全威胁是容器技术的重大安全隐患之一。在攻击者窃取 API 密钥和数据库密码的情况下，整个系统可能都会受到影响。Docker 允许用户同时运行多个容器。在发生安全漏洞的情况下，整体服务和运营可能会中断。因此，需要做更多的工作来保证数据库密码和 API 密钥安全。同此类详细信息要求必须保密。

为了进一步保护 Docker 免受此类攻击，需要将容器文件系统设置为只读选项。对于共享机密，利用环境变量不是一个好选择。运行没有特权标志的容器也将为避免危及安全攻击提供极大帮助。

3.6. 中间人攻击

在这种攻击中，恶意者将自己插入两个合法方的通信之中。此类攻击者监视、更改或窃取在双方通信之间传输的有价值信息[10]。网络隔离是防止此类基于网络的攻击的最有效方法，此方法可使得它们无法操纵或窃听主机及其他容器的网络流量。在这种情况下，OpenVPN(开放虚拟专用网络)提供了一种通过 TLS(传输层安全协议)加密实现虚拟专用网络(VPN)的最佳方法。

Docker 提供了一种封装 OpenVPN 服务器的简便方法。通过这种方式，可以在 Docker 平台上更轻松地操作 OpenVPN 服务器进程和数据管理。Docker OpenVPN 的镜像是预先构建的。它包含了在均衡和持久的环境中运行服务器所需的一切条件。Docker 提供了那些使标准用例的自动化配置脚本，但是如果需要，它也提供完整的手动配置。

3.7. ARP 欺骗攻击

ARP (地址解析协议)欺骗是一种安全攻击，攻击者借助局域网(LAN)发送假造的 ARP 消息[11]。这时，恶意者能够将其 MAC 地址与网络上合法系统的 IP 地址相关联，当攻击者的 MAC 地址链接到合法的 IP 地址时，攻击者将开始从该特定的 IP 地址获取每一位数据。

最初，Docker 开发人员较少关注这一情况——即 ARP 用于将 IPv4 映射到以太网硬件(MAC)地址，虚拟网桥也可以利用这些地址将以太网帧分配到正确的容器。由于 ARP 数据包未被过滤，因此 ARP 本身没有可用的安全机制。因此，容器可以模仿其他容器甚至主机，从而出现了 ARP 欺骗。IPv6 中的 NDP (邻居发现协议)也通过类似的方式使用。

如果攻击者获得其中一个容器的访问权限，则通过破坏容器的安全性，攻击者可以获取、操纵或重定向网桥的任何信息。此信息可以是在容器和外部之间运行的任何流量。在这种情况下，攻击者可能会

捕获到 Web 应用程序和数据库容器之间发送的任何秘密细节(如密码)。此外，恶意者还能将恶意负载注入网络连接中。

保护 Docker 容器免受此类攻击的有效的方法是运行没有 NET_RAW 功能的容器。这样，容器内的程序将无法创建 PF_PACKET 套接字。如果没有 PF_PACKET 套接字，则无法执行 ARP 欺骗攻击。另一种更适合保护 Docker 容器免受此类攻击的方法是利用“ebtables”来过滤掉以太网帧。通过这种方式，可以捕获并检测到具有错误发送方协议或硬件地址(ARP 欺骗)的 ARP 数据包[12]。它还允许过滤掉不正确的源地址(MAC 欺骗)。在这种条件下，攻击者将没有机会实现 ARP 欺骗攻击。

4. 安全部署指南

采取预防性安全措施可以使我们避免漏洞的发生。Docker 容器安全性也是如此。通过对 Docker 容器安全问题采取多种预防手段，可以开发更安全可靠的应用程序。本节将概述 Docker 平台的一些安全部署指南。

4.1. Docker 镜像

如前所述，镜像中毒是 Docker 的安全问题之一。但是，从 Docker1.3 及以后的版本提供加密签名支持。通过这种方法，用户将能够发现官方存储库镜像的真实来源和完整性。此功能将减少中毒镜像的危险，并降低可能的安全威胁。强烈建议所有镜像都应从经过身份验证的下载源下载并支持加密签名。

4.2. 网络命名空间

在 TCP 端口上运行 Docker 可能会造成重大的安全隐患。这种方法允许任何人访问特定端口以获得对容器的访问。这导致在主机上获得 root 访问权限，或者可以访问 Docker 组。因此，严格要求确保使用 SSL 对通信进行充分加密，同时通过 TCP 提供对守护进程的访问。这种方法可以防止未经授权的各方与之交互。

为了进一步增强安全性管理，可以对 Docker0 实现内核防火墙 iptables 规则。例如，可以限制 Docker 容器的源 IP 范围[13]。这将阻止容器与外界通信。

4.3. 日志记录和审查

日志记录和审计为 Docker 安全管理提供了额外的安全保护。通过这种方式，用户可以监控流量以确保不会发生任何可疑活动。

4.4. SELinux/AppArmor

Docker 提供 Linux 内核安全模块，如 AppArmor 和 Security-Enhanced Linux (SELinux) [14]。可以通过访问控制安全策略配置这些 Linux 内核安全模块。凭借配置这些安全模块，用户可以实现强制访问控制以限制系统资源。

通过配置 SELinux，Docker 将通过权限检查策略获得额外的安全层。SELinux 通过标签管理所有内容。在 Docker 系统中，每个进程，文件/目录和系统对象都有一个标签。系统管理员通过使用这些标签编写规则来管理系统对象和进程之间的访问。

与 SELinux 类似，AppArmor 是另一种 Linux 安全增强模型。AppArmor 提供对各个程序的控制访问。通过此模型，管理员可以将安全配置文件加载到每个单独的程序中，以限制和管理程序的功能。

要加载 SELinux 或 AppArmor 安全策略正在使用的标签限制可以使用 Docker 中新添加的--security-opt

参数进行配置，如下所示：

--security-opt="label:user:USER": Set the label user for the container

--security-opt="label:role:ROLE": Set the label role for the container

--security-opt="label:type:TYPE": Set the label type for the container

--security-opt="label:level:LEVEL": Set the label level for the container

--security-opt="apparmor:PROFILE": Set the apparmor profile to be applied to the container

Example: docker run

--security-opt=label:level:s0:c100，c200 -i -t centos bash

4.5. 守护进程的特权

建议不要使用--privileged 命令，因为--privileged 命令将允许容器访问主机上的所有设备，并且它将为容器提供显式 LSM(即 AppArmor 或 SELinux)配置。LSM 配置将提供与主机进程类似的控制级别[15]。

通过避免使用--privileged 命令可以帮助减少安全风险并承担妥协。合法用户应该能够使用-u 选项启动守护程序。它可以减少容器内强制执行的权限。例如：

docker run -u -it <container_name> /bin/bash

4.6. Cgroups

Cgroups 提供了一种计算方法来对每个容器的资源进行限制。因此，Cgroups 通过限制系统资源耗尽提供了大量的功能来避免 DoS 攻击[16]：

CPU 使用率：docker run -it --rm --cpuset=0,1 -c 2 ...

内存使用率：docker run -it --rm -m 128m ...

存储使用率：docker -d --storage-opt dm.basesize=5G

4.7. SUID/GUID 二进制文件

为避免容器突破而造成的安全漏洞，应禁止 SUID 和 SGID 二进制文件。这可以通过减少特定命令行参数进而减少容器功能来实现。

另一种方法是使用 nosuid 属性挂载文件系统。通过应用此命令，用户可以避免 SUID 导致的缓冲区溢出而造成的安全问题：

docker run -it --rm --cap-drop SETUID --cap-drop SETGID ...

4.8. 设备组

设备隔离也是避免许多安全漏洞的关键方法之一[17]。默认情况下，容器包含所有权限。为避免此类问题，需要通过内置的“--device”选项安装设备，不要将“-v”与“--privileged”参数一起使用。

通过第三组选项可以为设备利用一组严格的权限;“rwm”分别覆盖 read，write 和 mknod 权限。例如，可以通过以下命令设置声卡只读权限：

docker run --device=/dev/snd:/dev/snd:r ...

4.9. 服务和应用

如果 Docker 容器安全性受到损害，若此时有许多敏感服务正在运行，这种情况可能会导致巨大的灾难。因此，为了避免这种情况，请考虑隔离敏感服务。通过在虚拟机中运行敏感服务(SSH 服务)，我们可以在系统中添加额外的安全层。还应避免使用不受信任的应用程序在容器内以 root 权限运行。

4.10. Linux 内核

有时，过时的内核更容易出现安全漏洞。因此，内核与系统提供的更新实用程序(例如 apt-get，yum 等)为最新版本是非常重要的。通过将内核与 GRSEC 或 PAX 结合使用，可以有效地防范内核损坏错误。

4.11. 用户命名空间

目前，Docker 不直接支持用户命名空间。但是，Docker 的容器可以通过应用克隆系统调用或使用“取消共享”功能，在支持的内核上使用它们。通过 LXC 驱动程序支持 UID 映射，但是在本机 libcontainer 库中不被支持。

用户命名空间允许 Docker 守护程序作为主机上的非特权用户执行[18]。但是，此 Docker 守护程序将显示为像容器内一样执行。

4.12. libseccomp(和 seccom -bpf 扩展名)

系统调用过程对系统操作并不重要。应对其进行限制，以避免在受损容器内滥用。要限制 Linux 内核的系统调用过程，请使用 libseccomp 库。此功能目前尚未开发。此功能在 LXC 驱动程序中可用，但在 libcontainer 中不可用。

可以使用以下命令重新启动 Docker 守护程序以使用 LXC 驱动程序：

docker -d -e lxc

4.13. 完全虚拟化

从容器升级到主机可能非常危险。如果在 Docker 镜像中利用了内核漏洞，就会发生这种情况。要防止此类风险，请使用包含 Docker 的完整虚拟化解决方案，例如 KVM [19]。Docker 提供了嵌套 Docker 镜像以提供 KVM 虚拟化层的功能。

4.14. 安全审计

安全审计是保护系统免受重大安全风险的关键方法之一。应定期审核主机系统和容器，以评估和识别可能出现的漏洞和错误配置。这些漏洞和错误配置可能会对我们的系统造成危险，并可能危及资源[20]。

4.15. 多租户环境

容器应该在专用主机上运行。这对于集装箱的安全非常重要。当用户处理一些敏感操作时，它变得更加重要。

建议使用 Docker 容器内核的共享特性。因为 Docker 容器内核的多租户环境可以提供安全的职责分离。因此，强烈建议容器应该在专用主机上运行[21]。

通过将集装箱之间通信减少到非常低的水平，也可以实现更安全的环境。可以通过将 Docker 守护进程设置为使用--icc = false 来实现。另外，在需要时指定-link 与 Docker 一起运行也很有用。

4.16. Docker 内容信任

内容信任是一项新功能，可以为容器提供额外的安全层。Docker1.8.0 版中添加了此功能。它将允许 Docker 用户在下载 Docker 镜像之前符合容器镜像的合法性(在任何公共 Docker Hub 上都可用)。

此功能基本思想是保护 Docker 平台，并向用户保证他们不会在其基础技术架构上部署任何可能有害的东西。

5. 总结

当前，Docker 在云计算领域中发挥着巨大作用，其在可移植性、运行速度、资源消耗方面较传统的虚拟化技术有着绝对的优势，但追根到底，其本质上是对传统虚拟机技术在某些方面的缺陷上进行了优化。Docker 已逐步取代传统虚拟化技术在云计算平台的领导地位，成为云计算中不可或缺的关键技术。Docker 提供了一种轻量级的高效方法来打包应用程序及其所有依赖项。但是，一些安全问题阻碍了它的广泛使用。本文已经概述了当前一些存在的安全问题，并提供了解决这些问题的方法。同时本文还概述了一些安全部署策略，可以以更安全的方式在 Docker 上部署应用程序。这些指导和预防措施可为未来的应用开发提供更安全可靠的容器平台。虽然与 VM 等传统传统虚拟化技术相比，Docker 还存在着资源浪费、安全性风险等问题，但是随着技术和相关流程的日趋成熟，这些问题都将被逐一攻破，届时，Docker 将能更好的发挥其在云计算领域中的优势。

参考文献

[1] 刘景云. 浅析 Docker 虚拟化技术[J]. 网络安全和信息化, 2019(1): 73-81.

[2] 李娜. Docker 容器技术的发展及应用研究[J]. 数字技术与应用, 2018, 36(11): 95-96.

[3] Shu, R., Gu, X. and Enck, W. (2017) A Study of Security Vulnerabilities on Docker Hub. *Proceedings of the Seventh ACM on Conference on Data and Application Security and Privacy*, ACM, Scottsdale, 269-280. https://doi.org/10.1145/3029806.3029832

[4] Anderson, C. (2015) Docker [Software Engineering]. *IEEE Software*, **32**, 102-c3. https://doi.org/10.1109/MS.2015.62

[5] Reshetova, E., Karhunen, J., Nyman, T. and Asokan, N. (2014) Security of OS-Level Virtualization Technologies. *Nordic Conference on Secure IT Systems*, Tromsø, 15-17 October 2014, 77-93. https://doi.org/10.1007/978-3-319-11599-3_5

[6] Guo, Y., Rao, J., Cheng, D., *et al*. (2017) iShuffle: Improving Hadoop Performance with Shuffle-on-Write. *IEEE Transactions on Parallel & Distributed Systems*, **28**, 1649-1662.

[7] Li, S.-H., Yen, D.C., Chen, S.-C., Chen, P.S., Lu, W.-H. and Cho, C.-C. (2015) Effects of Virtualization on Information Security. *Computer Standards & Interfaces*, **42**, 1-8. https://doi.org/10.1016/j.csi.2015.03.001

[8] 鲁涛, 陈杰, 史军. Docker 安全性研究[J]. 计算机技术与发展, 2018, 28(6): 115-120.

[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. and Pietro, R.D. (2016) To Docker or Not to Docker: A Security Perspective. *IEEE Cloud Computing*, **3**, 54-62. https://doi.org/10.1109/MCC.2016.100

[11] 肖微. ARP 欺骗在网络中的应用及防范[J]. 通讯世界, 2017(5): 86-87.

[12] 张遥, 王森林. Docker 安全性研究[J]. 网络安全技术与应用, 2017(8): 32-33.

[13] 莘建浦. 基于 Docker 容器的网络安全实训平台的研究与实现[D]: [硕士学位论文]. 北京: 邮电大学, 2018.

[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] 张涛. Docker 全攻略[M]. 北京: 电子工业出版社, 2016.

[16] 任为. 对基于 Docker 的虚拟化技术的几点探讨[J]. 电子制作, 2018(14): 42-43+52.

[17] 郭甲戌, 胡晓勤. 基于 Docker 的虚拟化技术研究[J]. 网络安全技术与应用, 2017(10): 28-29.

[18] 蔡志强. 基于 Docker 技术的容器隔离性分析[J]. 电子世界, 2017(17): 195.

[19] 李明, 郭洋, 蒋明. 基于 Docker 的虚拟化技术研究[J]. 中国新通信, 2017, 19(9): 73-74.

[20] 吴芦峰. 容器级虚拟化的安全审计与监控研究[D]: [硕士学位论文]. 北京: 北京邮电大学, 2018.

[21] Turnbull, J. (2014) Docker Container Breakout Proof-of-Concept Exploit.

知网检索的两种方式：

1. 打开知网页面 http://kns.cnki.net/kns/brief/result.aspx?dbPrefix=WWJD
   下拉列表框选择：[ISSN]，输入期刊 ISSN：2161-8801，即可查询
2. 打开知网首页 http://cnki.net/
   左侧“国际文献总库”进入，输入文章标题，即可查询

投稿请点击：http://www.hanspub.org/Submission.aspx
期刊邮箱：csa@hanspub.org