Docker 容器逃逸的防御方法研究

陈俊杰，陈　奋，陈荣有
厦门服云信息科技有限公司，福建 厦门 361001

摘　要： Docker 作为当下最主流的容器引擎，基于易打包、分发、部署以及快速、轻量的操作系统级虚拟化等特点，被广泛应用于云原生虚拟化环境。然而，由于 Docker 与宿主机共享内核，具有容易受到容器逃逸的安全风险。攻击者可利用容器逃逸攻击影响到承载容器的底层基础设施的保密性、完整性和可用性。首先对 Docker 容器逃逸技术进行概述，介绍其含义、根源以及 ATT&CK 攻防矩阵。接着对 Docker 容器逃逸的国内外研究现状进行介绍，并提出一套基于容器生命周期的防御方案。该方案具有较好的完备性，可帮助用户用低成本、高效率的方式做好容器逃逸的防御工作。

关键词： Docker；容器逃逸；ATT&CK；防御

Research on Defense Mechanisms of Docker Container Escape

CHEN Junjie, CHEN Fen, CHEN Rongyou
Xiamen Fortcloud Information Technology Co., Ltd., Xiamen, Fujian 361001, China

Abstract: As the most mainstream container engine, Docker is widely used in cloud-native virtualization environments based on its features such as easy-packaging, distribution and deployment, and fast and lightweight operating system-level virtualization. However, Docker might be influenced easily by container escape since it shares the same kernel with host machine. Attackers can use container escape attacks to affect the confidentiality, integrity and availability of the underlying infrastructure that hosts the container. Firstly, this paper gives an overview of Docker container escape technology, and introduces to its definition, cause, and ATT&CK attack and defense matrix. Next, the current research status of Docker container escape at home and abroad is introduced, and a defense scheme based on the container life cycle is proposed, which can support users to defense container escape with lower cost and higher effectiveness.

Key words: Docker; container escape; ATT&CK; defense

1　引言

容器技术是云原生技术的代表之一，目前正处于快速发展的阶段。在当下的容器引擎中，Docker 居于行业“领头羊”的地位。该技术在带来效益的同时，也带来一系列的安全问题，其中最严峻且最具代表性的问题当属容器逃逸问题。攻击者可利用 Docker 容器与宿主机在内核层面的隔离性不足的特点，“逃逸”出其容器自身所拥有的权限，影响

作者简介：陈俊杰（1991—），男，安全研究工程师，硕士，主要研究方向为容器安全、Java Web 安全，E-mail: chenjj2@safedog.cn；陈奋（1981—），男，CEO，硕士，主要研究方向为云安全、大数据安全；陈荣有（1980—），男，研发副总，主要研究方向为云安全、主机安全、容器安全、网络安全攻防技术。

到宿主机和宿主机上其他容器的保密性、完整性和可用性。因而对 Docker 容器逃逸进行研究具有重要意义。然而在当前的相关研究中，业界对容器逃逸技战术、研究现状以及防御技术的认知仍不够全面，本文主要对容器逃逸问题进行原理分析，通过梳理国内外对容器逃逸防御的工作，提出一种基于 Docker 容器生命周期的容器逃逸综合防御实践方案，希望能对业界有所补益。

2 Docker 容器逃逸概述

2.1 容器逃逸的含义

“容器逃逸”是容器安全风险中最具代表性的高风险安全问题。业界对容器逃逸的定义尚未形成统一定论。文献[1]指出容器逃逸是一种过程和结果：首先，攻击者通过劫持容器化业务逻辑，或直接控制（CaaS 等合法获得容器控制权的场景）等方式，已经获得了容器内某种权限下的命令执行能力；攻击者利用这种命令执行能力，借助一些手段进一步获得该容器所在直接宿主机上某种权限下的命令执行能力。文献[2]认为容器逃逸的结果判断标准可包括敏感信息泄露（例如获得宿主机文件或内存的读写能力）。文献[3]则指出容器逃逸的重点是攻击和绕过将容器和主机以及其他容器分离的隔离和保护机制。通过对这些定义进行分析可以得出，容器逃逸攻击的实施并非一蹴而就，往往是一系列以“权限提升”为目的的攻击步骤的组合，并且达到容器逃逸目的的可能途径也是多样化的。

2.2 Docker 容器逃逸的根源

Docker 容器逃逸问题的根源在于 Docker 容器与宿主机共用内核，并且在内核层面的隔离性不足。这使得攻击者可通过利用漏洞“逃逸”出自身拥有的权限，实现对宿主机或宿主机上其他容器的访问，如图 1 所示。

在共用内核这个前提下，容器主要通过内核的 Cgroup 以及 Namespace 这两大特性来达到资源限制和容器隔离的目的。目前 Cgroup 对系统资源的限制已比较完善，但是对 Namespace 的隔离还是不够，只有 PID、Mount、Network、UTS、IPC 和 User 这几种。由于共享内核导致的固有缺陷，Namespace 的隔离问题难以完善，并且未来 Linux 内核社区也不会对此做太多的改进[4]。

2.3 Docker 容器逃逸技战术分析

近 3 年来，业界对 Docker 容器逃逸问题的研究十分活跃。其中阿里云提出的“云上容器 ATT&CK 攻防矩阵”可帮助开发和运维人员了解容器的安全风险和落地安全实践[5]。该攻防矩阵如图 2 所示。

由图 2 可知，攻击者想对单个 Docker 容器实施容器逃逸攻击的话，可能从初始访问、执行、持久化与权限提升等方面逐步入手。在实战过程中，该攻防矩阵里的各类攻击技战术出现的频率并不均等。而“权限提升”是攻击者实施容器逃逸的关键且高频的技战术。通过分析业界现有的容器逃逸安全研究文章可发现，该矩阵中的“权限提升技战术”已较好地覆盖了业界已知的攻击利用方式，如表 1 所示。

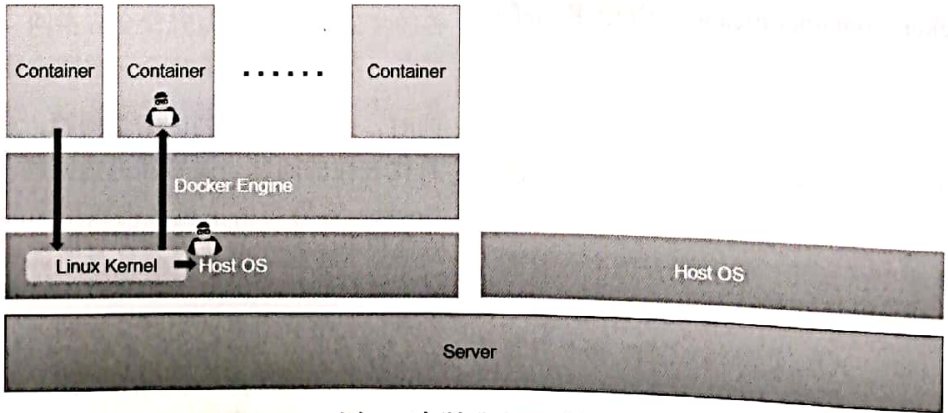

图 1 容器逃逸示意图

Initial Access 初始访问	Execution 执行	Persistence 持久化	Privilege Escalation 权限提升	Defense Evasion 防御逃逸	Credential Access 窃取凭证	Discovery 探测	Lateral Movement 横向移动	Impact 影响
云账号 AK 泄露	通过 kubectl 进入容器	部署远控容器	利用特权容器逃逸	容器及宿主机日志清理	K8S Secret 泄露	访问 K8S API Server	窃取凭证攻击云服务	破坏系统及数据
使用恶意镜像	创建后门容器	通过挂载目录向宿主机写文件	K8S Rolebinding 添加用户权限	K8S Audit 日志清理	云产品 AK 泄露	访问 Kubelet API	窃取凭证攻击其他应用	劫持资源
K8S API Server 未授权访问	通过 K8S 控制器部署后门容器	K8S cronjob 持久化	利用挂载目录逃逸	利用系统 Pod 伪装	K8S Service Account 凭据泄露	Cluster 内网扫描	通过 Service Account 访问 K8S API	Dos
K8S configfile 泄露	利用 Service Account 链接 API Server 执行指令	在私有镜像库的镜像中植入后门	通过 Linux 内核漏洞逃逸	通过代理或匿名网络访问 K8S API Server	应用层 API 凭据泄露	访问 K8S Dashboard 所在 Pod	Cluster 内网渗透	加密勒索
docker daemon 公网暴露	带有 SSH 服务的容器	修改核心组件访问权限	通过 Docker 漏洞逃逸	清理安全产品 Agent	利用 K8S 准入控制器窃取信息	访问私有镜像库	通过挂载目录逃逸到宿主机	
容器内应用漏洞入侵	通过云厂商 CloudShell 下发指令		通过 K8S 漏洞进行提权	创建影子 API Server		访问云厂商服务接口	访问 K8S Dashboard	
Master 节点 SSH 登录凭据泄露			容器内访问 docker.sock 逃逸	创建超长 annotations 使 K8S Audit 日志解析失败		通过 NodePort 访问 Service	攻击第三方 K8S 插件	
私有镜像库泄露			利用 Linux Capabilities 逃逸					

图 2 Docker & K8S 运行时威胁检测 Attack Matrix

表 1 对 Docker 容器逃逸的“权限提升技战术”的补充说明

技战术	说明
利用特权容器逃逸	Docker 允许特权容器访问宿主机上的所有设备，同时修改 AppArmor 或 SELinux 的配置
K8S Rolebinding 添加用户权限	K8S 通过 RBAC 做操作鉴权，允许管理员通过 Kubernetes API 动态配置策略。某些情况下运维人员为操作便利，会对普通用户授予 cluster-admin 的角色。攻击者收集到该用户登录凭证后，可直接以最高权限接管 K8S 集群。少数情况下，攻击者可以先获取角色绑定(RoleBinding)权限，并通过将其他用户添加为 cluster-admin 或其他高权限角色来完成提权
利用挂载目录逃逸	如果容器或 Pod 在启动时将宿主机的核心目录以写权限挂载(如/root, /proc, /etc 等)，则攻击者进入容器后可修改敏感文件并逃逸，攻击者可能利用的技战术包括但不仅限于： （1）修改/root/.ssh/authorized_keys 获取 SSH 登录权限 （2）读取其他进程空间内容 （3）利用/etc/crontab 执行定时任务
利用 Linux 内核漏洞逃逸	宿主机的内核漏洞会一并影响 Docker 的安全。攻击者可能利用的 Linux 内核漏洞包括但不仅限于： （1）CVE-2016-5195（“脏牛”漏洞） （2）CVE-2018-18955（Linux user namespace 内核提权漏洞）
利用 Docker 漏洞逃逸	Docker 自身软件安全性可能成为容器逃逸攻击的一环。攻击者可能利用的 Docker 漏洞包括但不仅限于： （1）CVE-2019-5736（Docker runC 容器逃逸漏洞） （2）CVE-2019-14271（Docker cp 容器逃逸漏洞） （3）CVE-2019-15752（Docker 权限许可和访问控制问题漏洞） （4）Shocker 攻击 （5）文件暴力破解
利用 K8S 漏洞进行提权	容器编排平台的软件漏洞可能成为容器逃逸攻击的一环。攻击者可能利用的 K8S 漏洞包括但不仅限于：CVE-2018-1002105（Kubernetes 特权提升漏洞）
容器内访问 docker.sock 逃逸	当 docker.sock 被挂载到容器内部时，攻击者可以在容器内部访问该 socket 并管理 Docker daemon。攻击者可能使用的方式包括但不仅限于：控制 Docker deamon，并创建特权的恶意容器，从而拿到 root shell
利用 Linux Capabilities 逃逸	容器在设计上使用 Cgroup、Namespace 等对宿主机的资源进行隔离及调度使用，同时也支持使用 Linux Capabilities 做细粒度的权限管控。不安全的权限分配可为容器逃逸提供通路

通过上述分析可推断出，在 Docker 容器运行时的环境中，容器运行环境的基础设施漏洞、用户的不安全配置以及 Docker 容器里的 Web 应用漏洞均可能成为容器逃逸漏洞的突破口。

值得重视的是，Docker 镜像的安全问题也能成为触发容器逃逸攻击的一环。例如，有的攻击者会在镜像中植入后门程序、病毒、木马、挖矿程序等，诱导用户使用。有的 Docker 镜像也存在着安全配置不合规问题（例如 CVE-2019-5021 这一 Alpine Docker 镜像漏洞）。

3 Docker 容器逃逸防御国内外研究现状

通过对国内外的研究成果进行分析，可发现为防御 Docker 容器逃逸问题，业界采用的方案包括：运行时异常检测、安全容器、基于 Docker 容器生命周期的安全运维，但这些方案都存在着一定的局限性。

3.1 运行时异常检测

通过对国内外一些主流 Docker 容器安全产品进行功能比对，可以发现多数容器安全产品均实现了基于异常行为的容器逃逸风险实时监测的功能，但存在以下不足：

（1）检测覆盖面不足（Docker 容器逃逸的技战术较多，要实现全面检测较为困难）；

（2）检测能力时效性不足（例如，当新的容器逃逸漏洞利用方式被公开后，对 0day 漏洞攻击的抵御能力不足；有些容器安全产品所支持的“shocker 攻击”与“文件暴力破解”的检测功能实际意义不大，因为受漏洞影响的 Docker 引擎版本不再是主流）。

为了对容器运行时的异常进行有效检测，安全人员进行了一些很有价值的研究，例如，Sysdig Falco 团队运用 eBPF 实现对 Linux 内核层系统的审核，可连续监视和检测容器、应用程序、主机和网络活动。eBPF 技术效果卓著，但对 Linux 的内核版本有着限制[6]。有研究人员提出了一种基于进程所属 Namespace 状态监测的防御方案，用之检测 Docker 容器逃逸攻击[7]，该方案具有发现 0day 漏洞的能力，但存在着检测滞后与性能开销较大等问题。

3.2 安全容器

业界有研究团队为了在轻量化和安全性之间达到较好的平衡，在“传统容器”的基础上，为“容器-主机”或“容器-容器”之间提供额外的保护，落地了“安全容器”。典型的“安全容器”创新技术包括了 Kata、gVisor、Firecracker[8]。但它们仍存在以下不足：

（1）具有安全风险（仍面临着已知或未知的容器逃逸攻击的风险）。

（2）引入现阶段仍不够主流的技术栈。

3.3 基于 Docker 容器生命周期的安全运维

业界有研究团队在文献[9]指出，在对 Docker runC（CVE-2019-5736）等漏洞做容器逃逸检测时，其安全产品支持为用户提供以下帮助：在“攻击前”做好安全基线工作（例如，限制从非信任的仓库拉取镜像、审查 SELinux 设置项、对以 UID 0 运行的容器做告警），在“攻击时”做好“异常行为检测”（例如软件安装、文件修改与文件执行）工作。在 Docker 容器生命周期内的多个阶段进行防控可取得良好的效果，但该文献介绍的实践方式在完备性和性能方面仍需改善。

4 Docker 容器逃逸防御技术实践

4.1 实践思路

为了低成本、高效率地做好容器逃逸的防御工作，应在 Docker 容器的全生命周期内开展落实最小权限、纵深防御的工作。在具体实施起来，可“因地制宜”，按环境开展防御工作，如表 2 所示。其

表 2 按生产环境对 Docker 容器逃逸进行防御与检测

环境	工作重点
非生产环境	镜像安全（镜像扫描、镜像加固）
生产环境	（1）资产清点 （2）镜像运行控制★（容器准入） （3）容器安全★（如异常行为分析） （4）网络安全（如基于 K8S 自带的网络策略做防火墙） （5）应用 K8S 的集群安全机制★（如 API Server 的认证与授权、Secret 秘钥凭据） （6）基线合规 （7）日志审计

中，在生产环境内的镜像运行控制、容器安全以及应用K8S集群的安全机制[10]是工作的重点。

在实施时亦可“因时制宜”，从攻击前、攻击时和攻击后这三个阶段分别开展防御与检测工作。这三个阶段的工作重点如表3所示。

表3 按阶段对Docker容器逃逸进行防御与检测

攻击前	攻击时	攻击后
资产清点	镜像运行控制	
镜像安全	容器安全	
基线合规	网络安全	日志审计
资产加固	运用K8S集群的安全机制	

4.2 实践案例介绍

下面通过介绍Docker runC（CVE-2019-5736）容器逃逸漏洞的攻防研究经验对表2、表3做补充说明。依据安全实施准则对攻击前、中、后这三个阶段开展检测与防御工作，如图3所示。

4.2.1 “攻击前”阶段工作

在检测方面应注重的工作包括资产清点和镜像安全工作。资产清点的核心工作是资产版本比对。资产版本比对工作有助于防御工作的开展。在具体实践时，若为了判别该runC容器逃逸漏洞是否会在当前的生产环境被利用，可先进行“资产版本比对”，若资产的版本不受漏洞影响，则无须在“攻击时阶段”对“修改二进制文件，以指向/proc/self/exe文件”等异常行为进行监控。这样的处理方式有助于降低性能开销。镜像安全的核心工作是镜像扫描与镜像加固，提防攻击者别有用心地制作可触发runC容器逃逸漏洞的恶意镜像并诱导用户使用。

在防御方面应注重的工作包括基线合规工作和资产加固工作。基线合规工作有助于收敛暴露给外部的攻击面。在具体实践时，可参照Docker与K8S的CIS Benchmarks等安全基线检查工具进行检查与加固。“资产版本升级”工作可帮助用户规避受漏洞影响的风险，在具体实践时，可在不影响用户以及业务正常运行的前提下，进行安全更新工作，确保所使用的资产不受已知漏洞的影响。

4.2.2 “攻击时”阶段工作

在检测方面应关注的检测项可包括宿主机上与容器内的异常行为监控（文件、网络、进程等方面）以及容器准入工作的合规性。在具体实践时，可运

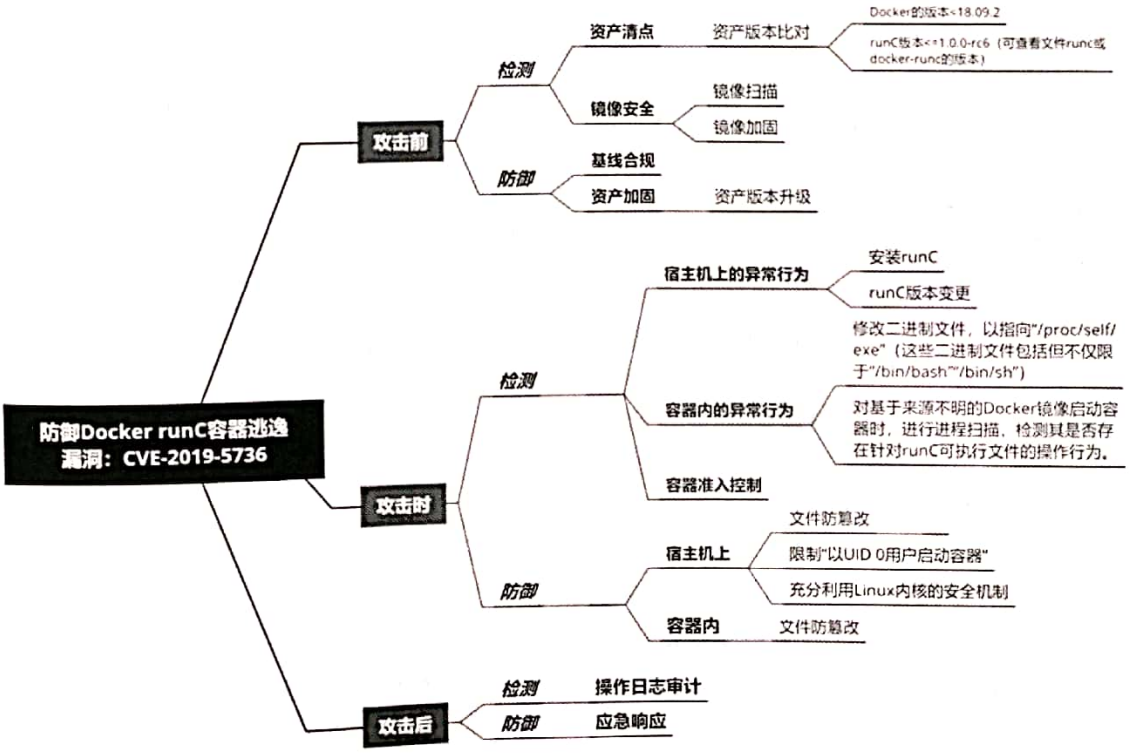

图3 对Docker runC容器逃逸漏洞进行防御与检测

用现阶段较为成熟的主机入侵检测技术经验，并选择合适的算法对宿主机与容器进行异常行为监控。在宿主机上监控的异常行为可包括安装 runC、变更 runC 版本等；在容器内监控的异常行为可包括对“/proc/self/exe”文件的修改（Docker 容器、微服务的功能一般是较为固化的，若发现越界的行为，应予以警惕），应对容器准入操作做监控（检测容器或 Pod 的配置是否符合安全策略，若不合规，应阻止其进入生产环境）。

在防御方面应注重的工作包括：在宿主机上以及容器内做文件权限控制以及在 Docker 宿主机充分利用 Linux 内核的安全能力。在具体实践时，可对敏感文件做“防篡改”限制，对 Linux 内核的安全机制进行充分运用（可做资源隔离的 Namespace、可用黑白名单限定进程调用的 Seccomp、可限定容器能力的 Capability、可限定应用程序访问控制权限的 Apparmor 均是可选用的安全技术）。

4.2.3 “攻击后”阶段工作

在检测方面应注重操作日志审计。在具体实践时，可通过日志收集容器将业务容器的日志统一发送到大数据平台，并进行分析、告警[11]。

在防御方面应注重应急响应。在具体实践时，若发现生产环境受到容器逃逸漏洞的攻击，应及时进行应急响应，提防危害扩大化。

5 结束语

本文的主要工作是对 Docker 容器逃逸的攻防技术做剖析，并立足于攻击前、中、后这三个阶段提出综合防御思路，希望能够为用户防御容器逃逸漏洞提供有益的参考。

贯彻落实最小权限、纵深防御的网络安全原则对防御 Docker 容器逃逸攻击具有重要意义。为做好 Docker 容器逃逸的防御工作，只展开运行时异常检测的工作是不充分的，须在做好镜像安全、基线合规、日志审计等工作的前提下，做好镜像运行控制、容器安全（充分运用 Linux 内核的安全机制）、应用 K8S 集群的安全机制等方面的工作。

在接下来的研究工作中，可对云上容器 ATT&CK 攻防矩阵中作用于容器逃逸漏洞的高频攻击技术进行研究分析，围绕采集、可视化、关联分析和响应的全流程建设开展工作。

参考文献：

[1] 绿盟星云实验室. [云原生攻防研究]容器逃逸技术概览[EB/OL]. [2020-10-22]. https://mp.weixin.qq.com/s/_GwGS0cVRmuWEetwMesauQ.

[2] VRANCKEN J. A methodology for penetration testing docker systems[D].Amsterdam: University of Amsterdam, 2020.

[3] JANSSEN R. Categorizing container escape methodologies in multi-tenant environments[D]. Amsterdam: University of Amsterdam, 2018.

[4] 华为 Docker 实践小组. Docker 进阶与实战[M]. 北京: 机械工业出版社, 2016: 132-133.

[5] 阿里云安全团队. 国内首个云上容器 ATT&CK 攻防矩阵发布，阿里云助力企业容器化安全落地[EB/OL]. [2020-10-24]. https://developer.aliyun.com/article/765449?groupCode=aliyunsecurity.

[6] SALAMERO J. Kubernetes runtime security with Falco and Sysdig[EB/OL]. [2020-10-29]. https://www.cncf.io/wp-content/uploads/2020/08/Kubernetes-Runtime-Security- with-Falco-and-Sysdig.pdf.

[7] 简智强. Docker 容器安全监控系统设计与实现[D]. 重庆: 重庆邮电大学, 2017.

[8] 睿智. 云原生之容器安全实践[EB/OL]. [2020-03-12]. https://tech.meituan.com/2020/03/12/cloud-native-security.html.

[9] BENSON H. The runC vulnerability-a deep dive on protecting yourself[EB/OL]. [2019-02-21]. https://www.stackrox. com/post/2019/02/the-runc-vulnerability-a-deep-dive-on-protecting-yourself/.

[10] 龚正. Kubernetes 权威指南:从 Docker 到 Kubernetes 实践全接触[M]. 北京: 电子工业出版社, 2019: 358-406.

[11] OPPO 互联网技术. 数万台服务器下的 Docker 深度安全实践[EB/OL]. [2019-12-11]. https://segmentfault.com/a/1190000021255340.