基于 LSM 的容器多级安全隔离系统设计与优化

熊明俊[1]，郭培馨[2]
（1. 金航数码科技有限责任公司，北京 100028；
2. 军工保密资格审查认证中心，北京 100000）

摘 要：随着容器技术在生产环境中的广泛应用，其安全隔离机制的不足逐渐显现。提出一种基于 Linux 安全模块框架（Linux Security Modules，LSM）的容器多级安全隔离模型，通过定义容器化的主体与客体，引入安全标签对容器进程和文件资源进行分类标记，构建细粒度的强制访问控制策略。在此基础上，开发了 DLSM（Docker LSM）原型系统，从容器进程隔离和共享数据卷隔离两个层面强化防护：限制容器对宿主机关键目录及其他容器文件系统的访问路径，防范越权与逃逸风险；对共享数据卷实施分组管理与权限分级，避免数据泄露或篡改。与 SELinux、AppArmor 等方案的对比实验表明，DLSM 在拦截容器逃逸、敏感目录挂载及跨容器通信等攻击场景中均实现 100% 拦截成功率，且各项性能开销控制在 5% 以内，是一种兼顾安全与效率的容器隔离方案。

关键词：容器隔离；多级安全；强制访问控制；LSM；Docker
中图分类号：TP309　　**文献标志码**：A　　**DOI**：10.19358/j.issn.2097-1788.2026.05.002

中文引用格式：熊明俊，郭培馨. 基于 LSM 的容器多级安全隔离系统设计与优化[J]. 网络安全与数据治理，2026，45(5)：11-17.
英文引用格式：Xiong Mingjun，Guo Peixin. Design and optimization of multi-level security isolation system for containers based on LSM[J]. Cyber Security and Data Governance，2026，45(5)：11-17.

Design and optimization of multi-level security isolation system for containers based on LSM

Xiong Mingjun[1], Guo Peixin[2]
(1. AVIC Digital Corporation Ltd., Beijing 100028, China;
2. Defence Industry Secrecy Examination and Certification Center, Beijing 100000, China)

Abstract: With the widespread adoption of container technology in production environments, deficiencies in security isolation mechanisms have become increasingly apparent. This paper proposes a multi-level security isolation model for containers based on the Linux Security Modules (LSM) framework. The model constructs fine-grained Mandatory Access Control (MAC) policies by defining containerized subjects and objects and introducing security labels to classify container processes and file resources. A Docker LSM (DLSM) prototype system is developed to strengthen protection at two levels: restricting container access to host critical directories and other container file systems to prevent privilege escalation and escape, and implementing grouped management with permission grading for shared data volumes to prevent data leakage or tampering. Comparative experiments with SELinux and AppArmor demonstrate that DLSM achieves a 100% interception rate in across container escape, sensitive directory mounting, and cross-container communication attack scenarios, with performance overhead kept below 5%, providing an effective solution balancing container security and efficiency.

Key words: container isolation; multi-level security; mandatory access control; LSM; Docker

0 引言

容器技术因启动快速、资源利用率高和部署灵活等优势，在云计算与制造业信息系统中得到广泛应用。然而，容器共享宿主机操作系统内核，主要依靠命名空间（Namespace）和控制组（cgroups）实现隔离[1]，在隔离强度上存在固有缺陷。一旦隔离机制被突破，恶意容器可对宿主机或其他容器发起攻击，导致数据泄露甚至业务中断[2]。近年来，随着多起容器逃逸漏

洞事件（如 CVE-2024-21626“Leaky Vessels”、CVE-2024-23651 等[3]）的曝光，容器安全隔离的紧迫性进一步凸显。

现有容器安全方案主要包括基于强制访问控制（Mandatory Access Control，MAC）的 SELinux[4] 和 AppArmor[5]，以及基于内核模拟的 gVisor 等沙箱方案。SELinux 策略编写复杂，在容器场景下配置与运维门槛较高[6]；AppArmor 配置相对简单，但细粒度隔离能力不足；gVisor 通过拦截系统调用实现强隔离，但性能损耗达 15% ~50%，不适用于高并发场景。近年来，扩展伯克利包过滤器（extended Berkeley Packet Filter，eBPF）技术被应用于容器运行时安全监控[7]，但 eBPF 被发现在云环境中存在跨容器攻击风险[8]。此外，Falco 等工具实现了基于系统调用的异常检测；无训练的容器异常检测和系统调用依赖图的逃逸检测也取得进展[9]；Kata Containers 等轻量级虚拟化方案从架构层面增强隔离；容器安全威胁建模为系统化防御提供了理论基础。上述方案虽然各有优势，但在共享数据卷的细粒度访问控制方面仍存在不足，难以满足多租户环境下“安全隔离与高效共享”的双重需求。

此外，容器自身安全机制存在多个薄弱环节[10]。例如：/proc、/sys 等系统关键目录在容器间共享[11]，命名空间配置缺陷导致越权[12]，共享数据卷缺乏差异化访问控制[13]等。Martinez Delbugio 等[14]提出的增强型 LSM 方案虽改善了内存安全性，但未涉及多级安全隔离。

针对上述问题，本文提出基于 LSM 框架的多级安全隔离系统 DLSM。该系统通过安全标签实现容器进程的强制隔离，通过分组与安全等级机制实现共享数据卷的细粒度访问控制，兼顾安全性与性能效率。DLSM 的核心创新在于：

（1）标签动态绑定机制，支持运行时策略热更新；

（2）轻量化策略执行引擎，性能开销控制在 5% 以内；

（3）数据卷细粒度访问控制，实现“同组共享、跨组隔离、高等级可读低等级”的多级安全模式。

1 基于 LSM 的多级安全隔离机制设计与实现

1.1 DLSM 设计思路与方案对比

DLSM 的核心设计思想是在内核态引入强制访问控制策略，从而突破自主访问控制（Discretionary Access Control，DAC）模型的局限，实现对容器进程与目标资源的全局强制约束。DLSM 采用多级安全（Multi-Level Security，MLS）思想，将容器划分为不同区域和不同安全等级，实现“同级互通、跨级受限”的访问控制模式，并将进程隔离与共享数据卷隔离相结合，构建可扩展、兼容性强的安全隔离体系。

在现有的容器安全方案中，SELinux 和 AppArmor 同样基于 LSM 框架实现强制访问控制，但两者在容器场景下均存在明显的局限性。SELinux 采用 Type Enforcement 机制，以“user:role:type:level”四元安全上下文作为访问判定基础，其策略体系面向通用操作系统而非容器环境，缺乏对容器 ID、容器分组等容器语义的原生感知。当容器数量增加时，管理员需为每个容器及其文件资源手动编写类型定义与转换规则，这会导致策略文件规模急剧膨胀，且容易因配置错误引发安全策略失效或容器运行异常[6]。此外，SELinux 的安全上下文在进程创建时绑定，运行期间通常不可变，无法适应容器动态创建、销毁和迁移的场景需求。AppArmor 采用基于路径名（pathname-based）的访问控制方式，通过配置文件限制进程对特定文件路径的访问。该机制在容器场景下存在两个关键缺陷：一是路径名匹配无法区分不同容器访问同一挂载路径（如共享数据卷）时的权限差异，导致同一路径对所有容器要么全部允许、要么全部拒绝；二是 AppArmor 不具备多级安全能力，无法实现基于安全等级的分级访问控制，在多租户共享数据卷场景下难以满足“高等级可读低等级、低等级不可读高等级”的安全需求。

针对上述不足，DLSM 在以下三个方面实现了技术突破：（1）容器感知的三元组标签机制：DLSM 设计了 {container_id，group_id，level_id} 三元组安全标签，将容器身份、分组归属和安全等级直接编码到文件的扩展属性中，使安全策略与容器语义深度绑定，避免了 SELinux 通用安全上下文在容器场景下的语义鸿沟；（2）标签动态绑定与策略热更新：与 SELinux 启动时加载、运行期固化的策略模式不同，DLSM 支持运行时动态调整容器的安全标签和分组策略，策略更新后通过内核态缓存同步机制即时生效，可适应容器快速创建和弹性伸缩的运维需求；（3）数据卷细粒度分级访问控制：DLSM 在 LSM 框架内原生实现了共享数据卷的分组隔离和多级安全访问控制，填补了 SELinux 和 AppArmor 在该领域的功能空白。

与传统 MAC 模型相比，DLSM 具有明显优势。传统的强制访问控制模型（如 Bell-LaPadula 模型）建立在“主体 - 客体”二元关系之上，主体通常为用户或进程，客体为文件或资源，安全标签（如密级）在系

统初始化时静态分配，访问判定基于“不上读、不下写”的信息流控制规则。这一模型在容器化环境中面临三方面挑战：首先，容器环境中的“主体”不再是单一用户进程，而是包含多个进程的容器进程集，访问控制的粒度需从单进程提升到容器级别；其次，容器间既需要严格隔离又需要在特定条件下共享数据卷，传统模型的二元关系难以表达“同组共享、跨组隔离”的复合访问需求；第三，容器的生命周期远短于传统系统中的用户账户，频繁的创建与销毁要求安全标签能够动态分配和回收，而非静态固化。DLSM 将传统 MAC 的“主体 – 客体”二元模型扩展为“容器进程集 – 数据卷文件 – 容器组”三维关系，将多级安全思想从“用户 – 文件”的一维安全等级推广到“容器身份 × 组归属 × 安全等级”的多维安全空间，在保留 MAC 强制约束优势的同时，实现了面向容器场景的灵活策略管理。

表 1 对比了 DLSM 与主流方案的差异。

1.2 隔离机制框架设计

多级安全隔离机制的核心框架如图 1 所示，由以下部分构成：

（1）策略配置与解析层：读取和解析外部配置文件，包括受限路径、容器组信息、安全标签等，将抽象安全策略转化为可执行的访问控制规则。

（2）主体与客体标识模块：主体定义为容器进程集，客体为宿主机关键目录、容器读写层及共享数据卷文件。通过 PID 与容器 ID 映射、文件路径与安全标签绑定，为内核访问判定提供上下文信息。

（3）访问控制决策模块：基于 MLS 安全标签匹配机制，容器仅能访问属于自身 ID 的文件与非受限宿主机目录。对共享数据卷文件，需比较容器的组 ID、等级 ID 与目标文件安全标签，判定读写权限。

（4）安全标签管理与持久化模块：在文件系统扩展属性中记录文件的安全标签（如 {container_id，group_id，level_id}），保证容器或宿主机重启后策略一致。

表 1　典型容器安全隔离方案对比分析

比较维度	SELinux	AppArmor	gVisor	eBPF 方案	DLSM
隔离粒度	粗粒度（基于 Type）	中等粒度（基于路径）	极细粒度（内核模拟）	细粒度（系统调用级）	多级细粒度（标签 + 分组）
配置复杂度	极高	中等	低	中等	低（自动化）
数据卷支持	仅静态标签	路径匹配有限	良好	不直接支持	动态绑定细粒度控制
性能开销	<2%	<2%	15% ~50%	<3%	<5%
主要不足	配置复杂易误配置	无法防护内核级漏洞	性能损耗大	需额外开发策略	需依赖内核版本

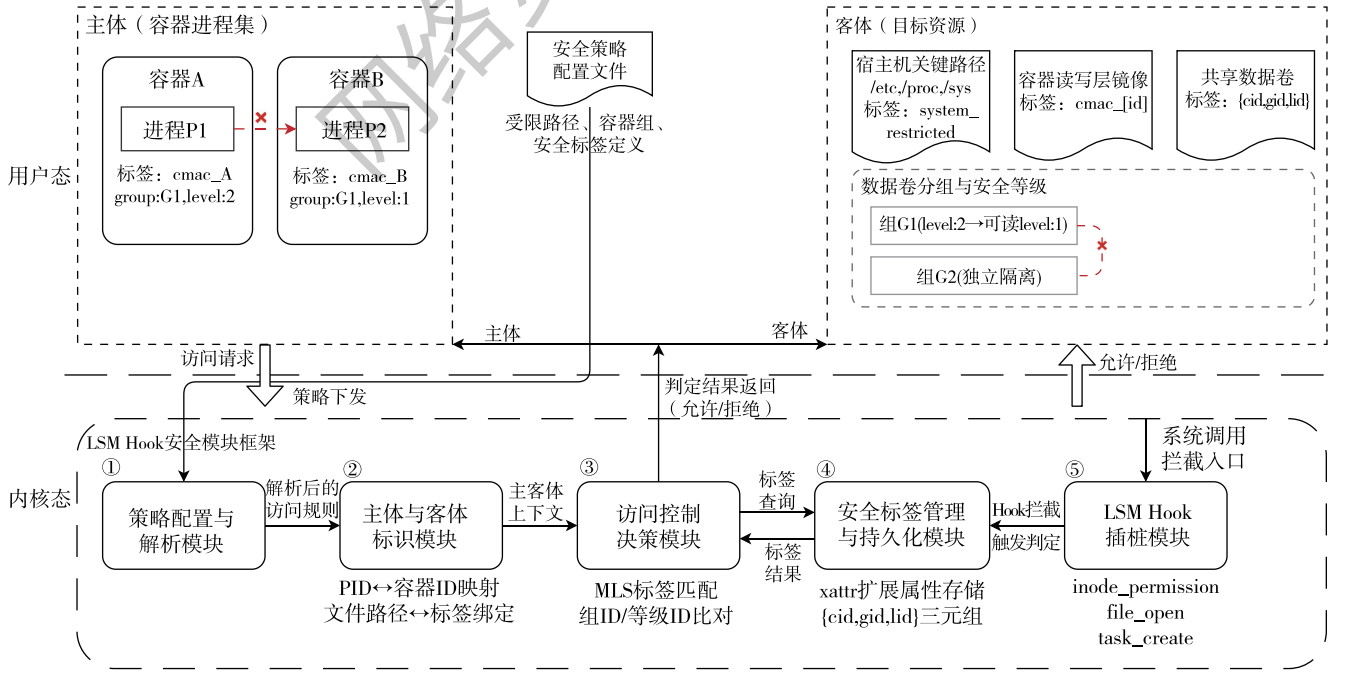

图 1　多级安全隔离总体架构

（5）内核态 LSM Hook 插桩：借助 LSM 提供的 inode_ permission、file_ open、task_ create 等钩子函数，将访问控制逻辑嵌入文件系统访问与进程创建的关键路径，实现内核级强制控制。

1.3 DLSM 系统整体架构

在隔离机制框架基础上，DLSM 系统进行了扩展，其整体架构如图 2 所示，主要由以下三个模块构成：

（1）容器进程隔离模块：确保容器进程仅能访问属于自身的读写层镜像文件及允许的宿主机文件，阻止对其他容器或受限目录的访问。在初始化阶段，扫描 Docker 安装路径，建立容器 ID 与读写层镜像目录的映射；在标签化阶段，为受限目录和容器镜像目录打上安全标签（如 system_ restricted、cmac_[container_id]）；在决策阶段，根据进程容器来源与目标文件标签匹配结果判定访问权限。

（2）共享数据卷隔离模块：在保证隔离的前提下支持容器间数据共享。通过分组机制将容器划分为不同组，每组有唯一组 ID，组内容器分配等级 ID。不同组间容器无法互访文件，同组容器根据安全等级判定权限——高等级可读低等级文件，低等级容器受限于只读或不可访问。

（3）策略管理与动态更新模块：提供运行时动态更新安全策略功能，支持配置文件热更新。策略更新后自动同步至内核安全标签缓存，保证新策略即时生效，并对所有拒绝的访问操作进行日志记录。

1.4 DLSM 系统实现

1.4.1 容器进程隔离实现

容器进程隔离的核心目标是限制每个容器的文件系统访问权限，禁止对宿主机关键路径或其他容器读写层的非法访问，从而防止容器逃逸攻击和越权操作。在标准 Docker 架构中，容器与宿主机共享内核，隔离仅依赖 Namespace 和 cgroups，/proc 和/sys 等目录通常以共享方式挂载，容器可从中提取宿主机信息。DLSM 在内核态引入基于 LSM hook 的强制访问控制逻辑，通过安全标签与容器 ID 匹配实施严格管控。

容器进程隔离包含三个步骤：

（1）容器标识与路径映射：每个容器具备唯一容器 ID，DLSM 在初始化时扫描 Docker 存储目录，建立容器 ID 与文件路径的对应映射表；

（2）文件标签化：宿主机敏感目录（如/etc、/var/lib/docker、/proc 等）标记为 system_ restricted，每个容器的读写层文件标记为 cmac_[container_id]；

（3）访问控制决策：LSM hook 捕获文件访问请求，若目标标签为 system_ restricted 则拒绝，若标签为 cmac_[container_ id] 且请求进程属于该容器则允许，否则拒绝。

Hook 插桩主要采用 inode_ permission、file_ open 等钩子，在文件打开和权限判定阶段阻断非法请求。安

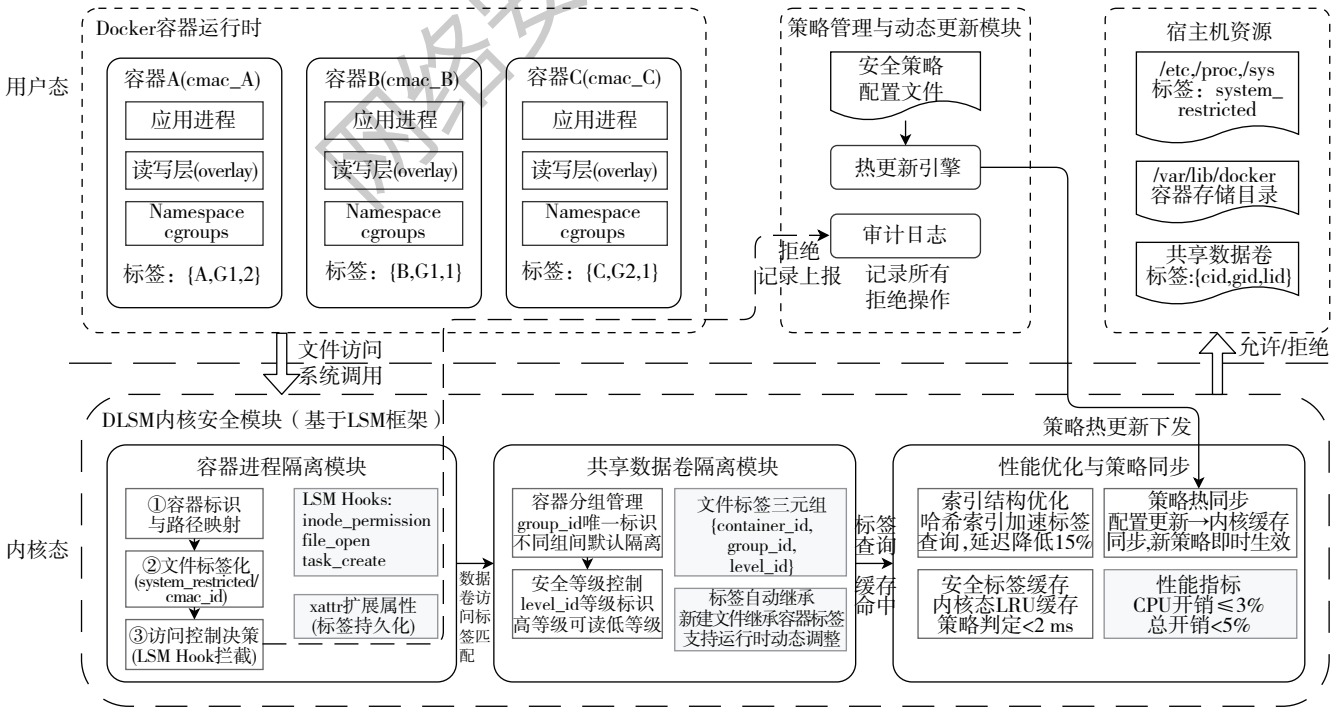

图 2　DLSM 整体架构图

全标签利用 Linux 扩展属性（xattr）存储，保证标签信息随文件持久化。通过读取/proc/[pid]/cgroup 信息获取进程所属容器 ID，从而实现容器识别。

1.4.2 共享数据卷隔离实现

在科研、制造等应用场景中，容器间存在数据共享需求。DLSM 引入容器分组与多级安全标签机制，实现安全共享与越权防护的平衡。

（1）容器分组管理：管理员根据业务需求将容器划分为多个独立组，每组有唯一组标识（group_id）。不同组容器间默认相互隔离。

（2）安全等级引入：在组内引入访问等级（level_id），高安全级别容器可读取低级别数据，低等级容器受限，实现“高读低”的多级安全访问控制。

（3）文件标签结构：文件安全标签记录为三元组{container_id，group_id，level_id}，通过与访问进程标签比对实现访问控制。管理员通过配置文件定义分组与等级，容器创建新文件时自动继承安全标签，且支持运行时动态调整。

1.4.3 实现性能优化

DLSM 实现了索引结构优化与缓存机制，降低访问控制开销。测试表明，相比原始 LSM，DLSM 在大规模容器环境下平均文件访问延迟降低约 15%，策略判定延迟稳定在毫秒级（<2 ms），CPU 开销增加≤3%。同时引入配置文件热更新功能，提升策略管理灵活性。

2 系统测试与实验分析

为验证 DLSM 系统的有效性与性能表现，搭建了实验环境并设计了多种攻击场景进行测试。

2.1 实验环境

实验在高性能服务器上进行，具体配置如表 2 所示。

表 2 实验环境配置

配置类别	配置项	具体参数
硬件	CPU	Intel Xeon Gold 6248R @ 3.00 GHz，24 核
	内存	64 GB DDR4
	存储	512 GB NVMe SSD
软件	操作系统	Ubuntu 22.04.3 LTS
	内核版本	Linux 5.15.0-91-generic
	容器运行时	Docker CE 24.0.5，containerd 1.6.21
测试规模	并发容器数	50 个实例，每项测试重复 10 次取均值

2.2 安全性对比测试

采用容器渗透测试工具 CDK（Container Penetration Toolkit）及公开漏洞，利用脚本构建测试用例集，涵盖三类典型攻击场景：

（1）容器逃逸：利用 CVE-2024-21626 和 CVE-2024-23651 尝试突破容器边界；

（2）敏感目录挂载：尝试挂载/etc/shadow、/var/run/docker.sock 等敏感路径；

（3）非授权跨容器通信：通过共享内存、Unix 域套接字等方式非法访问邻居容器数据。

DLSM、SELinux 和 AppArmor 三种方案在相同攻击场景下的安全防护效果对比如表 3 所示。

表 3 安全攻击拦截对比测试结果

攻击场景	测试用例	DLSM	SELinux	AppArmor
容器逃逸	CVE-2024-21626	拦截（45 μs）	拦截（52 μs）	未拦截
	CVE-2024-23651	拦截（38 μs）	部分拦截①	未拦截
敏感目录	mount 宿主机根目录	拦截（32 μs）	拦截（48 μs）	拦截（35 μs）
	读取/etc/shadow	拦截（28 μs）	拦截（42 μs）	拦截（30 μs）
	访问 docker.sock	拦截（25 μs）	拦截（40 μs）	部分拦截②
跨容器通信	共享内存越权	拦截（18 μs）	未拦截	未拦截
	Unix 套接字越权	拦截（22 μs）	未拦截	未拦截
数据卷越权	跨组数据卷读取	拦截（15 μs）	未拦截	未拦截
	低写高等级卷	拦截（16 μs）	未拦截	未拦截
拦截成功率/%		100	55.6	33.3
平均误报率/%		0.2	1.8	0.5

注：①CVE-2024-23651 为 BuildKit 构建缓存竞态条件漏洞，攻击者通过操纵构建缓存挂载点的竞态窗口实现容器逃逸。SELinux 的默认容器策略（container_t 类型）仅对容器运行时的文件系统访问进行类型约束，未覆盖 BuildKit 构建阶段的临时缓存挂载行为——当攻击利用 TOCTOU（Time-of-Check-to-Time-of-Use）竞态条件在短暂时间窗口内替换缓存挂载内容时，SELinux 可阻止最终的越权文件写入（因目标路径的 Type 标签不匹配），但无法拦截竞态窗口内的缓存内容篡改，导致攻击链的前半段得以执行。②AppArmor 基于路径名的访问控制可限制容器进程直接读写/var/run/docker.sock 文件，但当容器通过符号链接、bind mount 或/proc/[pid]/root 路径间接引用该套接字时，AppArmor 的路径匹配规则无法追踪路径的等价关系，导致间接访问路径绕过了配置文件中的显式路径限制

对比结果表明，DLSM 在所有 9 类攻击场景中均成功拦截，SELinux 在跨容器通信和数据卷越权场景中因缺乏对应策略而无法拦截，AppArmor 在容器逃逸和部分跨容器场景中存在防护盲区。DLSM 的拦截延迟整

体优于 SELinux（平均低 27%），这得益于其轻量化的标签匹配机制。攻击拦截延迟对比如图 3 所示。

2.3 性能开销评估

在开启与关闭安全模块的情况下，分别使用 UnixBench 和 wrk 对容器性能进行对比测试，结果如表 4 所示。

实验表明，虽然 DLSM 的性能开销略高于 SELinux 和 AppArmor（均≤5%），但其安全防护能力显著优于后两者。综合考虑安全性与性能，DLSM 在可接受的性能损耗下提供了全面的多级安全防护，更适合对安全性要求较高的生产环境。性能开销对比如图 4 所示。

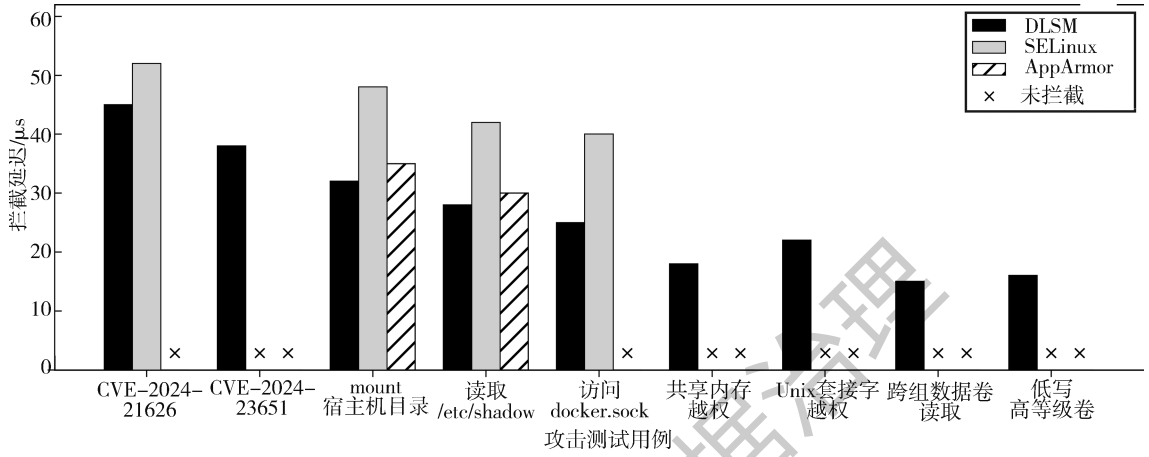

图 3 攻击拦截延迟对比

表 4 性能开销对比测试结果

测试指标	原生 Docker	DLSM	SELinux	AppArmor
UnixBench 综合得分	4 520	4 408（-2.5%）	4 443（-1.7%）	4 465（-1.2%）
文件 I/O 吞吐量/(MB/s)	450	433（-3.8%）	441（-2.0%）	445（-1.1%）
HTTP 吞吐量（wrk)/(req/s)	12 500	12 180（-2.6%）	12 275（-1.8%）	12 350（-1.2%）
HTTP 平均延迟/ms	8.0	8.3（+3.8%）	8.2（+2.5%）	8.1（+1.3%）
系统调用延迟增量/μs	—	+0.03	+0.02	+0.01

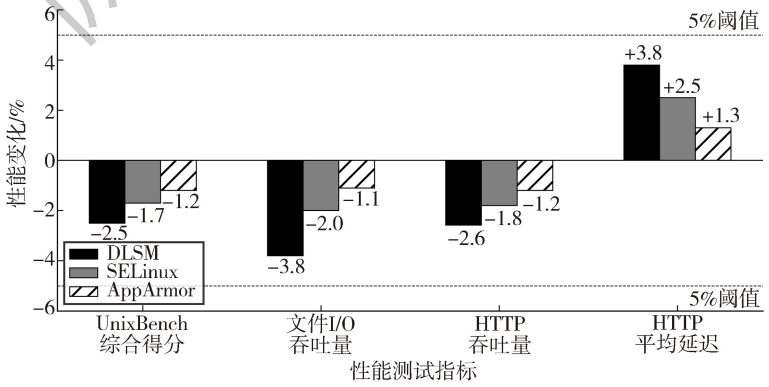

图 4 性能开销对比

为进一步分析 DLSM 各功能模块对系统性能的影响，将总开销按组件进行分解，结果如图 5 所示。其中，策略匹配模块的 CPU 和延迟开销最大，标签查询模块的内存开销最高，而缓存同步和日志记录的开销较小。这表明后续优化应优先聚焦于策略匹配效率的提升。

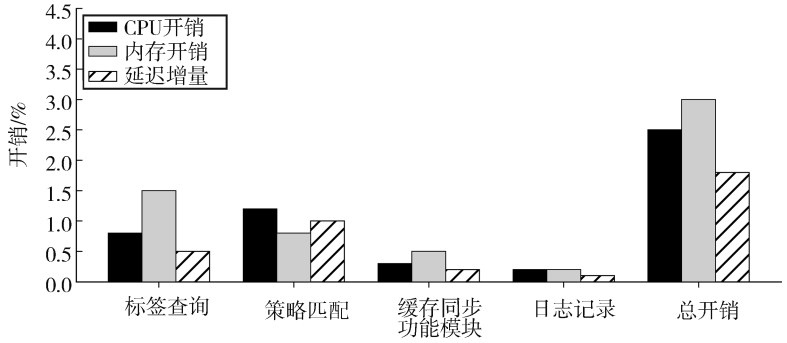

图5　DLSM 各组件性能影响分解

3　结论

本文围绕 Docker 容器在信息系统中的安全隔离问题，提出并实现了基于 LSM 的多级安全隔离系统 DLSM。系统通过安全标签实现容器进程强制隔离，通过分组与等级机制实现共享数据卷细粒度访问控制，并通过索引优化与缓存机制将性能开销控制在 5% 以内。与 SELinux、AppArmor 的对比实验表明，DLSM 在容器逃逸、敏感目录挂载、跨容器通信及数据卷越权等场景中均实现 100% 拦截，拦截成功率显著优于现有方案。

未来研究将集中在两个方向：一是引入 eBPF 技术以实现非侵入式实时监控，进一步降低性能开销并支持动态策略更新；二是结合异常检测技术构建容器行为画像，识别和阻断未知攻击变种。

参考文献

[1] GAO X, GU Z, KAYAALP M, et al. ContainerLeaks: emerging security threats of information leakages in container clouds[C]// Proceedings of the 47th Annual IEEE/IFIP International Conference on Dependable Systems and Networks. Denver: IEEE, 2017: 237 – 248.

[2] JARKAS O, KO R, DONG N, et al. A container security survey: exploits, attacks, and defenses[J]. ACM Computing Surveys, 2025, 57 (7).

[3] Palo Alto Networks. PAN-SA-2024-0002: Impact of leaky vessels vulnerabilities (CVE-2024-21626, CVE-2024-23651, CVE-2024-23652, and CVE-2024-23653)[EB/OL]. (2024 – 02 – 22)[2025 – 12 – 01]. https://security.paloaltonetworks.com/PAN-SA-2024-0002.

[4] 肖永康, 纪翠玲, 谢宝恂, 等. SELinux 的安全机制和安全模型[J]. 计算机应用, 2009, 29 (S1): 66 – 68.

[5] 褚冰曦. 基于零信任策略的 Docker 容器安全加固方案的设计与实现[D]. 天津: 南开大学, 2022.

[6] 朱明达. 容器安全防护技术研究与开发[D]. 西安: 西安电子科技大学, 2024.

[7] 黄轲, 李璇, 周庆飞, 等. 基于 eBPF 的容器运行时可信监控方案[J]. 信息网络安全, 2025, 25 (2): 306 – 326.

[8] HE Y, GUO R, XING Y, et al. Cross container attacks: the bewildered eBPF on clouds[C]//Proceedings of the 32nd USENIX Security Symposium, 2023: 5971 – 5988.

[9] YU X, HAN Z. A novel container escape detection method based on system call dependency graph[J]. Applied Sciences, 2024, 14 (3): 1120.

[10] SULTAN S, AHMAD I, DIMITRIOU T. Container security: issues, challenges, and the road ahead[J]. IEEE Access, 2019, 7: 52976 – 52996.

[11] Resolved Security. CVE-2025-52881: Privilege escalation vulnerability in runc (Go)[EB/OL]. (2025 – 11 – 04)[2026 – 03 – 04]. https://www.resolvedsecurity.com/vulnerability-catalog/CVE-2025-52881.

[12] AHMAD R, ALSMADI I. Cyber security in containerization platforms: a comparative study of security challenges, measures and best practices[J]. arXiv preprint arXiv: 2404.02029, 2024.

[13] ZENG W J, FAN R, WANG Z, et al. Research on Docker container network isolation and security management for multi-tenant environments[C]//Proceedings of the 2023 International Conference on Communication Network and Machine Learning. New York: ACM, 2023: 1 – 6.

[14] MARTINEZ DELBUGIO J, MADISETTI V K. Enhanced memory-safe Linux security modules (eLSMs) for improving security of Docker containers for data centers[J]. Journal of Software Engineering and Applications, 2024, 17 (5): 259 – 269.

(收稿日期: 2026 – 01 – 12)

作者简介:

熊明俊 (1987 –), 男, 本科, 高级工程师, 主要研究方向: 信息安全、容器安全。

郭培馨 (1994 –), 女, 硕士, 工程师, 主要研究方向: 计算机科学与技术、网络安全。

版权声明

凡《网络安全与数据治理》录用的文章，如作者没有关于汇编权、翻译权、印刷权及电子版的复制权、信息网络传播权与发行权等版权的特殊声明，即视作该文章署名作者同意将该文章的汇编权、翻译权、印刷权及电子版的复制权、信息网络传播权与发行权授予本刊，本刊有权授权本刊合作数据库、合作媒体等合作伙伴使用。同时，本刊支付的稿酬已包含上述使用的费用，特此声明。

《网络安全与数据治理》编辑部