浪潮信息

基于eBPF的内核态容器运行时安全实践

浪潮信息云峦KeyarchOS基于eBPF技术的容器安全方案

甄鹏

浪潮信息操作系统安全专家

01

容器面临的安全挑战

- 容器运行前安全风险
- 容器运行时安全威胁

02

KeyarchOS基于eBPF技术的解决方案

- 容器运行时安全方案
- 技术架构
- 效果展示

03

总结与展望

- 总结
- 展望

容器面临的安全挑战

容器运行前及运行时的安全挑战

容器运行前安全风险

容器运行前的风险主要指的是在容器实际运行之前，可能面临的各种安全风险和挑战。这些风险可能影响容器的安全性、稳定性和可用性，进而影响整个应用程序或系统的正常运行。

镜像

- 恶意镜像
- 漏洞镜像
- 敏感信息泄露

生态

- 编排工具漏洞
- 编排工具配置缺陷

运行环境

- 宿主机内核漏洞
- 宿主机内核配置缺陷

容器运行时安全威胁

容器运行时安全威胁指的是在容器化应用部署、执行和管理过程中，可能面临的各种安全问题和风险。这些威胁可能源自容器技术本身的局限性、配置不当、外部攻击者的恶意行为或者是容器运行时环境（如宿主机、网络、存储等）的漏洞。运行时威胁总结为两大类：

容器逃逸

容器本应提供轻量级的隔离环境，但如果隔离机制存在缺陷或配置不当，攻击者可能会突破这些隔离限制，访问或控制其他容器或宿主机上的资源。

- 危险配置导致的容器逃逸
- 危险挂载导致的容器逃逸
- Docker程序自身存在漏洞导致的容器逃逸
- 容器所在系统内核漏洞导致的逃逸

容器入侵

容器入侵通常指的是攻击者利用容器技术中的安全漏洞，对容器进行未授权的访问、控制或破坏。

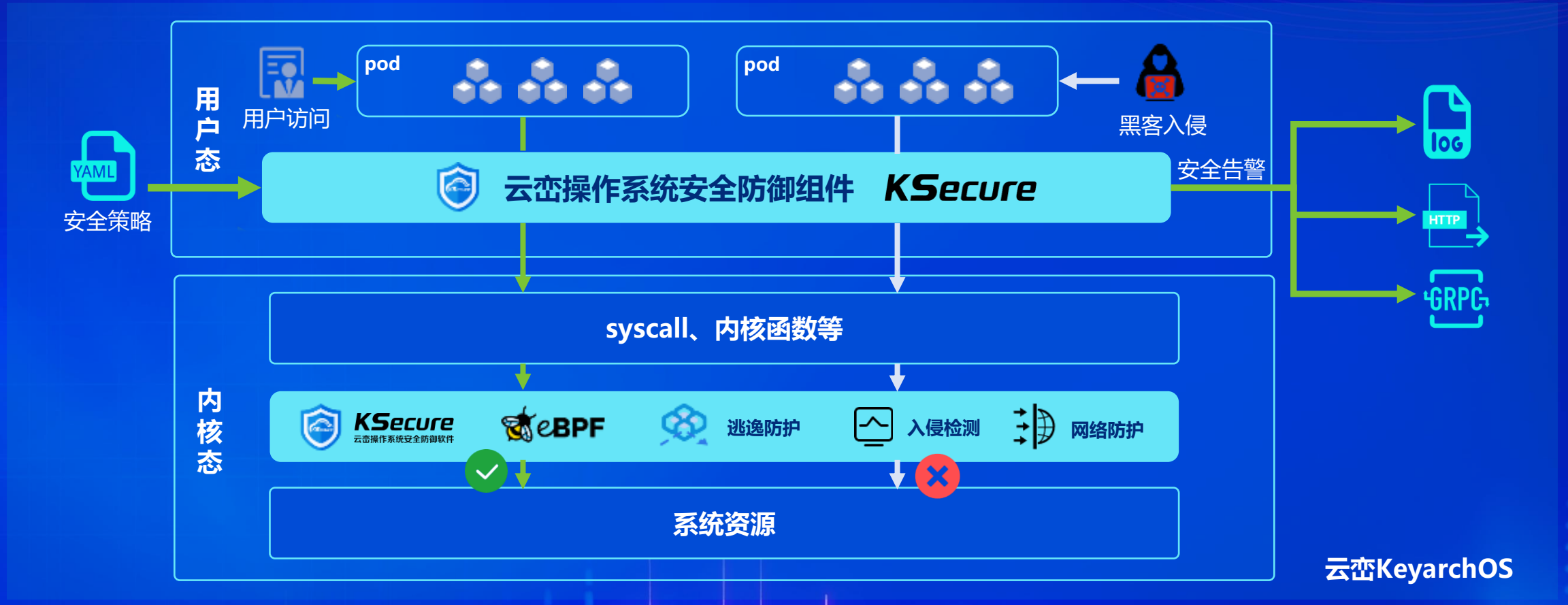
- 获取控制权限
- 上传病毒加密数据
- 隐藏痕迹窃取数据
- 横向扩展

KeyarchOS基于eBPF技术的解决方案

容器运行时安全方案及关键技术

容器运行时安全方案

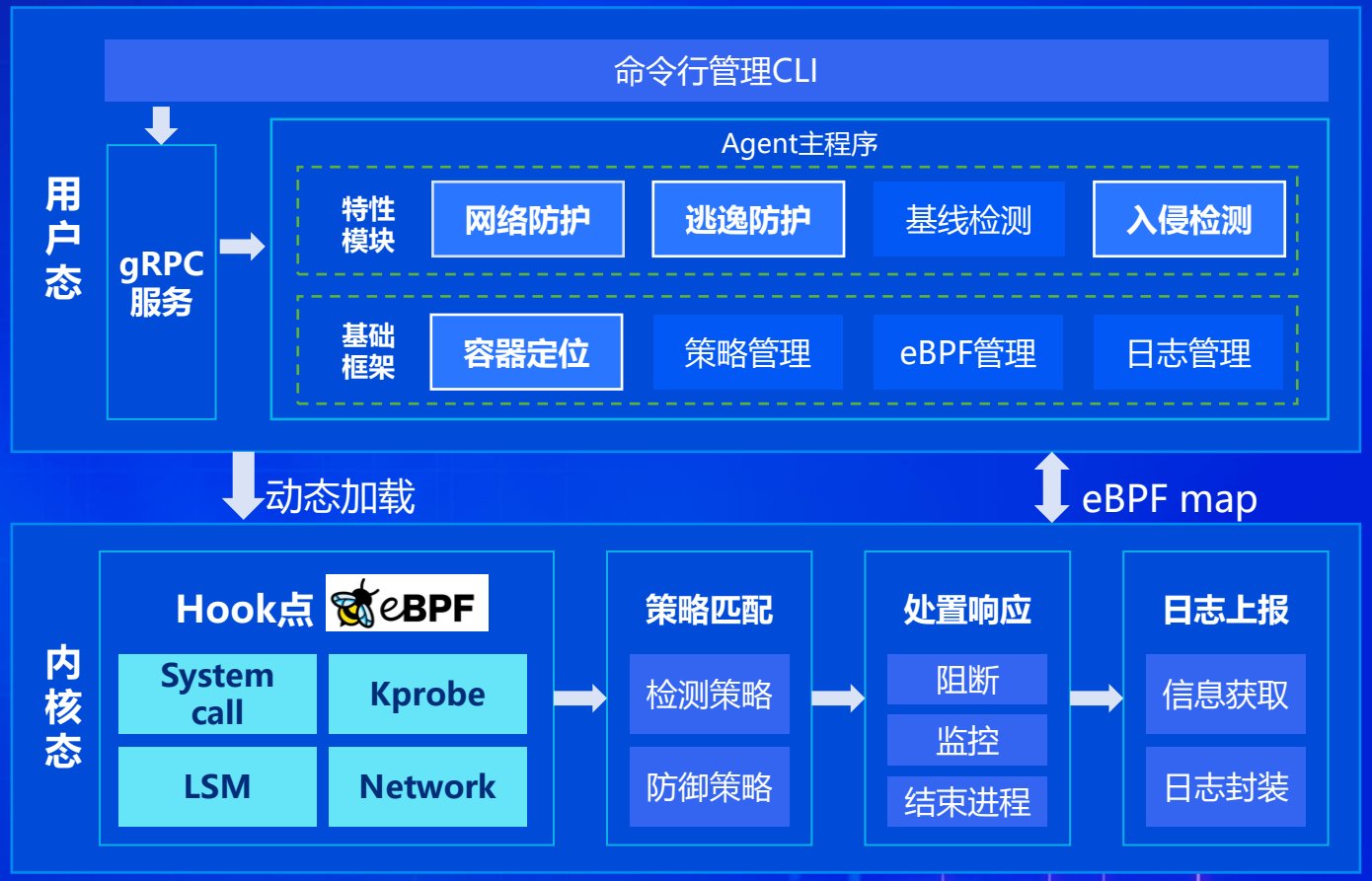
容器与宿主机共享内核，挂载eBPF程序至宿主机内核，一是能够同时监控宿主机和容器，二是部署一套安全组件能监控所有容器，在部署效率和安全管理上均有优势。



浪潮信息

技术架构

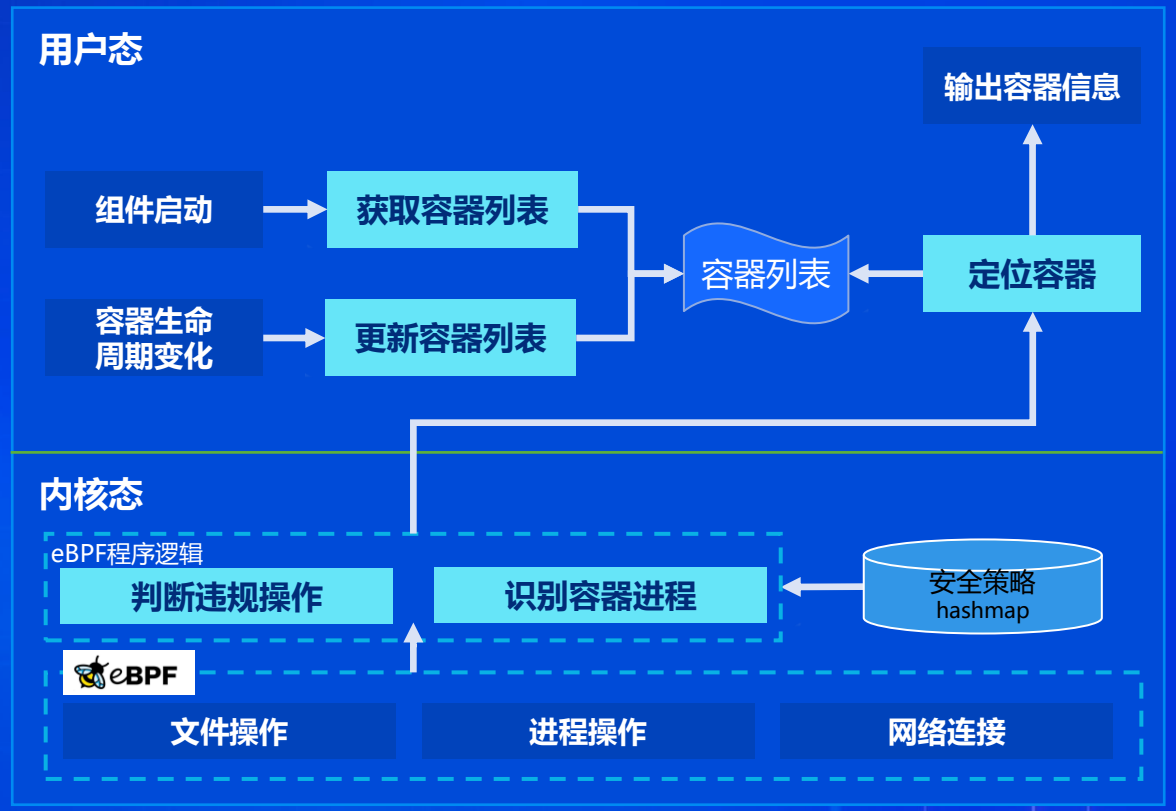
基于eBPF的系统内多层次hook技术，将eBPF程序hook到操作系统内核的多个层级（syscall、 LSM、 network 、kprobe内核函数），在各个hook点加载安全策略对系统和应用程序行为监控和拦截



- **用户态**：提供CLI用户接口，通过gRPC服务与主进程通信。通过基础框架支撑安全特性功能开发，通过特性开关调用将eBPF程序加载至内核
- **内核态**：在内核对应事件触发时，运行相应的 eBPF 字节码程序，与配置的安全策略匹配，根据策略响应并上报安全日志
- **通信**：用户态和内核态策略下发以及日志上传通过eBPF map实现

技术架构：容器定位

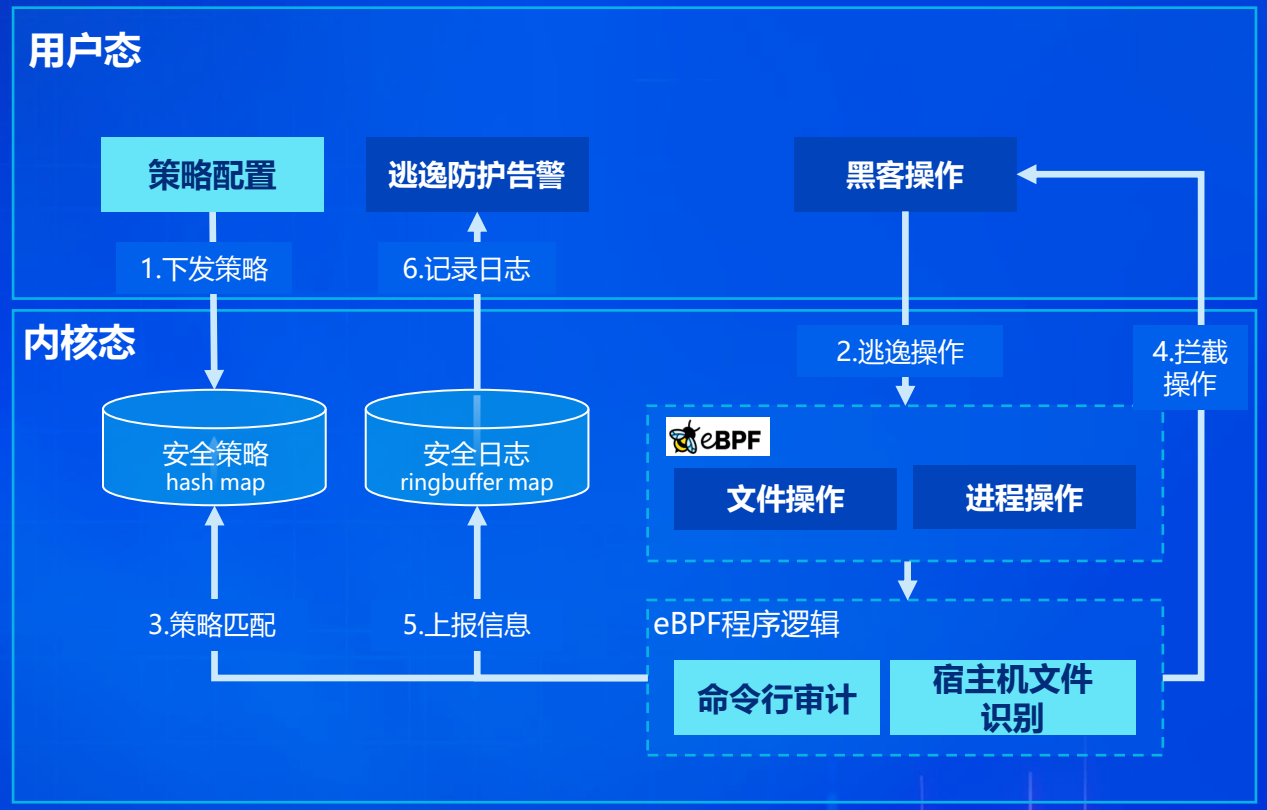
eBPF程序均挂载在容器宿主机的内核态，部署于宿主机的容器发生入侵或逃逸行为时，内核态eBPF程序无法直接定位容器信息，需要内核态获取更多进程信息，配合用户态获取的容器信息定位具体容器



- **内核态命名空间信息采集：**eBPF程序监控内核函数，获取进程信息，通过进程PID命名空间和挂载命名空间，识别容器进程和宿主机进程。将该容器进程信息、PID命名空间、挂载命名空间上传至用户态。
- **用户态容器信息采集及容器定位：**获取宿主机所有运行容器列表，监控容器创建和销毁，更新容器列表。通过容器客户端获取容器信息（容器进程信息、PID命名空间、挂载命名空间、容器名称、容器镜像、Pod信息等）。内核态容器信息上传至用户态，通过PID命名空间、挂载命名空间定位容器。

技术架构：逃逸防护

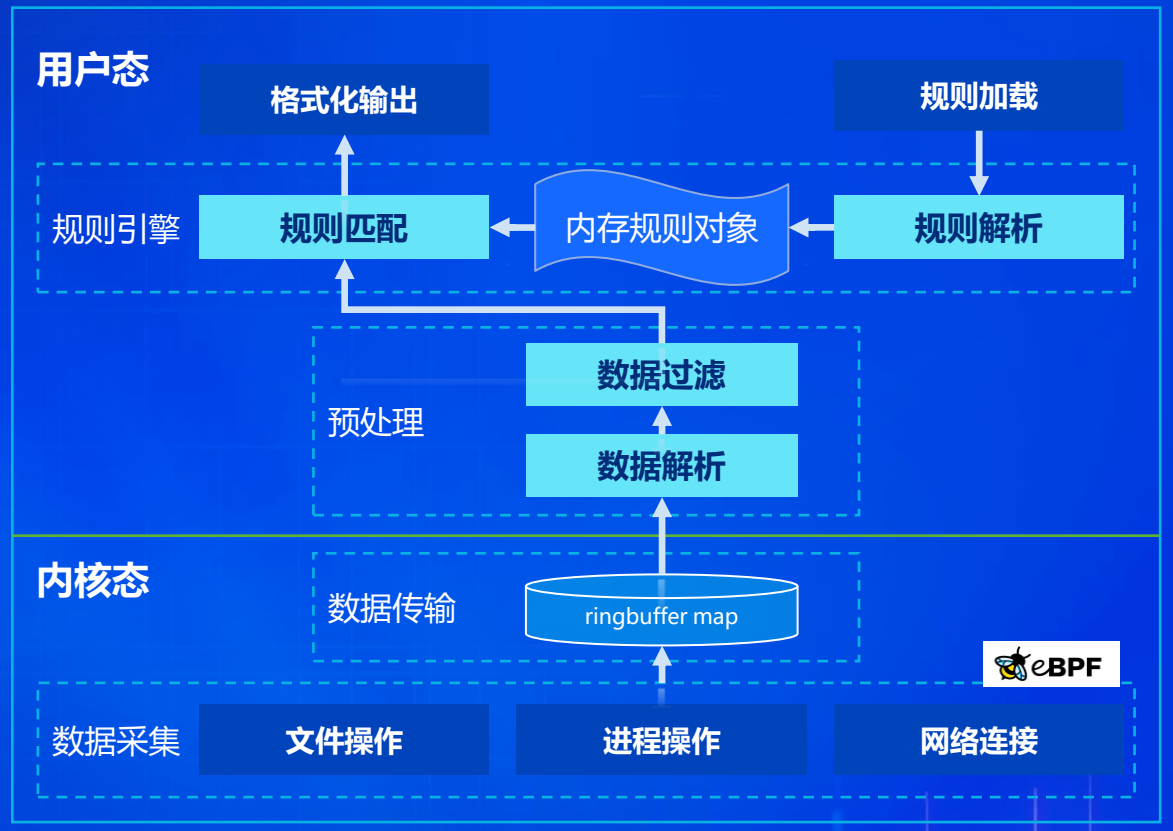
分析容器逃逸行为，从容器内发生操作的进程和文件维度识别并阻断逃逸行为。具体而言，对进程命令行参数审计结合可疑命令库识别逃逸行为；对容器进程操作的文件路径分析，判断是否为宿主机文件，并进一步识别逃逸行为。



- **容器内进程命令行审计**：通过Kprobe获取进程的命令行参数，在LSM hook逻辑中将命令行参数与可疑逃逸指令库比对，对匹配的进程判定为逃逸行为上报告警，并通过eBPF LSM机制进行细粒度拦截
- **容器内操作文件所属识别**：在文件操作的系统调用处挂载eBPF LSM程序，获取文件的path参数，结合内核数据结构目录项、虚拟文件系统挂载点等内核数据计算该文件在宿主机的实际路径，路径中包含容器信息则为容器内文件，否则识别为逃逸行为

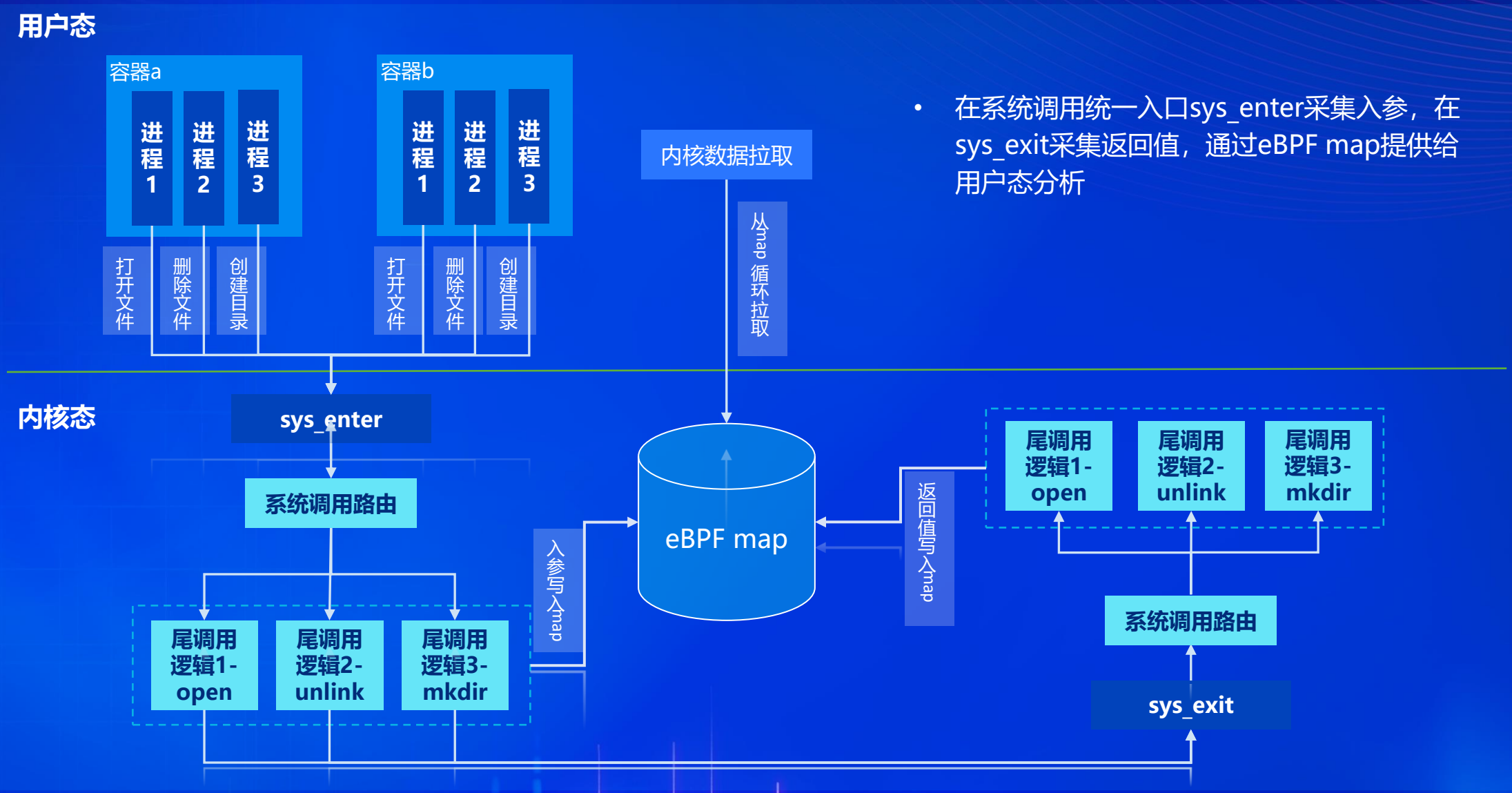
技术架构：入侵检测

将eBPF程序通过kprobe、tracepoint技术挂载至内核，监控系统中的文件操作、进程创建、消亡、调用以及网络连接等行为。基于**MITRE ATT&CK**框架构建入侵检测内置规则，为检测引擎提供判断依据，实现入侵事件识别



- **数据采集：**通过eBPF程序采集文件、进程、网络等系统调用入参以及返回值，存储至ring buffer类型的map
- **数据预处理：**读取ring buffer数据，根据不同的系统调用解析成特定的数据结构，并过滤掉不符合条件的数据
- **规则匹配：**将已加载的规则解析成内存对象，拉取预处理后的数据与内存中的规则对象进行比对，与规则匹配的数据诊断为入侵行为，格式化后输出

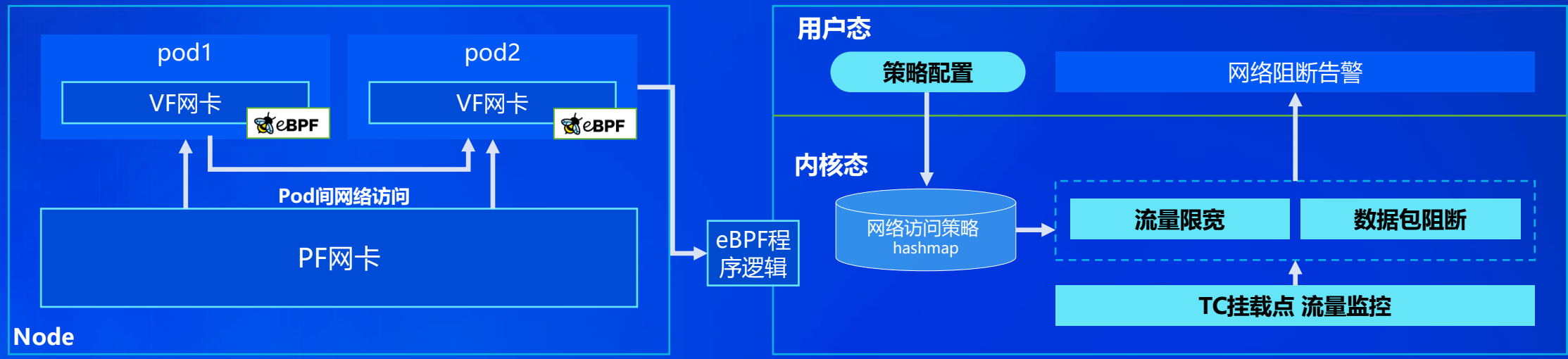
技术架构：入侵检测



- 在系统调用统一入口sys_enter采集入参，在sys_exit采集返回值，通过eBPF map提供给用户态分析

技术架构：网络防护

针对直接使用主机的物理网卡(underlayCNI)进行容器网络通信的场景，在pod内的虚拟网卡上监控网络流量，并根据网络防护策略限制流量和阻断数据包

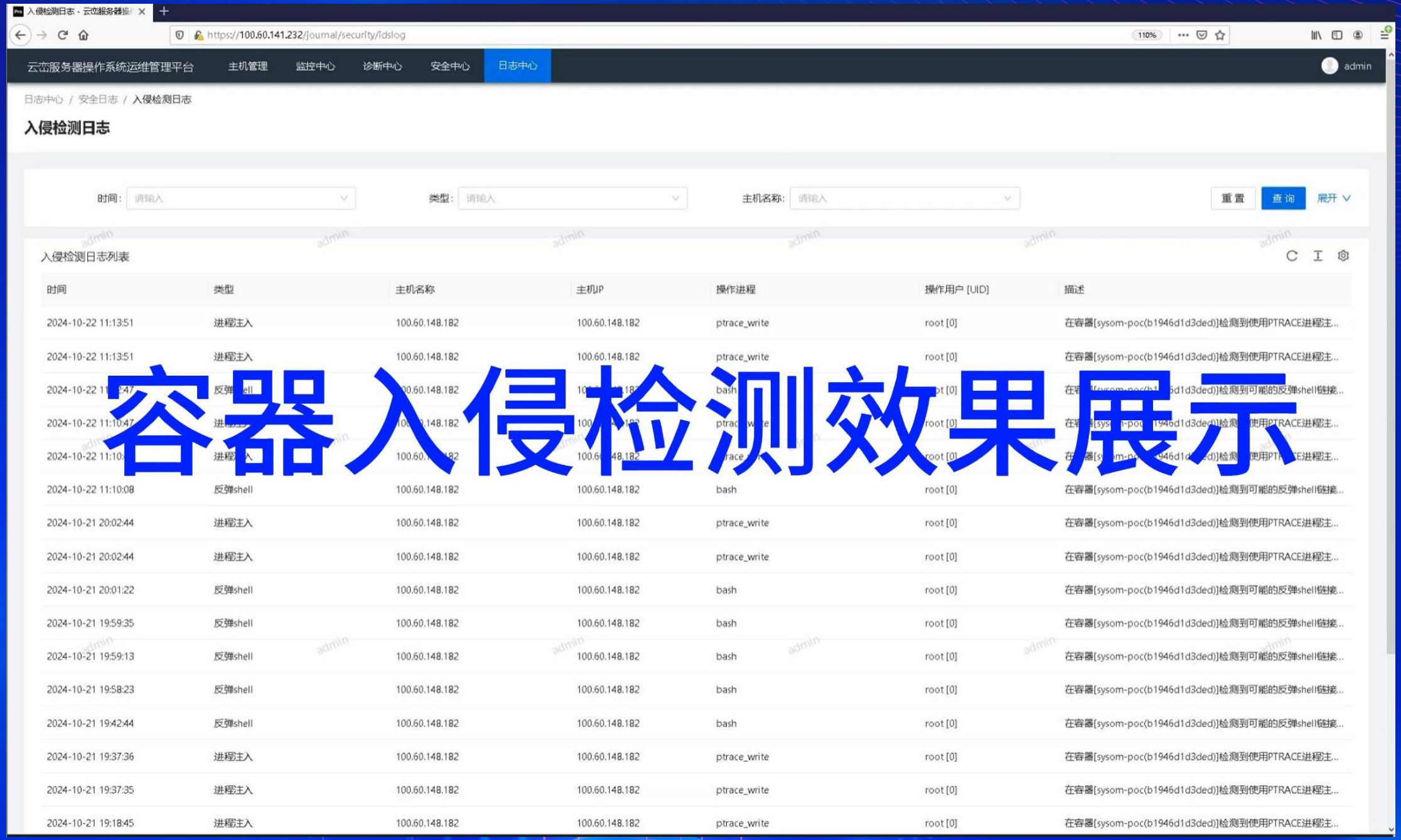


- **挂载网卡：** PF 网卡可以将物理网卡的资源划分为多个 VF 网卡，每个 VF 网卡可以独立配置并分配给不同的容器组pod，所有进出容器组pod 的数据包均经过VF网卡，此处可监控所有数据包。
- **挂载点：** TC(Traffic Control 流量控制框架)程序可以在数据包的 ingress 和 egress 点触发，能够做到数据包的阻断和流量限宽
- **阻断逻辑：** 内核态eBPF程序根据网络访问策略（IP、端口），阻断违规数据包（通过返回DROPPING阻断数据包，通过rate和ceil，用于控制数据包的传输速率和最大速率

技术架构：网络防护

eBPF网络访问控制方案具有如下特点：

- **兼容性强**：eBPF程序作用于操作系统网络入口，能够避免与运行在操作系统内核网络协议栈的iptables、ipvs等传统容器间通信、负载均衡等网络处理功能形成依赖或干扰，从而能够兼容不同网络插件。
- **性能更优**：安全加固eBPF程序在操作系统网络入口生效，缩短了网络包处理路径，提升了整体性能。
- **可扩展性高**：基于eBPF可编程的特点，可以快速开发满足更多的需求场景，比如提供更加灵活易用的安全加固语义规则、支持标记/镜像等更多的网络包处理方式。
- **易于观测**：通过eBPF实现内核态与用户态交互，将被eBPF规则拦截的流量记录上报到用户态处理程序，从而可以直观的查询相关信息，便于问题的处理和统计分析。


入侵检测日志 - 云峦服务器操作
https://100.60.141.232/journal/security/idslog
云峦服务器操作系统运维管理平台
主机管理
监控中心
诊断中心
安全中心
日志中心
admin
日志中心 / 安全日志 / 入侵检测日志
入侵检测日志
时间：
类型：
主机名称：
重置
查询
展开
入侵检测日志列表
时间 类型 主机名称 主机IP 操作进程 操作用户 [UID] 描述
2024-10-22 11:13:51 进程注入 100.60.148.182 100.60.148.182 ptrace_write root [0] 在容器[sysom-poc(b1946d1d3ded)]检测到使用PTRACE进程注...
2024-10-22 11:13:51 进程注入 100.60.148.182 100.60.148.182 ptrace_write root [0] 在容器[sysom-poc(b1946d1d3ded)]检测到使用PTRACE进程注...
2024-10-22 11:10:47 反弹shell 100.60.148.182 bash root [0] 在容器[sysom-poc(b1946d1d3ded)]检测到可能的反弹shell链接...
2024-10-22 11:10:47 进程注入 100.60.148.182 ptrace_write root [0] 在容器[sysom-poc(b1946d1d3ded)]检测到使用PTRACE进程注...
2024-10-22 11:10:08 反弹shell 100.60.148.182 100.60.148.182 bash root [0] 在容器[sysom-poc(b1946d1d3ded)]检测到可能的反弹shell链接...
2024-10-21 20:02:44 进程注入 100.60.148.182 100.60.148.182 ptrace_write root [0] 在容器[sysom-poc(b1946d1d3ded)]检测到使用PTRACE进程注...
2024-10-21 20:02:44 进程注入 100.60.148.182 100.60.148.182 ptrace_write root [0] 在容器[sysom-poc(b1946d1d3ded)]检测到使用PTRACE进程注...
2024-10-21 20:01:22 反弹shell 100.60.148.182 100.60.148.182 bash root [0] 在容器[sysom-poc(b1946d1d3ded)]检测到可能的反弹shell链接...
2024-10-21 19:59:35 反弹shell 100.60.148.182 100.60.148.182 bash root [0] 在容器[sysom-poc(b1946d1d3ded)]检测到可能的反弹shell链接...
2024-10-21 19:59:13 反弹shell 100.60.148.182 100.60.148.182 bash root [0] 在容器[sysom-poc(b1946d1d3ded)]检测到可能的反弹shell链接...
2024-10-21 19:58:23 反弹shell 100.60.148.182 100.60.148.182 bash root [0] 在容器[sysom-poc(b1946d1d3ded)]检测到可能的反弹shell链接...
2024-10-21 19:42:44 反弹shell 100.60.148.182 100.60.148.182 bash root [0] 在容器[sysom-poc(b1946d1d3ded)]检测到可能的反弹shell链接...
2024-10-21 19:37:36 进程注入 100.60.148.182 100.60.148.182 ptrace_write root [0] 在容器[sysom-poc(b1946d1d3ded)]检测到使用PTRACE进程注...
2024-10-21 19:37:35 进程注入 100.60.148.182 100.60.148.182 ptrace_write root [0] 在容器[sysom-poc(b1946d1d3ded)]检测到使用PTRACE进程注...
2024-10-21 19:18:45 进程注入 100.60.148.182 100.60.148.182 ptrace_write root [0] 在容器[sysom-poc(b1946d1d3ded)]检测到使用PTRACE进程注...

容器入侵检测效果展示

总结与展望

总结

基于eBPF技术的容器防护是一种先进的安全措施，它利用eBPF的特性来增强容器的安全性。eBPF技术已经在安全领域取得了显著成果，并且随着技术的不断进步和应用范围的扩大，未来将展现出更大的潜力和价值。

轻量级安全
eBPF允许在内核中执行用户定义的程序而无需修改内核源代码，这使得安全解决方案更加轻量且灵活。

细粒度控制
通过eBPF技术可以实现对网络流量、系统调用等进行细粒度的监控和控制，这对于容器环境中的安全防护尤为重要

高性能
eBPF操作直接在内核层面完成，避免了用户态和内核态之间的频繁切换，从而提高了性能。

动态更新
eBPF程序可以在运行时动态加载和更新，这意味着可以在不重启系统的情况下调整安全策略。

集成性
eBPF可以很好地与其他容器管理工具（如Kubernetes）集成，为容器提供统一的安全管理界面。

展望

更深入的集成

预计未来 eBPF 将更紧密地集成到容器编排平台（如 Kubernetes）中，为容器提供更加无缝的安全管理体验。

智能化安全

结合 AI 和机器学习技术，eBPF 可以实现更加智能的威胁检测和响应机制，自动识别异常行为并采取相应措施。

标准化与规范

随着 eBPF 在容器安全领域的应用越来越广泛，相关的标准和最佳实践也将逐步建立起来，促进技术的规范化发展。

跨平台支持

当前 eBPF 主要在 Linux 系统上得到广泛应用，未来可能会有更多的操作系统支持 eBPF 技术，扩大其应用场景。

社区合作与发展

开源社区的合作将进一步推动 eBPF 技术的发展，提供更多高级功能和支持，加速技术创新的步伐。

THANKS

欢迎关注官方公众号