QINGTENG
青藤云安全

云原生安全白皮书

目录

执行摘要

目标

当前，采用云原生的开发和部署模式已日渐成为技术行业的发展趋势。同时，整个生态系统，包括技术、产品、标准和解决方案，也正在不断扩展，要求决策者要时刻了解复杂的设计。特别是CISO，要在这个动态发展的领域，实践业务价值。同时，云原生模式还鼓励改变消费模式，并采用现代化工作流（例如敏捷方法和 DevOps 流程），这就要求将安全集成进来。

问题分析

由于传统的基于边界的安全已不再可行，同时由于云原生明确关注快速开发与部署，在这种情况下，安全问题也就变得很复杂。这就要求转变安全范式来保护应用，将基于边界的安全方式转变为更接近动态工作负载的安全方式。动态工作负载是基于属性和元数据（例如标签）来确定的。通过这种方法，可以识别并保护工作负载，满足云原生应用的规模化需求，同时还能适应不断变化的需求。

安全范式的转变要求提高在应用安全生命周期中的自动化程度，并通过设计架构（例如零信任）来确保其安全。将容器化作为云原生环境中的一项核心变化，也需要新的最佳实践。容器化的安全实施会涉及组织机构内的多个相关方，并且会对追求业务目标的开发人员和运维人员的工作效率产生重大影响。云原生应用仍然需要开发、分发、部署和运维，但是这种范式就决定了，要有效实现这些目标，需要新的安全机制。

可以在应用生命周期的不同阶段（“开发”、“分发”、“部署”和“运行时”），对云原生开发进行建模。云原生安全与传统安全方法的不同之处在于，可以确保在生命周期的不同阶段将安全嵌入进来，而不是通过单独管理的安全干预措施来确保生命周期的安全。

要注意，如果不对这些概念、工具和流程的使用和集成进行长期的教育和培训，云原生的落地和应用可能就会难以难以为继，甚至被打回原形。

全生命周期的不同阶段

开发阶段

云原生工具旨在在应用生命周期的早期阶段引入安全。安全测试需要及早发现不合规情况和配置错误，以便缩短可行性反馈的周期，进行持续改进。这种方法可以确保，可以按照针对管道中其他问题而提出的类似工作流（例如 bug 修复或 CI 故障）解决安全故障问题。通常，需要先解决好这些安全问题，才能推动软件在管道中的进一步操作。

这种模式的现代化安全生命周期是围绕着代码开发而展开的，代码开发遵循的是推荐的设计模式（例如 12-Factor），并确保了所交付工作负载的完整性。从概念上来讲，云原生与基础架构即代码（IaC）密切相关，旨在确保通过早期安全检查集成，让控件能够按预期状况运行。通过这些控件和集成可以尽早识别错误配置并实施 IaC 和编排清单中的最佳实践，以减少长期成本并提高安全价值。

分发阶段

在支持软件快速迭代的模型中，软件供应链安全尤其重要。云原生应用生命周期需要采用一些方法，不仅可以验证工作负载本身的完整性，还可以验证工作负载的创建过程和运维方式。加之需要一直使用开源软件和第三方运行时镜像（包括上游依赖项的层），面临的挑战就更大了。

生命周期管道中存在的工件（例如容器镜像）需要进行连续的自动扫描和更新，确保免受漏洞、恶意软件、不安全编码方法和其他不当行为的侵害。完成这些检查后，重要的是对工件进行加密签名以确保完整性并强制执行不可否认性。

部署阶段

在整个开发和分发阶段集成了安全后，可以实现实时、持续验证候选工作负载属性（例如，验证签名的工件，确保容器镜像和运行时安全策略以及验证主机的适用性）。与工作负载一起部署安全工作负载的可观测性功能，可以以高度信任的方式监控日志和可用指标，对集成安全进行补充。

运行时阶段

预计云原生环境在设计时就会提供策略实施和资源限制功能。运行时工作负载的资源限制（例如 Linux 内核 cgroup 隔离）是在云原生环境中集成到应用生命周期的更高级别时的限制性和可观测性示例。可以将云原生运行时环境本身分解为相互关联的组件层，这些组件具有不同的安全问题（例如，硬件、主机、容器镜像运行时、编排）。[1]

在云原生运行时环境中，全球各行各业及各类组织机构都采用了微服务架构用于各类应用。应用通常由几个独立的、单一用途的微服务组成，这些服务通过容器编排层进行服务层抽象后进行通信。确保这种相互关联的组件体系结构安全的最佳实践包括：确保只有经过批准的进程才能在容器命名空间内运行，防止和通知未授权的资源访问情况，监控网络流量来检测恶意工具的活动。服务网格是另一种常见的抽象，它为编排服务提供了整合和补充的功能，而不会对工作负载软件本身进行任何更改（例如 API 流量日志、传输加密、可观测性标记、身份验证和授权）。

建议

云原生安全旨在确保实现像传统安全模型一样的谨慎性、完整性、信任和威胁防护，同时整合了临时性、分布式和不变性等现代概念。这些瞬息万变的环境支持故障转移，以便进行快速迭代。在这种环境中，需要在开发管道中实现自动化，以此来确保最终结果的安全性。组织机构必须认真分析并应用这些核心安全概念，缩短对加固和环境控制的延迟，并且需要让参与的第三方遵循相同的标准，同时平衡与云功能相关的、针对其安全支持者的持续教育和培训。

由于还要考虑到其他层的复杂性，而且还需要关注广泛的组件网格，因此，必须通过在整个生命周期内以及在运行时环境中集成安全，来防止未经授权的访问。强烈建议组织机构根据相关攻击框架[2]来评估安全防御堆栈，明确防御堆栈能够涵盖哪些威胁。此外，组织机构需要采用一些新的

[1]另一个要考虑的模型是云、群集、容器和代码：
https://kubernetes.io/docs/concepts/security/overview/
[2]示例——MITRE ATT&CK 框架（Kubernetes）

方式和方法将安全左移，扩大 DevOps 的能力。对于在生命周期内进行的创新，需要在管道之前、之中、之后持续妥善地验证所有组件。[3]

结论

在整个组织机构中有策略地执行云原生安全时，可以大规模地实现高可用性、保证、弹性和冗余，从而确保客户和开发人员能够按预期的速度安全访问所需资源。安全本身仍然是一个跨学科领域，不能与开发生命周期隔离开来，也不能视为纯粹的技术领域。开发人员、运维人员和安全人员必须加强交流和合作，才能继续推动该领域和行业的发展。与任何技术创新一样，人、激情以及整个发展过程共同造就了云原生社区，并实现了云原生安全。

[3]安全左移后，通常会让组织机构忽略对运维安全的监控。但安全存在于生命周期的所有阶段，并且组织机构必须不断评估其业务和技术流程的方方面面，这样才可能超越现代安全范式，从而形成一种文化和习惯。