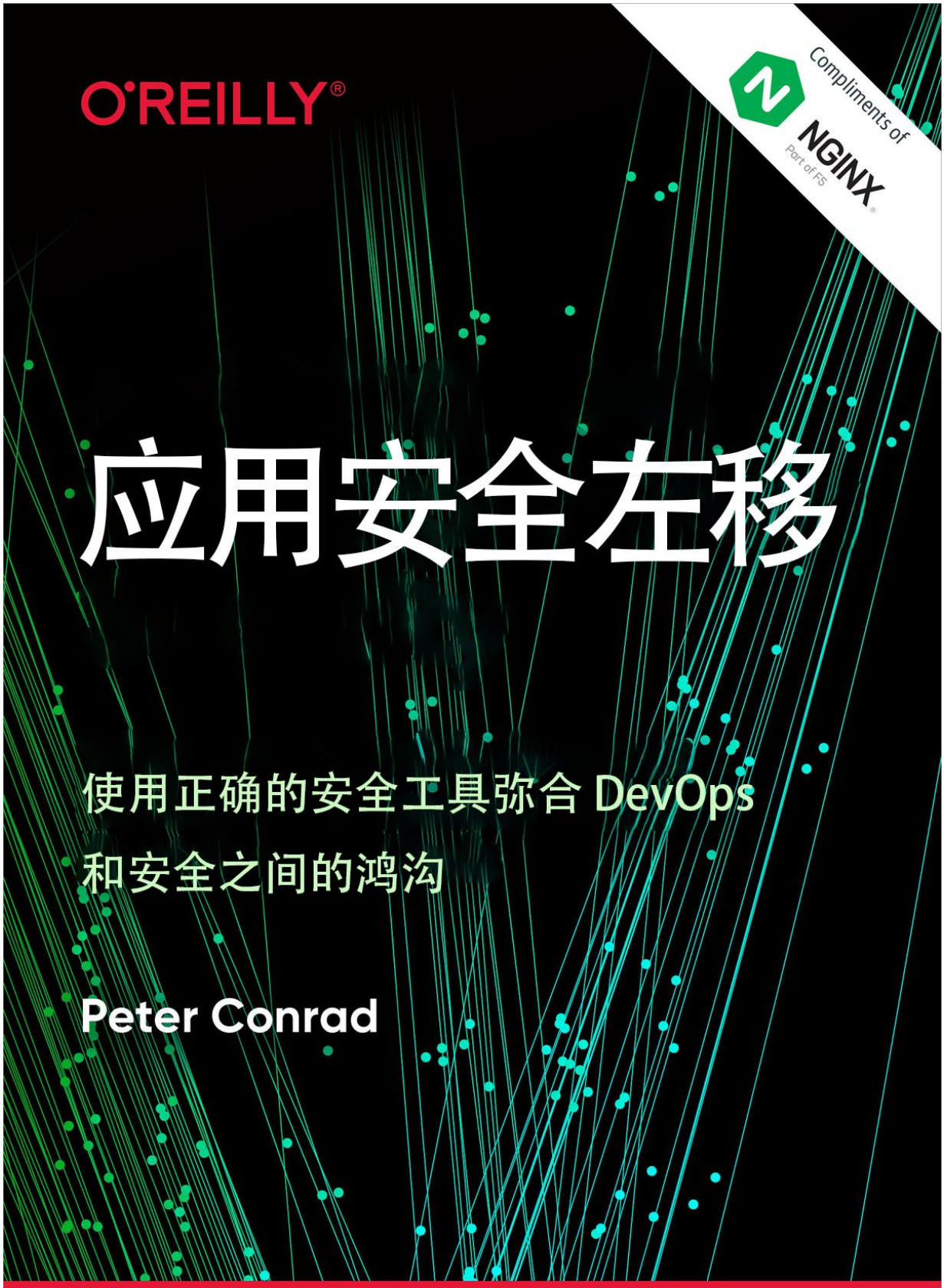

O'REILLY®
Compliments of
NGINX
Part of F5
应用安全左移
使用正确的安全工具弥合 DevOps
和安全之间的鸿沟
Peter Conrad
REPORT

应用安全左移

使用正确的安全工具弥合 DevOps 与安全之间的鸿沟

彼得·康拉德

应用安全左移

作者：彼得·康拉德

由 O’Reilly Media, Inc. 出版，地址：1005 Gravenstein Highway North, Sebastopol, CA 95472。

购买 O’Reilly 书籍可用于教育、商业或促销用途。大多数图书也提供在线版本（http://oreilly.com）。如需了解更多信息，请联系我们的企业/机构销售部门：800-998-9938 或 *corporate@oreilly.com*。

采购编辑：Jennifer Pollock 开发编辑：Gary O’Brien 制作编辑：Kate Galloway 文案编辑：Liz Wheeler 校对：Gregory Hyman 室内设计师：David Futato 封面设计师：Randy Comer 插画师：Kate Dullea

2022 年 4 月：第一版

第一版的修订历史

2022-04-06：首次发布

这项工作是 O’Reilly 和 NGINX 合作的一部分。请看我们的编辑独立性声明.

978-1-098-12732-9 [LSI]

前言

从设计到部署，安全对于应用程序开发和支持中的每个人都很重要。无论你是开发人员、安全或运维工程师，还是公司的 CISO（首席信息安全官），都已经正在考虑安全问题。全面地考虑安全，需要考虑所有可用的工具及其在软件开发流水线中的位置。

将安全左移意味着从流水线的最早阶段就引入工具和流程来承载安全性，而不是在最后阶段匆忙地进行一些安全测试草草了事。

本报告将帮助你了解为什么安全左移是如此重要以及如何做到这一点。你将了解组织在更快、更安全地交付应用程序的压力下所面临的挑战，同时左移如何帮助解决这些问题，并且需要整个组织改变思维才能实现这一切。

在报告结束时，你将能够向你的组织提出一些应用程序安全建议，并且将通过工具来创建你和你的团队可以应用的策略，以使安全成为每个人的首要任务。通过团队之间和谐的关系和一组共享的安全优先级，任何组织都可以通过将安全左移成功地使其应用程序、服务和开发流程更加健壮。

介绍

左移并不是一个新想法。随着现代软件流程的发展使一切变得连续，左移已经成为传统被局限于开发与生产阶段之间的测试阶段的各种流程的规范。安全左移意味着从设计的最早阶段开始实施安全策略、控制和设计，一直持续到生产。

尽管人们普遍认为安全左移是一个好主意，但很难就哪些工具和方法最适合这项任务达成一致。一部分公众的讨论关注在与代码和容器相关的扫描工具、自动修复以及专为现代应用程序和基础设施设计的其他新的安全工具上。具备在运行时保护应用程序的悠久历史的工具似乎已经过时，被贴上“遗留”工具的标签，在现代软件开发环境中失去了地位。

事实证明，这些有着长期历史的工具在当今的企业中仍非常重要。 诸如 Web 应用程序防火墙 (WAF) 这类工具提供了运行时保护和有关应用程序性能的宝贵反馈，帮助开发人员和安全工程师完善代码和策略。 WAF 和其他工具并没有停滞不前，而是已经适应了现代基础设施和应用程序，随着威胁行为者开发出更复杂的攻击而变得更加智能。

随着企业将应用程序迁移到云端，左移变得越来越重要。现代应用程序不再是单体，而是由大量服务组成，这些服务呈现出复杂、多样化的攻击面，仅靠代码扫描和良好的编程实践已无法防御。左移必须不仅仅是设计上的安全性。企业需要执行一个有意识的文化转变，使安全成为每个人的责任，从设计到生产的整个过程都要涵盖，这一转变的驱动力是经由利用来自运行时工具的反馈而不断完善的政策。

传统上，开发团队和安全团队似乎总是处于对立状态。开发人员面临着快速交付更多功能的压力，而安全团队则被视为看门人，有时会停止开发以调查问题。安全左移所必需的文化变革需要开发团队与安全团队的共同合作。这意味着每个参与者都需要新的工作方式。安全团队必须成为良好实践的推动者，提供保护应用程序和基础设施的护栏，同时又不会阻碍工程进展。从设计到部署，开发团队必须承担一定的安全责任。

从根深蒂固的流程到预算，再到内部政治，许多因素都可能致使如此重大的文化转变举步维艰。本报告对云原生应用程序架构的当前安全形势、一些类型的威胁以及将左移策略引入持续集成和持续部署 (CI/CD) 软件流水线的一些策略和工具进行了调查。

第 1 章 DevOps 和 DevSecOps

传统的软件开发过程专注于创建单体应用程序，它将设计、构建、测试和交付分为不同的阶段，但是事后看来，这种方法显然既不灵活也不高效。为了应对日益增长的压力，敏捷开发方法将大型任务分解为若干更小的单元，从而使得流程变得更加灵活。随着研发人员遍布全球，将大家束缚在单一的代码库上已不再现实。应用程序架构将从单体模型转向微服务，这使得分散的团队可以在不同的服务上分别工作且不会互相影响。管理日益庞杂的微服务开发迫使软件开发流水线中的所有步骤都是连续的，这也是 DevOps 发挥作用的所在之处。

DevOps

DevOps 是一种概念，是涵盖了多项目标的一系列实践。DevOps 将开发和运营结合在一起，模糊了开发、构建、部署和维护应用程序各阶段之间的职责界限，也打破了负责其中不同任务团队之间的界限，使他们能够建立应用程序和功能的持续集成和部署。图 1-1 展示了软件开发和部署 CI/CD 方法的各个阶段。

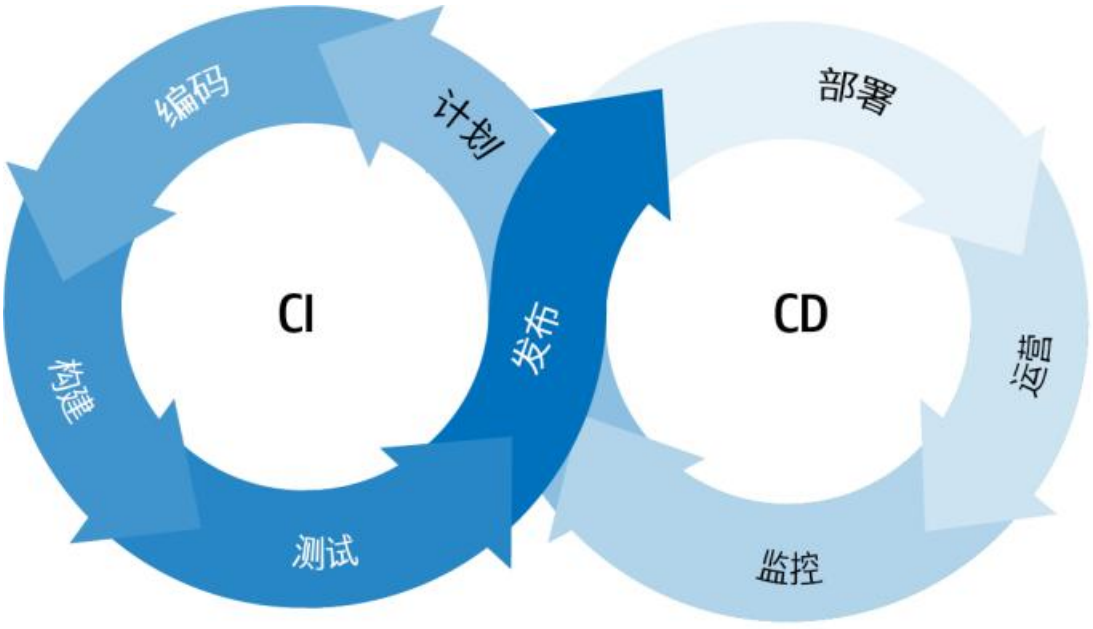


图 1-1. DevOps CI/CD 流水线中的软件开发阶段

持续集成意味着在开发后尽快将代码变更合并到共享的主代码库中，使代码库与每个人的最新工作保持同步，以便可以尽快进行测试和发布。持续部署使发布的过程以及生产部署的测试能够自动化运行。通过不断合并和测试变更，DevOps 流水线降低了独立工作的分布式团队发生集成冲突的风险。

从本质上讲，DevOps 旨在打破阻碍质量持续提升的障碍。通过相关步骤左移到

开发流水线的早期阶段，测试和反馈循环变得更快，这意味着开发人员开始更多地承担测试自己代码的职责。 DevOps 在很大程度上依赖自动化来降低每一步的风险，同时改进编码、测试和部署变更的工作流程。

DevOps 带来了更高效的开发和部署，同时质量更高、故障更少、恢复时间更短。但是 DevOps 有一个没有明确解决的问题，那就是安全性。越来越快的压力致使团队很难优先考虑安全性，这也被视为开发过程中的障碍。

在传统的瀑布开发模型中，安全性通常是在最后添加的。图 1-2 展示了瀑布模型，其中涉及安全性的方面通常发生在验证和维护步骤中。

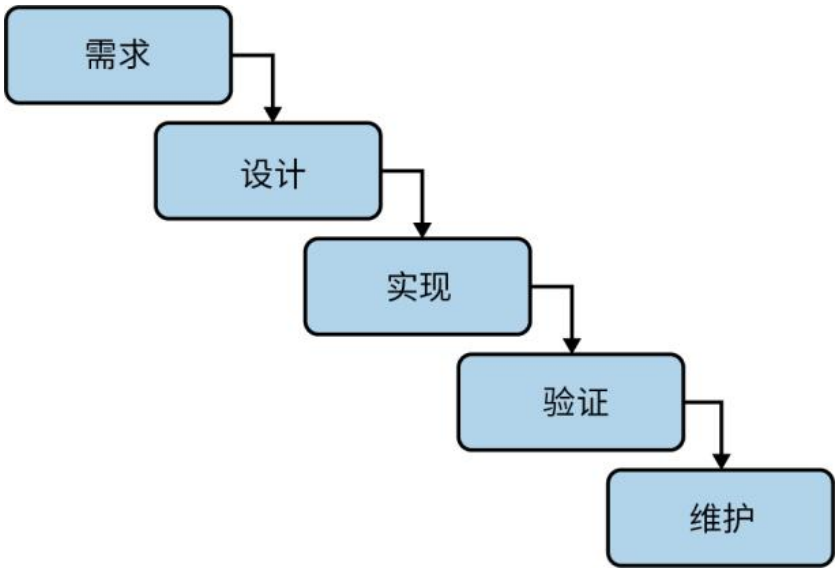


图 1-2. 软件开发的传统瀑布模型

这种安全方法在传统的单体软件开发过程中并不十分有效，对于基于现代微服务的应用来说就更糟糕了。因此出现了一种新的思维方式，将安全性集成到整个开发过程中：从 DevOps 演变为 DevSecOps。

DevSecOps

当开发和安全性分开时，自然会产生摩擦。开发人员可能会将自身视为变革的驱动力，而将安全视为持续的障碍。众所周知，安全团队会因执行审计或调查问题而叫停开发。同时，开发人员却一次又一次地制造或忽略相同的问题，而不采用明确的解决方案。

DevSecOps 将安全性作为 DevOps 的优先事项，将其置于应用程序开发的中心环节，并且对软件流水线中的每个人建立安全第一的理念。在 DevSecOps 模型中，安全是每个人的职责。这也是减少开发人员和安全工程师之间产生摩擦的第一步。

与 DevOps 一样，DevSecOps 专注于打破团队之间的重重阻碍，使沟通变得更

透明化，目标是从一开始就在每个产品中构建安全性，在开发过程中尽可能地保持原有的速度和敏捷性。正如 DevOps 赋予更多开发人员测试自己代码的职责一样，DevSecOps 将构建安全且合规的代码视为每个开发人员的首要任务。

DevSecOps 将安全实践构建到应用程序开发生命周期的每个阶段，并在每个步骤提供反馈。安全性通常甚至在设计阶段之前就开始了，以培训的形式帮助开发人员学习安全的编码实践。安全团队与开发人员一起工作，帮助培训他们，记录安全策略和最佳实践，并指导每个人接纳安全思维方式。随着组织的 DevSecOps 实践的成熟，不同的团队开始将自己视为单一文化的一部分，并将安全视为核心目标。

DevOps 和 DevSecOps 如何改变团队

虽然安全性曾经被视为开发结束时的附加功能，但 DevSecOps 使人们意识到应用程序中的每个组件都有可能受到攻击，并且必须在设计上保证绝对的安全。通过为软件开发过程带来灵活性和透明度，DevSecOps 便转向于可在任何环境中部署的云优先软件。随着越来越多的公司认识到需要将安全性纳入软件开发的核心，DevSecOps 逐渐成为主流，DevSecOps 已经被证实是 DevOps 的自然演变，就像 DevOps 自然地从敏捷方法中成长出来一样。

DevSecOps 所代表的理念转变意义重大。安全和 DevOps 团队的工作有了一个共同的目标：快速、安全地将高质量的产品推向市场。开发团队也将不会感觉到总是被安全流程阻断其工作，安全团队也将发现自己不再需要重复解决相同的问题，这使得团队能够保持强大的安全态势，能够发现并且防止漏洞、配置错误或者违反合规性策略的行为等。开发、运营和安全团队共同进行威胁建模，共享知识，籍此预测或者消除产品在其开发过程中涉及的潜在缺陷。

除了态度的转变之外，DevSecOps 还提供了切实的好处。通过尽早地发现问题，团队可以为其客户提供更好、更安全的产品和服务。当紧急补丁或意外漏洞存在较少时，最终用户体验也会更好。DevSecOps 可以更早地发现漏洞并且在部署之前就修复它们，这样也可以减少客户的宕机时间，从而使企业更具成本效益。

采纳 DevSecOps 的挑战

采纳 DevSecOps 实践具有非常明显的优势，但这并不意味着其中的每一步都很容易，从保护分布式应用程序的日常机制到重大的理念变革，DevSecOps 都会为团队带来潜在的挑战。

传统意义来说，安全性侧重于易于理解的应用程序边界，通常围绕在单个数据中

心之间。随着企业采用 DevOps、云原生以及基于微服务的应用架构，应用程序和安全性之间的挑战也将会分散开来。这些应用程序由在多个环境中运行的微服务组成，跨多个网域进行通信，并且处理来自全球的设备和用户数据，这就使得这些服务之间的攻击面又大又难定义，几乎不可能盘点这些服务之间的所有交互或者是通过公共和专用网络传输的所有数据。

与此同时，当应用程序的所有权从团队转移到更多的部门时，安全性很容易被抛在一边。如果单个工程团队认为安全是专门的安全团队的问题，那么他们就不会重视安全性。这往往会将安全性推向软件开发过程的后期阶段，此时安全性也将会变得更加困难且效率极低。

只有当开发团队充分了解安全时，左移才有效。开发人员不仅需要良好的安全开发实践经验，而且还需要足够的知识来理解他们负责解决的问题，这也就意味着需要花费时间和金钱来进行培训。

解决这些问题的关键是创建一种协作的理念，以安全为焦点建立快速、持续的迭代过程。这意味着之前许多独立分散的团队需要学习如何在一起协作：进行开发、IT 运营和保持安全。安全性必须成为整个软件开发过程中各个方面的一种实践，并且必须保持持续，且不能中断。开发团队必须提供指导并且指导过程也不能中断，就像流水线的开发和运营部分一样保持连续性。

好消息是收益大于成本，当每个人的工作涉及到安全时，这些问题就能以花费更少的费用更早的得到解决，并且也为参与软件开发过程中的每个人以及客户提供更好的用户体验。

第 2 章 安全左移

随着现代软件架构和软件供应链变得更加复杂，安全威胁也不断演变。对那些高关注度的攻击分析表明，漏洞可能出现在软件开发生命周期 (SDLC) 的任何时刻。2020 年，威胁行为者获得了一家名为 SolarWinds 的软件公司的构建系统的访问权限，恶意修改了软件更新，然后分发给客户。同年，攻击者破坏了名为 Codecov 的代码扫描工具的上传脚本，使他们能够访问客户计算机上的环境变量。此类攻击利用公司的自身的供应链来分发恶意软件，使威胁行为者能够访问归属于公司客户的基础设施。事后检测攻击不再是一个可行的解决方案。只有通过在整个 SDLC 中集成安全性，组织才能保护自己及其客户。

DevSecOps：安全左移

传统上，安全和其他测试放在软件开发过程的最后阶段，即设计和构建阶段之后。仅在流程结束时才进行安全测试，决定软件是否适合发布的结果。将安全左移意味着通过设计将安全嵌入到整个 SDLC 之中。

在传统的瀑布式开发中，软件遵循从设计到部署的线性路径。相比之下，DevOps 软件环境中的持续集成和部署意味着一切都在随时发生。为了理解现代 SDLC 的安全左移理念，我们必须暂时展开 CI/CD 流水线的无限符号，如图 2-1 所示。

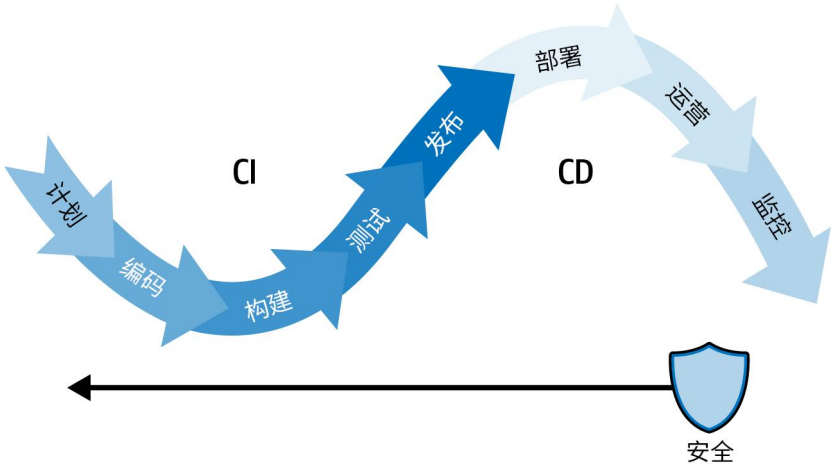


图 2-1：通过将安全嵌入到 CI/CD 流水线的各个阶段来实现安全左移安全

安全左移需要在整个软件供应链中采取强有力的安全措施，从计划直到已部署软件的运营和监控。将安全视为应用程序设计整体的一个组成部分可帮助团队尽早发现并修复潜在问题。据埃森哲称，将安全左移可以将构建成本降低 70%，并将上线时间

缩短一半。及早发现和修复问题可以增强应用程序的质量和安全性，避免事后设计和应用缺陷解决方案，并减轻运营团队的应用程序维护负担。

左移不仅仅是为了更早地发现漏洞。 DevSecOps 是一种安全第一的文化转变，旨在通过设计预防问题，同时还假设漏洞将始终存在——由 Google Cloud 的 DevOps Research 和 Assessment 团队（DORA）制作的 《加速 DevOps 状态状态报告 2021》，称之为诊断方法。为了快速自动评估安全性，DevSecOps 使用多种工具作为 CI/CD 流水线的一部分，包括以下工具：

- *静态应用安全测试 (SAST)*

  扫描源代码以查找缺陷，向开发人员提供早期反馈。

- *容器镜像扫描工具*

  通过自动扫描容器以识别漏洞。

- *动态应用安全测试 (DAST)*

  在运行时扫描应用程序以检测通过扫描代码难以显现的问题。

- *运行时应用程序自我保护 (RASP)*

  分析应用程序的运行状态并阻止可疑活动。

- *Web 应用程序防火墙 (WAF)*

  监控应用程序级网络流量以检测和阻止攻击。

特别是，WAF 可以提供针对于应用程序的保护。从历史上看，安全和网络运维团队使用 WAF 来保护在集中式数据中心生产环境中运行的应用程序。在现代的云原生基础设施环境中，这种方法不够快也不够高效。等待应用程序投入生产、标记 WAF 识别的问题并在下一版本的代码中修复它们不再有意义。现代开发方法是将 WAF 纳入构建和测试阶段的早期阶段，并在出现问题时予以解决。这意味着 WAF 本身必须能够左移。

可以根据应用程序的端点、API 和数据来配置 WAF，从而更轻松地检测异常情况，同时保留合法流量。与仅在协议级别处理流量的策略相比，这提供了更细粒度的安全性。 WAF 还可以了解使用该应用程序的用户和设备以及他们需要哪些权限，从而可以自动执行策略，例如为每个客户端提供最低的必要权限。通过这种方式，WAF 可以高效、准确地进一步左移安全性，从而为正在开发和部署的应用程序提供量身定制的保护。反过来，这可以开发出更强大的应用程序，并可以部署在更广泛的环境中。使用某些 WAF，您可以将安全策略封装为代码，从而可以构建更强大的策略，并

可以使用源代码管理来开发和演进。

安全左移的挑战

在任何组织中，安全左移都涉及技术和文化方面的挑战。不同的团队有不同的角色。开发人员努力快速前进，为他们的项目增加创新元素，安全团队不断努力降低风险，即使这意味着放慢速度。这可能会导致摩擦或冲突。为了缓解这些挑战，CI/CD 流水线需要采用新的流程和工具，团队必须学会以新的方式使用现有工具。所有相关团队必须努力实现必要的一致性、可见性和透明度，以便让安全性成为 SDLC 新的一部分。

在一些组织中，不同的团队在如何自我组织和工作方面有很大的灵活性，但它可能使团队难以一致地左移安全性。很难对整个组织内可能未统一开发的应用程序和服务提供整体视图。可见性是安全的关键，尤其是在漏洞检测和合规性领域。如果整个 CI/CD 流水线没有一致的可见性，就很难以有意义的方式左移安全。随着团队的成熟，他们倾向于与组织范围内的最佳实践保持一致，但当左移时，您会迫不及待地希望这种情况发生。如果项目不遵循最佳实践，就不可能实现良好的安全性。每个团队都必须具有良好的测试覆盖率和对其他团队流程的高度可见性，以便开发人员可以安全地共同创新。

如果公司文化对于问责过于正式，那么安全团队之外的人员可能很难在 SDLC 早期承担额外的安全测试责任。当出现问题时，害怕受到指责可能会导致人们逃避责任，因为他们认为这可能会导致不良后果。这个问题可以通过自动化工具在一定程度上缓解，但对于参与应用程序开发和运维的每个人来说，主动承担安全责任仍然很重要。

左移的一个陷阱是未能尽早与安全团队进行强有力的接触。安全团队必须准备好与更广泛的群体互动，其中许多人可能在安全方面没有太多专业知识。开发人员并不总是习惯于将安全视为开发过程的一部分。如果没有安全团队的指导，开发人员并不总是对潜在的安全风险有深刻的认知。这通常需要培训和指导，这给安全团队带来了额外的责任负担，并且是安全人员与组织中其他人之间潜在的摩擦点。

如果安全性没有得到增长的预算，团队可能会理所当然地认为他们被要求用更少的钱做更多的事情。更糟糕的是，如果开发团队没有为这些活动做好充分准备，安全团队可能会当他们试图帮助教育其他人如何通过设计构建安全性时，遇到阻力。安全团队必须配备足够的人员、做好准备，并准备好从第一天起就与开发和运维团队合作。敏捷方法和 DevOps 提高了软件开发和部署的速度，要求公司在不牺牲可靠性和

安全性的情况下加快行动速度。这给安全团队带来了沉重的负担，并使安全性成为限制快速迭代开发周期的瓶颈。安全性不仅负责保护正在开发的应用程序，还负责保护开发工具。

随着安全左移，安全团队熟悉的一些工具仍然有用。其他不是为自动化和集成到现代软件开发基础设施而设计的历史遗留工具，很难在左移策略中使用。解决这些问题需要时间。这可能会导致人们一直认为安全团队是快速、高效的开发流程的障碍，从而破坏了安全团队试图与开发和运维团队建立起好的关系。

为什么左移很重要

客户与服务或应用程序的每次交互都代表着公司与客户之间的信任是加强还是被破坏。随着威胁行为者变得更加聪明和迅速，稳定且可信的互动变得尤为重要。在大型组织中，存在大量潜在的漏洞，其中任何一个漏洞都可能对组织构成生存威胁。

虽然正在开发的应用程序和服务是最重要的安全目标，但软件开发越来越依赖第三方开源库和软件包，这些库和软件包本身可能包含漏洞。如果不能发现这些缺陷，它们就可能会在整个软件流水线中一直传播到构建和部署阶段，从而容易使应用程序或开发环境本身受到攻击。在开发结束时将安全性归咎于测试过程将使整个流水线中存在漏洞。此外，工具本身可以呈现攻击面。在大型组织中，安装的工具数量可能相当大，并且可能包括废弃或孤立的工具，其唯一剩下的作用就是等待其漏洞被利用。

将安全左移有助于安全跟上 DevOps 软件开发模型的步伐，同时管理新兴的漏洞和风险。将安全左移将其从烦人的后续附加转变为设计约束，从而降低了修复和预防漏洞的成本和复杂性。

变革策略

将安全左移的好处是显而易见的：与在瀑布式开发过程的尾声进行安全测试相比，更早地发现问题更有效，成本也低得多。然而，为了使其发挥作用，所有相关团队都必须改变对安全的思考方式、彼此之间的互动方式以及对出现的问题的响应方式。这种文化变革不是安全左移的先决条件，也不是安全左移的结果，而是一个并存的条件。文化变革推动左移，反之亦然。您可以采取以下几个步骤来开始旅程。

定义你的策略

第一步是了解安全左移对您的组织意味着什么。您必须清楚地了解成功是什么样子，以便您的团队知道目标是什么。这意味着定义角色和所有权、沿途的里程碑、衡量进展的方法以及清晰的愿景。为了实现成功所需的文化变革，每个参与者都需要与

共同目标保持一致。例如，这意味着了解当前的威胁形势以及安全左移有何帮助。

只有让每个人都成为利益相关者，您的组织才能完成文化变革最重要的方面：让安全成为每个人的责任。安全团队可以帮助其他团队从计划阶段开始，在编写一行代码之前，在整个 CI/CD 流水线中构建持续的安全性。程序员在实现和测试每个组件时必须首先考虑安全性。良好的安全实践必须成为每个人的第二天性。

了解供应链

了解组织中的软件供应链可能比看起来更困难。通常，不同的业务部门使用非常不同的软件开发流程和工具。除非组织已经步调一致，否则很难理解每个团队在其构建中包含哪些依赖项、哪些工具很重要、哪些工具已被放弃，以及代码如何从各个开发人员的笔记本电脑流向 CI/CD 流水线。将安全左移意味着将安全融入到软件供应链的各个方面。

开发人员必须在开发的每个阶段更加意识到安全风险。选择包和库作为依赖项、规划应用程序架构、部署环境、编写函数和运行测试都是具有安全影响和安全需求的过程。

开发人员必须负责评估和管理这些风险，并继续积累知识，从而使他们的工作更加安全。

提供护栏

不仅开发人员必须适应新的安全思维方式。安全团队也必须从停下工作来排除故障或者解决问题的守门思维转变为建立护栏，以帮助开发在更少问题的情况下继续行进。通过构建新的策略、工具和建议，安全团队可以帮助开发人员更安全地前进。通过自动化和自助服务，可以有助于开发人员自助实现安全性。安全工程师可以使用他们对可能发生的攻击的广泛知识，帮助开发人员更有效地构建安全性。

自动化和反馈非常重要，因为它们使开发人员能够适应并不断前进。测试自动化工具、代码和容器扫描工具以及自动化集成工具减轻了开发人员的一些负担，同时为他们提供了在开发过程早期修复问题所需的信息，并且避免将漏洞引入最终产品。拥有能够提供足够自动化反馈的工具非常重要，从而使安全、开发和运维团队能够不断调整。

使培训持续进行

当安全成为开发人员的新责任时，他们意识到创建安全代码必须采用的实践，这时会有一个陡峭的学习曲线。第一步是评估开发团队的安全知识，并帮助教育每个人

安全编码实践。安全之旅永无止境。在学习如何减轻威胁和漏洞方面，是没有“完成”的时候的。开发人员必须持续学习作为其工作的一部分。

当安全团队从一开始就教导其他团队如何将安全性构建到产品中时，他们也在学习，发现新的威胁类型和缓解威胁的新方法。随着安全团队不断地学习新知识，他们必须与开发团队共享这些信息，从而帮助每个人的安全实践不断得到发展。

第 3 章 应用安全

随着软件架构的变化，应用安全在过去几年中不断发展。应用安全的主要目标保持不变：减少漏洞和其他威胁。如今的软件开发生命周期更快、迭代性更强、分布性更强。随着团队在地理位置上分散，并且应用组件被分离为云基础架构上的微服务，网络已成为一个易受攻击面。随着保护应用程序的复杂性不断增长，加快前进的压力也持续不断。软件开发的这些新现实就需要新的应用安全方法。

安全领域正在如何变化

在传统的单体软件开发时代，安全主要处于边缘位置。一旦建立了安全边界，保护它就相对直接了。计划、编码和测试应用程序是连续的阶段，通常只在最后阶段实施安全。

在现代软件开发流程中，一切都变得更加迅速。开发阶段是迭代的、并行的和连续的。基础设施本身，曾经代表的是需要计划、订购、交付和安装的物理硬件，现在变成了虚拟且可替代的。提供新硬件的过程几乎是即时的，这意味着组织可以根据需求的变化快速进行扩展和收缩。单体架构已经让位于微服务集合，这些微服务集合是在每天发布多次的流水线中开发的。所有这些因素加起来，在各个层面上带来了快速而持续的变化。

云基础设施呈现出更广泛、更复杂的攻击面。仅在几年前，我们还可以合理地假设基础设施和网络边界是稳定和明确定义的，形成了一个可防御的边界。但在云或混合环境中，这种情况不再适用。这些边界变得模糊，存在许多可以随时改变的访问和入口点。几乎任何资源都可以通过少数配置更改变为公开或私有的，并且敏感数据经常通过公共网络传输。容器化将单体应用程序转变为网络上可用的大型服务集合，每个服务都有自己的边界。

DevOps 正在将安全引入一个新模式，注重透明度、即时性、敏捷性和持续集成。安全左移在应用程序、容器、微服务和 API 的流程中构建和使用安全工具和控制，依靠自动化来保持一致性并跟上快速的开发步伐。这种“安全即代码”的方法，类似基础设施即代码，必须使用声明性策略来维护所需的安全状态，不妨碍创新。

威胁

由于应用程序和服务严重依赖分布式基础设施，网络本身已成为关键的攻击点。第 7 层（应用层）是特别令人感兴趣的目标。第 7 层互连服务和系统之间的复杂通信

使其难以确保安全。讽刺的是，诸如 HTTPS 和传输层安全 (TLS) 等安全传输协议实际上可以帮助攻击者隐藏有效负载，使某些类型的攻击更难以检测。

拒绝服务攻击

攻击应用层的新领域是一种古老的威胁：拒绝服务 (DoS) 攻击。第 7 层攻击相对简单且成本低廉，因为所需的只是能够向应用程序发出 HTTPS 请求或 API 调用。传统的防御策略在设计上使应用程序层暴露在这些攻击之下，因为它们是合法客户端使用服务或应用程序的机制。

现代 DoS 攻击通常采用以下几种形式之一：

- *GET 和 POST 攻击*

  攻击者通过发送大量请求使服务器超负荷运行。

- *慢速攻击*

  攻击者通过缓慢消耗资源来降低服务器的速度。

- *TLS 攻击*

  攻击者通过发送无效消息占用 TLS 服务器资源。

在 GET 和 POST 攻击中，攻击者发送大量请求，导致服务无法响应真实的用户，通常使用一组被劫持的服务器网络或与互联网连接的设备来进行攻击。在慢速攻击中，攻击者发送合法的请求，以便服务保持连接，但随后减慢速度，占用服务器资源。例如，在 Slowloris 攻击中，攻击者会缓慢地发送部分请求标头，使服务在等待每个请求头的剩余部分时，耗尽可用连接。在 TLS 攻击中，攻击者通过针对网络流量安全机制本身来限制整个网络的带宽。无论是在以上哪种情况下，目标都是相同的：使应用程序或服务对真实用户不可用。

DoS 攻击可能很难检测到。攻击者可以使用加密或网络地址转换 (NAT) 来掩饰其攻击来源，从而使检测或过滤恶意流量变得更加困难。由于有效负载被加密，除非在端点处进行区分，否则很难将攻击与合法流量区分开来。击败此类攻击的一种策略是速率限制，即设置允许通过的请求数量的上限。然而，与第 3 层和第 4 层分别处理网络和传输的 DoS 攻击相比，对第 7 层的攻击可能只需要较低数量的请求就可以造成严重破坏。这意味着仅靠速率限制不足以缓解这些问题。安全工具必须能够足够可靠地区分合法流量和攻击，以便攻击发生时快速识别。

攻击者变得更加老练，使用人工智能和机器学习会使他们的攻击更难以检测。基

于静态规则的安全性无法跟上攻击策略的变化以及应用架构的变化，这些变化使新兴攻击成为可能。检测针对应用程序的攻击需要对应用程序本身的行为有深入的了解，建立基线以帮助确定哪些流量是合法的。这意味着每个应用程序、服务和端点都必须单独受到保护，并使用针对该应用程序或服务 API 的定制工具。

其他威胁

尽管 DoS 攻击代表了对应用程序层攻击面的简单利用，但也可能存在更隐蔽的威胁。在许多情况下，这些攻击会利用未能保护数据或代码的情况。例如，身份验证和访问控制攻击通常通过修改会话标识符或请求的其他部分、利用弱密码或密码恢复过程来实现，或者在暴力攻击的情况下，通过在一段时间后猜测有效凭据来实现。注入攻击绕过数据验证缺陷，将恶意 SQL 命令、脚本或数据插入到应用程序或服务中。

在某些情况下，威胁可能是由于意外而内部产生的。安全控制配置错误、在构建过程中使用过时或易受攻击的包作为依赖项以及不安全的设计都可能会在应用程序中打开漏洞，而没有人能够意识到这些漏洞。因此，在网络上的所有端点采用自动化工具来提供冗余安全检查非常重要。

DevSecOps 方法

DevSecOps 面临的挑战是：通过内置的安全解决方案与现代云原生应用架构的速度和复杂性保持同步，实现向左移动（即从开发过程的早期阶段开始注重安全）。工具和流程正在不减慢 DevOps 团队速度的情况下，逐渐融入安全性。从某种程度上说，转向 DevSecOps 与之前的敏捷、集成测试和持续集成的转变并没有太大差异。

安全左移有助于通过使安全成为工作流程中每个阶段的组成部分来保护应用程序，从而使工程师能够在漏洞成为问题之前发现它们。在这种模式下，开发人员首先要用安全的眼光审视应用程序、服务或功能的拟议架构。从设计的开始，重要的是要查看可用的端口和组件，要传输或暴露的信息以及如何在传输和静态存储中保护这些数据。这些问题和其他问题必须在早期就贯穿于整个开发流程。幸运的是，有一些工具和技术可以帮助简化这个过程。这些包括自动化分析工具、策略管理方法、深度防御，以及正确地使用传统的安全工具。

自动分析工具

人类的专业知识是无可替代的，但自动代码扫描和安全测试工具可以为安全问题的搜寻带来一致性、速度和持续性。除了代码审计和审查之外，自动化分析工具还有助于预防和检测代码库、依赖项以及已部署的应用程序或服务中的漏洞。

SAST 和 DAST 工具可以自动检测正在开发的应用程序和服务中的一些安全问题。 SAST 从内到外检查应用程序，扫描源代码是否存在漏洞，例如潜在注入、身份验证漏洞和其他可利用的缺陷。 SAST 发生在 CI/CD 流水线的早期，即在构建应用程序并将其部署到临时或测试环境之前。 SAST 的工作原理是分析源代码与一组编码的符合程度，旨在提供强大安全保护的规则。 DAST 与正在运行的应用程序配合使用，从外到内对其进行测试，而无需了解源代码或构建代码的框架。 DAST 的目的是采用攻击者的方法，寻找通过安全漏洞进入的途径。 DAST 作为应用程序测试的一部分运行，通常可以发现 SAST 无法发现的编码缺陷。

尽管 SAST 和 DAST 有助于保护应用程序代码库的安全，但这些工具并不能直接解决第三方库和包中的潜在漏洞。这些上游依赖项可以由构建过程直接指定，也可以由其他依赖项间接指定，这使得难以对所有包含的软件包及其可能包含的漏洞进行清单管理。软件代码分析有助于跟踪依赖关系、检测哪些组件包含已知漏洞并提供信息来帮助开发人员采取措施缓解这些漏洞。

策略管理

安全策略有助于保护部署应用程序和服务的环境。安全策略的目标是允许有效的流量、使用和请求，同时阻止对服务或应用程序的威胁、滥用和恶意误用。例如，根据 API 的预期合法使用对安全策略进行建模是有意义的，以便策略管理工具可以正确执行每个策略的目标。

通过以声明方式管理策略，直接从 API 模式获取策略，可以在安全策略中精确地表示 API 契约。当模式发生变化时，策略也会相应改变。通过这种方式，策略管理变得自动化、可与 API 模式一起进行版本控制，并在源代码控制中进行有效管理；由于策略是从 API 本身派生的，因此它实际上是以代码形式的策略。这样可以更轻松地为每个服务定义单独的策略，并使用控制面板来管理它们。

深度防御

任何单一层面的保护都可能失败，从而让攻击通过。例如，在拒绝服务（DoS）攻击期间，第一层防御是阻止来自已知由攻击者控制的 IP 地址的请求。正如我们所见，攻击者可以使用加密或 NAT 来隐藏其真实来源，或者使用劫持的设备网络同时从多个 IP 地址进行攻击。第二层防御可以采取额外的步骤，例如阻止与攻击具有相似特征的请求。这种策略将保护层逐层叠加，以便一层失效时可以由另一层进行覆盖，称为深度防御。深度防御有助于防止攻击者首次侵入系统，同时在系统内提供多层防

御，减少一旦攻击者进入系统后可以探索的范围。

左移传统工具

针对云原生架构和基础设施设计的现代安全工具是积极主动和智能化的，这导致许多人忽视了最初设计用于保护单体应用程序周围简单边界的工具。所谓的“传统”工具正在不断演进和适应新的生态系统，通过执行运行时安全策略，以新的方式用于保护集群、容器和微服务。

例如，WAF 是防御应用层攻击的强大工具。毕竟，如果一个 WAF 可以保护一个大范围，那么许多个 WAF 就可以保护许多个小范围。为了发挥作用，WAF 本身必须不断演进和向左移动。 WAF 必须轻量化、易于部署并适合定制部署，以保护应用程序、集群和软件开发流程中的不同组件。对于云原生应用程序，每个 WAF 必须了解其保护的服务的 API，最好是通过直接从源代码控制中获取模式。为了适应现代 CI/CD 流程，WAF 必须能够以声明方式自动化安全策略，以便开发人员和安全团队可以将安全性视为可配置的资源，而不是必须手动配置的内容。

第 4 章 场景：文化转变

向云原生容器化架构的迁移需要时间。许多企业仍然依赖于传统的应用程序、基础设施以及瀑布式开发。安全方面，通常用一种 "书挡"的模式堵两端：在开发之初定义安全需求，在最后生产环境上执行渗透测试，而中间并没有太多的安全措施。领导层明白这种方法无法扩展，但转向 DevSecOps 颇为复杂，需要改变不同团队之间的关系。

通常，企业会首先在开发流程中建设静态代码分析和组成分析工具。这虽然可以在流水线上构建安全机制，但却不是一个全面的解决方案。保护代码的责任仍然完全由安全团队承担。他们可以向开发人员提供反馈，但这种关系并不像真正的 DevSecOps 那样具有协作性。只有实现真正的文化转变，让每个人都落实安全性，才能有可能保证端到端的安全。其结果是能够在从设计到运行的整个流水线中部署各种工具，包括传统的安全工具。

最后一点很关键：无论开发过程中的漏洞检测有多好，你仍然需要强大的运行时安全性。即使在设计中构建了安全性，也不可能在流水线的早期捕捉到 DoS 攻击。像 WAF 这样的工具可以在整个过程中插入，提供最后一道防线和强大的反馈源，帮助团队在安全方面更好地协作。

虽然文化转变需要所有团队的合作，但往往安全团队必须率先采取行动。通过提高透明度，安全团队可以克服把关人的印象。将威胁建模和安全策略作为代码进行管理，有助于让工程师了解他们的计划和策略，从而促进合作。工程师知道应用程序是如何设计的，而安全团队则要花时间观察它在生产中的表现。自动化测试、WAF 等工具可以帮助验证应用程序的性能是否符合设计要求，从而更容易确保安全是内置的，而不是附加的。成功转变的关键之一是培养一些拥护者，让他们向其他人展示安全左移的重要性。

实例：欧洲某大型银行

面对灵活的竞争，欧洲一家大型银行需要快速构建新的基础设施，以便将其众多数据中心和员工转移到现代化的应用架构中，同时还要符合一系列金融及其相关的合规性。尽管他们已经有了防火墙和其他安全措施，但其安全团队和开发团队之间的联系仍不紧密。新应用的测试通常是将软件安装在虚拟机（VM）上，然后向安全团队发送电子邮件，要求进行测试。公司内各团队都有各自的预算，相互之间交流不多，关系也不算融洽。

采用自动化势在必行，如何实现却在内部产生了很大的阻力。安全部没有购买工具的预算，应用程序开发人员不想承担 CI/CD 流水线的负担，网络团队不想搞中央自动化进而可能影响其他所有团队。更没有人愿意为其他人承担管理工具的工作。但与此同时，却没有人愿意失去人员、权力或预算，结果就是一场内部的冷战。

执行团队明白端到端安全的价值，并赋予应用程序开发人员更多权力，让他们与安全人员并肩协作，共同构建现代应用程序，这一转变也包括了架构从基于虚拟机转向编排容器。在建立真正的 DevSecOps 流水线的过程中，他们选择了 WAF 等工具，这些工具可以通过 RESTful API 进行控制和操作，帮助他们实现安全和基础设施的自动化。最终使组织的协作性和透明度得到了提高。

实例：澳大利亚某银行

在世界各地，银行业都会受到严格监管，且竞争激烈。而客户有时会根据响应贷款申请的速度来选择银行。这种快节奏促使云原生架构的加速推广，从而方便服务更多面客渠道和应用系统。而内部面临的最大挑战是如何快速将应用程序投入生产。过去那种应用开发动辄几个月，最后才进行测试的时代已经一去不复返了。

为了减少交付阻力，团队正在向云迁移，努力将基础设施和安全作为代码来管理，从而实现发布自动化。面对这样的转变，澳大利亚一家著名银行的安全团队很快就意识到，还用把关的方式在云上行不通。在内部部署的基础设施中，安全团队拥有很大的控制权，而云上则不同，它可以让开发团队在安全团队不知情或不参与的情况下启动任何他们需要的东西。为了继续为组织提供价值，安全团队明白他们需要与其他团队密切合作。

在向云迁移的推动下，安全团队在应用程序运行环境中管理 WAF 等工具更加积极，还为流水线建立了设计和构建变更的反馈途径。安全团队帮助管理 WAF 等工具的配置及其他策略代码化，使发布变得更容易。结果是团队之间的合作更加融洽，帮助银行在瞬息万变的市场中更有效地竞争。

实例：某能源公司

在许多企业中，应用程序开发人员与其他团队之间的分歧是文化变革中的主要障碍。欧洲的一家大型能源公司正面临着更多分歧。该公司是一家公私合营企业，其目标往往是分裂的。私营企业掌握着钱袋子，通常是决策者。而对公民负责的公营部门则受到法规和官僚主义的束缚。这使得向 DevSecOps 转型尤其具有挑战性，因为安全左移涉及了很多团队之间的紧密合作。此外，决策者往往不是技术人员，有时他们

很难看到技术方案的价值。事实上，在流程开始时，公司还没有采用虚拟机，而是在裸机服务器上运行。

采用云原生 DevSecOps 的主要障碍是惰性。许多人希望保持他们认为的行之有效的现状。DevSecOps 的故事很难在公司内部推销给那些不想失去权力或预算，不了解技术，不关心 REST API 和自动化等功能的人。公司知道自己必须适应时代的发展，但官僚主义和惰性让他们原地踏步，没有变革的路线图。最终，公司遇到了现有工具的限制，不得不寻求新的解决方案。

公司开始制定需求清单，包括监控、定制和可编程性。公司内部的倡导者不得不努力帮助决策者了解技术的能力和限制。最终，推动了与其他已经采用 DevSecOps 工具的公司进行交流，包括在流水线中左移 WAF 和其他工具。归根结底，迈向 DevSecOps 并非只是让安全和其他团队合作。解决方案在于教育决策者，让他们了解为什么端到端的安全很重要，以及如何实施。

实例：某通讯公司

电信行业一直在变化。随着带宽已成为一种商品，电信公司必须在不放弃支持网络基础设施、服务交付和其他 IT 目标的同时，通过拓展附加产品来保持竞争力。向 5G 的过渡正加速迫使电信公司适应并采用云原生架构。

电信公司的文化变革有时比银行或能源公司更快，因为安全团队的规模较小。电信公司的安全团队可能有 30 人，而银行的安全团队可能是这个数字的 20 倍。在安全团队规模较小的情况下，挑战就转移到了对应用开发人员的教育上。他们可能认为安全是事后才考虑的问题，一个知识渊博的支持者可以帮助工程师理解，在没有适当控制措施的情况下发布代码会带来哪些问题。

公司将架构过渡到云原生时，在自动化流水线中构建安全性有助于减少摩擦。通过添加运行时控制和代码提交时的代码扫描，公司可以走向成熟的 DevSecOps 流水线。持续的教育可以帮助应用程序开发人员理解端到端安全是每个人的责任。

结论

安全左移对于现代云原生应用程序不仅仅是特定的技术或实践。尽管通过设计构建安全性很重要，但更需要整体方法：像威胁行为者一样思考，考虑所有可用的工具，并做出决定性的文化转变，在安全、运维和工程团队之间架起桥梁。

像威胁行为者一样思考意味着承认无论内置了多少安全性，漏洞都将始终存在。在整个设计过程中，威胁建模可以帮助每个人理解、预测和缓解漏洞，但在运行时保护和监控应用程序以及用于构建它们的工具也很重要。这在网络的应用层尤其重要。互连系统和服务之间通信的复杂性使得应用层既难以保护安全，又对威胁行为者有吸引力。

DoS 攻击等威胁无法从应用程序中设计出来，并且可能需要大量的努力来检测和缓解。当 DoS 攻击被挫败时，攻击者可以稍微改变方向以逃避安全控制，然后继续攻击。随着攻击者变得更加老道，DevSecOps 工作也必须变得更加精明。

SAST 和 DAST 工具是在测试期间捕获已知安全问题的好方法。容器扫描工具可以识别附加的漏洞。在生产中，可以定制 WAF 等工具来保护单个应用程序、服务或 API。通过这种方式，WAF 可以在不干扰合法流量的情况下检测异常和攻击，同时提供有关应用程序性能和威胁的反馈。这有助于提供更强大的策略和代码本身的优化，从而能够驱动开发更安全的应用程序，使其安全部署在各种各样的环境中。

安全左移的核心是文化变革，使每个人在整个开发过程及之后都成为安全的利益相关者。整个组织的团队必须掌控安全，改变他们响应问题的方式以及与其他团队的互动方式。让所有团队在安全愿景上保持一致可能具有挑战性，但定义成功（因素）是至关重要的第一步。每个人都需要了解威胁形势、开发和生产中的应用程序状态以及每个人在实现共同安全目标中的角色。开发人员、运维团队和安全工程师必须更加了解彼此的实践、流程和关注点。

安全团队必须愿意从把关思维转变为提供护栏的方式，帮助每个人避免出现问题。政策、建议和工具（包括有效利用 WAF 的反馈）可以帮助开发人员安全地前进。反过来，开发人员必须承担起自己的安全之旅，了解沿途的风险。从设计阶段，通过选择依赖项，通过构建和测试，安全性必须是他们的首要考虑。随着开发人员学习如何更安全地构建应用程序，以及随着威胁的发展，持续的培训是必须的。安全永远不会停止，必须像 CI/CD 流水线中的其他流程一样，持续学习。

改变组织中的文化从来都不是一件容易的事，而且每个公司的情况都不同。在某

些情况下，挑战在于让领导层了解安全为何如此重要。在其他情况下，第一步是弥合安全与开发人员之间的裂痕。没有团队愿意失去权力、预算或对自己路线图的控制。安全可能感觉像是一种强加，甚至像是一种攻击。安全拥护者可以帮助建立理解、找到摩擦源并消除摩擦。

最终，只有当每个人共同努力时，安全领域才能实现真正的左移。

关于作者

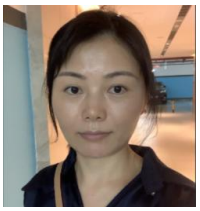
Peter Conrad 是一位技术作家，在硬件和软件环境中拥有丰富的开发经验，涵盖了从消费电子和电信到物联网和企业软件的各个领域。通过与不同领域专家进行面试和合作，他有效地向不同的观众展示技术，并作为联络人负责获取文件的审查和批准，同时也是用户的倡导者，培养了强大的人际交往能力。

关于译者（按翻译章节顺序）

胡帅，中国联通集团技术专家，中国 DevOps 社区精英译者，优秀志愿者。在 DevOps 社区进行过《MLOps 第一次亲密接触-基于 Kubeflow 的机器学习工程链入门》《从 DevOps 到 GitOps 带你构建基于 K8S 的微服务流水线》等线上技术分享，并进行过多篇文章和书籍的翻译工作。

许焱，Synopsys 软件安全高级架构师。中国 DevOps 社区核心组织者。

社区翻译贡献：10-Tips-for-DevOps-Success-in-2020 译者



余丹，Della，资深项目经理，敏捷教练，中国 DevOps 社区志愿者。

社区翻译贡献：

◆ KBP05-Terminating with Grace （一审）

◆ DevSecOps - Developer motivation vs Business Prioritisation（译者）

◆ How to Manage Linux Endpoints with Automation（译者）

◆ Configuration Strategy for Continuous Delivery of Microservices（一审）

◆ Remediation Strategy for Continuous Delivery of Microservices（一审）

冷大鲲，架构师，DevOps 产品设计，NPDP、CSM & DOM。中国 DevOps 社区核心组织者。

社区翻译贡献：

◆ 《97 Things Every SRE Should Know》译者团队

◆ 《谷歌白皮书 - DevOps 转型的投资回报率》审校团队

◆ 《产品经理和产品负责人如何（应该）有效地合作》审校

◆ 《2022 DevOps 状态报告》译者团队

◆ 《2023 DevOps 状态报告》译者团队

冀利斌，蜚语安全华北区解决方案 & 技术支持经理，DevOps 顾问，ITIL & CSM & DOM。中国 DevOps 社区志愿者。

社区翻译贡献：《97 Things Every SRE Should Know》译者团队