软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,0000,00(00):0000 0000 [doi: 10.13328/j.cnki.jos.006276]

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel: +86-10-62562563

DevSecOps:DevOps 下实现持续安全的实践探索*

戴启铭[1,2], 毛润丰[1,2], 黄 璜[1,2], 荣国平[1,2], 沈海峰[3], 邵 栋[1,2]

[1](计算机软件新技术国家重点实验室(南京大学),江苏 南京 210023)

[2](南京大学 软件学院, 江苏 南京 210093)

[3](澳大利亚凯斯琳大学彼得费伯商学院计算机系,悉尼 NSW 2060)

通讯作者: 邵栋, E-mail: dongshao@nju.edu.cn

摘 要: 国内外各大软件企业正广泛实施 DevOps 相关实践以提高产品交付和部署频率,与此同时,面对日益严峻的网络安全环境,软件系统中的安全问题日益突显.耗时的安全实践因为快速交付,在软件开发活动中难以得到有效贯彻.也正因如此,在开发和运维流程中有效集成安全控制手段,实现整个软件生命周期的持续安全,已成为各大企业向 DevOps 转型过程中亟需思考的问题.DevSecOps 作为在 DevOps 下持续解决安全问题的有效方案,也因此受到了学术界和工业界的广泛关注,并逐渐成为软件工程领域的研究重点.近年来,随着 DevSecOps 的研究和实践发展,人们对 DevSecOps 有了更全面的认识,也引入了更多安全实践.为此,本文从 DevSecOps 的背景、特征、实践、裨益和挑战五个方面进行了归纳和总结,首次向国内软件工程社区全面介绍 DevSecOps 的核心内容,重点阐述了 DevSecOps 最新的理论研究和工业界实践现状,进而为从业者实际落地 DevSecOps 提供参考,也为研究者探索 DevSecOps 提供便利,并呼吁更多的研究者参与到 DevSecOps 的研究中来.

关键词: DevOps 安全;DevSecOps;持续安全;DevSecOps 实践

中图法分类号: TP311

DevSecOps: Exploring practices of realizing continuous security in DevOps

DAI Qi-Ming[1,2], MAO Run-Feng[1,2], HUANG Huang[1,2], RONG Guo-Ping[1,2], SHEN Hai-Feng[3], SHAO Dong[1,2]

[1](State Key Laboratory for Novel Software Technology (Nanjing University), Nanjing 210023, China)

[2](Software Institute, Nanjing University, Nanjing 210093, China)

[3](Discipline of Information Technology, Peter Faber Business School, Australian Catholic University, Sydney NSW 2060)

Abstract: DevOps practices have been widely implemented by software companies to increase the frequency of product delivery and deployment. However, in the face of increasingly challenging network security, security problems in software systems are becoming prominent. Time-consuming security practices are difficult to be effectively implemented in software development activities because of rapid delivery. Integration of security control measures into software processes to realize continuous security needs to be urgently investigated for companies to transit to DevOps. DevSecOps, a solution to realize continuous security in DevOps, has attracted widespread attention fromacademia and industry, and has also gradually become a hot research topic in the field of software engineering. In recent years, as DevSecOps research and practice develop rapidly, people have gained a more comprehensive understanding of

*基金项目: 国家自然科学基金(62072227, 61802173); 国家重点研发计划(2019YFE0105500); 江苏省政府间双边创新项目(BZ2020017); 南京大学计算机软件新技术国家重点实验室创新项目(ZZKT2019B01)

Foundation item: National Natural Science Foundation of China (62072227, 61802173); National Key Research and Development Program of China (2019YFE0105500); Intergovernmental Bilateral Innovation Project of Jiangsu Province (BZ2020017); Innovation Project of State Key Laboratory for Novel Software Technology (Nanjing University) (ZZKT2019B01).

收稿时间: 2020-9-15; 修改时间: 2020-10-26; 采用时间: 2020-12-15; jos 在线出版时间: 2021-01-15

DevSecOps and more relevant security practices have been introduced. Hence, this paper summarizes the five aspects of background, characteristics, practice, benefits and challenges, with the aim to introduce the core content of DevSecOps to thesoftware engineering community in China for the first time in detail. Focusing on the latest theoretical research content of DevSecOps and the current state of corporate practice, we also aim to provide a reference for practitioners to implement DevSecOps practices. We hope this paper could provide some foundation for researchers to explore DevSecOps and call for more researchers to participate in the research of DevSecOps.
Key words: DevOps security; DevSecOps; Continuous security; DevSecOps practice

1 背景

从 21 世纪初开始,为了应对复杂多变的市场环境和频繁变更的业务需求[1],敏捷原则在软件开发中不断普及,以 Scrum[2], 极限编程(eXtreme Programming)[3],看板(KanBan)[4]为代表的敏捷开发方法被广泛应用.但主要关注开发过程的敏捷开发方法通常容易忽略软件生命周期中的其他部分.DevOps 的出现改变了这样的软件开发模式,它覆盖了开发与运维的各个阶段,成功将开发和运维紧密地集成在一起,将价值快速且持续地交付给用户[5,6].经过十余年的发展,DevOps 仍未能形成清晰明确的定义[22,26,37],但在 DevOps 工作模式下,开发人员和运维人员通过持续集成、持续部署[25]等自动化过程相互协作,共同解决软件生命周期中遇到的各种问题.有研究在调查国内 DevOps 应用现状后表明,在企业中应用 DevOps 能够显著提高 IT 服务能力[39],为企业创造出巨大的市场收益.DORA 的 2019 年度 DevOps 年度调查技术报告[32]也给出预测,未来几年 DevOps 团队数量将会继续呈现增长趋势.

DevOps 为企业带来了高频率的发布和部署.例如,Facebook 公司能够在一天内完成高达 500 次的部署上线[33],Microsoft 架构师 Igor Fesenko 也表示应用 DevOps 能够极大缩短部署所需的时间[91].但是在这种高速的开发模式下,软件安全通常难以得到保障[8,11].一旦软件中的安全漏洞被黑客发现并加以利用,企业将承受难以估量的损失.例如,2018 年,由于应用程序中存在 22 行不安全的 JavaScript 代码,英国航空公司遭受到黑客的恶意攻击,直接导致了大约 38 万消费者的个人信息和付款信息被泄露[34],造成了极其恶劣的影响.因此,在 DevOps 中引入安全实践来保障软件的安全性已成为各大软件公司的基本共识.

为了给软件安全开发提供必要的指导.早在 2006 年,McGraw[16]就列举了一些可应用于传统的瀑布开发模式下的安全实践.但不同于传统的瀑布开发模式,DevOps 下的开发节奏更快,迭代周期更短,很多耗时的安全实践将难以适用.为了在 DevOps 模式下实现保障软件安全的目标,企业亟需对传统的安全实践进行改进,或者采用全新的安全实践方法来匹配 DevOps 的开发节奏.

合理、正确地集成安全实践离不开安全团队的全力支持与配合.DevOps 成功打破了开发团队和运维团队之间的沟通壁垒,实现了价值的快速交付[38].但在 DevOps 环境下,安全团队与其他团队之间的交流却有待加强.开发和运维团队需要理解安全工作,安全团队也需要为其他团队提供必要的安全指导.为此,DevOps 需要进行相应的拓展,进一步打破安全团队与开发和运维团队之间的交流竖井,进而在企业内部形成人人对安全负责的安全文化.

在上述背景下,2012 年 Gartner 研究员 Neil MacDonald 提出了安全 DevOps 的相关概念,即 DevSecOps.他指出了 DevOps 在安全方面存在缺失[35],并提出在 DevOps 开发模式中融入与之速度相匹配的安全控制方法,来保障产品的开发过程和交付质量. DevSecOps 的出现极大地丰富了 DevOps 理论体系,为在 DevOps 开发模式下解决安全问题,提供了新的解决思路和实践方式.通过在整个软件生命周期的全阶段中集成安全实践,实现了 DevOps 开发节奏下的持续安全.2017 年 RSA 会议(https://www.rsaconference.com/usa/2017)首次为 DevSecOps 设置了专门的议题和讨论会,DevSecOps 也被提升到前所未有的高度,并在国际上掀起了对 DevSecOps 的研究浪潮.在这样的研究氛围下,DevSecOps 的理论研究和实际应用得到了迅速发展,并取得诸多成果.例如,Myrbakken 等人[10] 从定义、特征、益处和挑战四个方面初步介绍了 DevSecOps.Prates 等人[21]着重对 DevSecOps 中的度量指标进行了深入研究.

然而,国内对 DevSecOps 这一重要理论的认知基本停留在空白阶段,尽管国内已经有部分研究者注意到 De

vSecOps 这一概念[81,82],但这些研究关于 DevSecOps 理论内容的论述并不完整,易造成读者的片面理解.此外,DevSecOps 为软件工程领域带来了哪些机遇和挑战也未明确指出,更难以在实际生产中指导 DevSecOps 的落地,这些都严重制约了 DevSecOps 在国内的发展.为此,本文决定正式向国内软件工程社区全面且详细介绍 DevSecOps 这一重要概念及相关内容,期望为今后 DevSecOps 实践落地和进一步研究奠定基础.

目前工业界中的 DevSecOps 实践经验多以非公开出版的文献[41],即灰色文献的形式加以总结和分享,这些灰色文献对于理解 DevSecOps 实践现状具有重要意义,也是对学术文献研究内容的一种补充[42].因此,除了关注正式发表的学术文献之外,本文也对讨论 DevSecOps 的灰色文献进行了收集,以期反映 DevSecOps 最新的研究内容和工业界实践现状.

工业界与学术界都达成了这样一种共识,即 DevSecOps 是 DevOps 在安全领域的拓展[10].它在本质上与 DevOps 有很多相似处,它们都是作为一种理念来指导软件的开发过程,但是具体的实践方式却与实际业务场景、公司组织结构等因素息息相关.Gartner 在对业界 DevSecOps 实施现状进行调查[36]后,提出了一个 DevSecOps 的实践框架(如图 1),该框架强调了安全实践的重要地位.它通过贯穿整个 DevOps 生命周期的安全监控和日志分析来保障持续交付的过程,进而加强安全团队与开发、运维团队之间的交流和协作.除了对原有 DevOps 流程进行改进之外,DevSecOps 还要求企业在自身的组织结构和公司文化上进行一定的调整,进一步促进各个团队之间的合作.

本文组织结构如下:第 1 节介绍了 DevSecOps 的相关研究背景,指出了由 DevOps 转向 DevSecOps 的必要性.第 2 节介绍了本文资料收集的基本过程及数量分布.根据收集到的资料,第 3 节基于 CAMS 模型着重介绍了 DevSecOps 的主要特征,反映出研究者和从业者对 DevSecOps 理解上的共识.第 4 节我们详细介绍了在 DevSecOps 核心原则指导下诞生的一些典型 DevSecOps 安全实践.第 5 节我们讨论了在组织中引入 DevSecOps 带来的裨益,也阐述了引入 DevSecOps 所需要面临的挑战.第 6 节展望了 DevSecOps 未来的发展和研究方向,第 7 节对全文内容进行了总结.

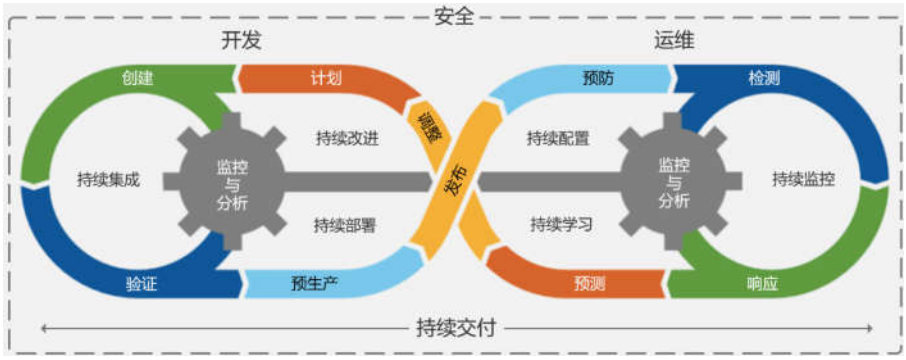

Fig.1 The DevSecOps practice framework proposed by Gartner[36]

图 1 Gartner 提出的 DevSecOps 实践框架[36]

2 学术文献与灰色文献中的 DevSecOps

为了更全面地反映 DevSecOps 在学术界和工业界的研究及应用现状,本文收集了与 DevSecOps 这一主题相关的学术文献和灰色文献进行总结分析.但由于目前国内外对 DevSecOps 的研究总体还处于起步阶段,且文献之间的质量良莠不齐,难以进行系统化的文献综述.因此,本文的重点是向国内软件工程社区介绍 DevSecOps 的基本概念以及如何进行 DevSecOps 实践,力图唤起国内从业者和研究者对 DevSecOps 更多的应用和研究热情.

本文的文献搜集及分析过程借鉴了 Garousi 等人[80]提出的研究方法,进而规范化文献的搜索及筛选过程,

整个过程如图 2 所示.在确定主题后,组内成员围绕 DevSecOps 的基本概念和相关实践讨论得到一份可执行的计划,包含确定搜索字符串、搜索范围、论文筛选、整理分析等多个部分,在整个过程中,组内成员严格遵守既定的计划,以减少在搜索和筛选过程中的主观性,如果出现意见不一致,则通过组内讨论和咨询专家的方式予以解决.

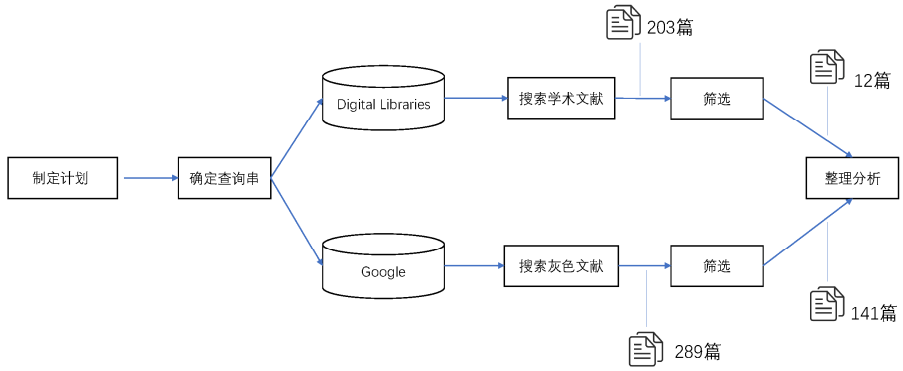

Fig.2 The Process of Literatures Collection

图 2 文献收集过程

根据制定的计划,本文首先收集相关术语来构建查询串, 然后用连接词"AND"和"OR"组成查询串分别进行学术文献和灰色文献的检索,具体查询串如下: DevSecOps OR SecDevOps OR DevOpsSec OR (DevOps AND Security) OR "Continuous Security".

本文对文献的检索过程分为学术文献搜索和灰色文献搜索两个阶段.

- 对学术文献的搜索,本文选择软件工程领域主要的四个电子数据库进行自动检索,包括 IEEE Xplore, ACM DL, SpringLink 和 Science Direct,同时将搜索范围限制在 2012 年 1 月至 2019 年 10 月之间,累计获得学术文献 203 篇.
- 对灰色文献部分的检索,本文参考 Garousi 等人[80]提出的搜索建议,使用 Google 进行灰色文献的检索,累计获得灰色文献 289 篇.

考虑到灰色文献的特殊性,对检索得到的两份不同类型的文献集,本文分别应用了不同的筛选标准,如表 1 所示,以形成最终的文献资料集合.

Table 1 Inclusion/Exclusion Criteria for academic literature and grey literature

表 1 学术文献与灰色文献的纳入/排除标准

	学术文献	灰色文献
纳入标准	● 属于软件工程 ● 能够全文查阅 ● 需要发表在经过评定的会议或者期刊上 ● 需要关注 DevOps 下的安全问题	● 用英文发表 ● 内容要与 DevSecOps 相关,包括 DevSecOps 实践经验等 ● 能够全文查阅
排除标准	● 不是用英文发表 ● 少于 6 页的短文 ● 不是一级研究 ● 重复的研究	● 广告、软文 ● 视频 ● 重复的研究

对两种文献集分别应用筛选标准之后,最终保留 12 篇学术文献和 141 篇灰色文献进行分析.我们对文献中讨论 DevSecOps 特征、实践、裨益与挑战的相关内容进行了记录,并对记录的内容应用主题分析方法加以总结

得到最终结果.每一篇文献都保证至少得到两名成员的审阅,当出现意见不一致时,第三名成员将一起评审决定是否保留.

截至 2019 年 10 月,图 3 展示了学术文献的年份分布情况.尽管 DevSecOps 这一概念于 2012 年提出,但直到 2016 年才出现相关的学术研究.从分布趋势上看,近年来对 DevSecOps 的研究稳步上升,并于近两年数量呈倍数上涨,正逐渐成为行业热门话题.

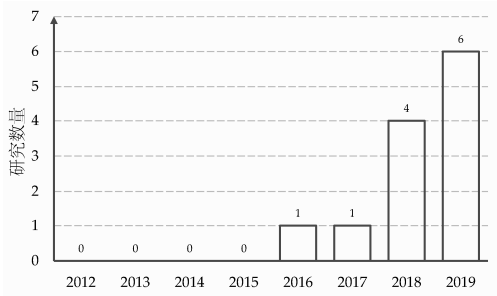

Fig.3 Year distribution of academic literatures

图 3 学术文献年份分布情况

截至 2019 年 10 月,图 4 展示了灰色文献的年份分布情况,其中有 8 篇灰色文献未能找到明确的发表时间,因此未在图中进行展示.从分布趋势上来看,除 2014 年未找到确定日期的文章外(可能是受 Google 检索策略以及存在未明确发布日期的灰色文献影响),其余年份对 DevSecOps 的研究数量逐年快速上升,这表明工业界对 DevSecOps 这一理念展开了广泛的讨论,并逐步应用于实际的工业生产中.

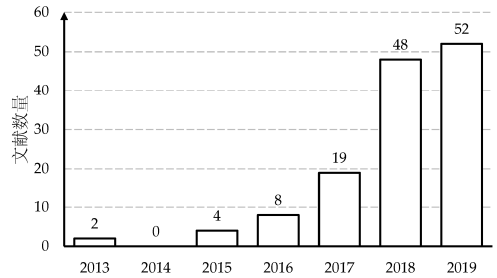

Fig.4 Year distribution of grey literatures

图 4 灰色文献年份分布情况

3 DevSecOps 主要特征

DevOps 对现代软件的开发和运维产生了深远影响[37],部分研究通过总结 DevOps 的主要特征[22,26],来反映研究者和从业者对 DevOps 这一概念的理解.其中,CAMS 模型是目前认可度较高且广泛使用的特征模型[7,23,24].它是由 Willis[45]首先提出,从文化(Culture)、自动化(Automation)、度量(Measurement)和共享(Sharing)这四个方面高度概括了 DevOps 的主要特征[23,24],为 DevOps 实践提供了概念上的指导.相似地,DevSecOps 作为 DevOps 在安全方面的必要拓展[10],它将 DevOps 延伸到安全领域[12],我们发现其特征也可以基于 CAMS 模型进行概括和拓展.本节使用 CAMS 模型详细比较了 DevSecOps 与 DevOps 特征上的主要差异(如表 2 所示),也阐述了 DevSecOps 各个特征之间的关联关系,具体解释如下.

文化(Culture): DevOps 文化试图打破开发团队和运维团队之间的沟通壁垒,让两个团队共同对最终交付的产品质量负责[20],促进两个团队之间的交流和协作[19].DevOps 强调了开发和运维两个团队之间的协作,而

DevSecOps 进一步提出安全团队也需要加入到这样的协作中.观念上,DevSecOps 指出每个人都需要对安全负责,发现并解决安全问题不再是安全人员独有的任务.所有参与软件开发和运维工作的人员都需要主动承担安全责任,并具备一定的安全意识和安全技能,尽力将软件的安全风险降至最低.在传统的开发流程中,安全团队一般位于整个工作流的末端.产品被开发完成之后,再去进行安全检测和质量把控[29].倘若在平时的开发过程中不进行安全控制,那么最后的检查通常会发现大量的安全问题,开发人员就需要花费大量的时间进行修复,这不仅会延误产品的交付时间,也会耽误新功能的开发进度.为处理好这一问题,DevSecOps 提倡将安全工作尽可能左移[10],并分散到日常的开发和运维的活动中.在组织结构上,DevSecOps 期望组织管理者能够充分意识到在 DevOps 流程中添加安全元素的重要性,在组织内部大力宣传和支持 DevSecOps 运动,积极提高安全团队在组织中的地位,让安全团队参与到整个软件交付的过程中去,进一步促进组织间各个团队的合作与交流.

Table 2 Comparison of main features of Devops and DevSecOps

表 2 Devops 与 DevSecOps 主要特征比较

	DevOps	DevSecOps
文化(Culture)	● 加强开发团队和运维团队之间的交流与协作	● 促进开发、运维、安全团队之间的沟通与交流 ● 人人对安全负责 ● 在软件开发的早期加入安全实践
自动化(Automation)	● 使用合适的自动化工具和基础设施构建 DevOps 流水线 ● 自动化重复性的、手动的工作任务	● 在 DevOps 流水线中集成自动化安全工具 ● 安全即代码
度量(Measurement)	● 检测健康状况以保障系统正常运转 ● 检测多种指标以评估和改进工作流程	● 加入安全方面的度量指标
共享(Sharing)	● 共享知识、工具与技术 ● 开发团队与运维团队之间分享想法和问题	● 在 DevOps 的共享氛围中加入安全团队 ● 安全团队帮助其他团队改进安全工作流程、提高安全意识和技能

自动化(Automation):为加快产品交付的速率,组织需要在内部建立合适的工作流程,将自动化工具集成到 DevOps 流水线中[72,73],以实现持续集成、持续部署等活动[25,26].DevSecOps 在此基础上,提出需要将安全自动化流程嵌入到软件的开发和运维中,以实现整个 DevOps 生命周期的持续安全.这样的做法能够显著提高最终交付的产品的安全质量,也能够提升组织自身的安全信心.DevSecOps 指出加入的安全实践不应当拖慢组织的开发速度,仍然需要支持组织高频率的交付行为[27],自动化地完成既定地安全任务.这样的自动化安全控制手段也被称为安全即代码(Security as Code)[7],它是 DevSecOps 强调的重点部分之一.DevSecOps 的目标是实现 100%的安全自动化控制[10],以期在没有人为干预的情况下,实现对全部安全问题的检测和反馈.

度量(Measurement):在 DevOps 中,度量作为提高流程可见性、判断方法有效性的重要手段之一,其重要性不言而喻.工作人员通过观察不同的度量指标(如代码提交频率、部署频率等),能够及时了解到系统当前的运行状态[28]和项目的整体进展,对组织工作流程中发现的问题加以总结并改进.在 DevSecOps 环境下,除了需要检测通用的 DevOps 度量指标之外,也添加了安全方面的相关度量.相关人员通过持续检测这些安全指标,及时进行信息监控和反馈,继而实现持续安全的目标.例如,DevSecOps 鼓励在开发过程中跟踪威胁和漏洞[10],并对发现的漏洞和威胁做好记录和管理[7].DevSecOps 建立的度量标准通常需要与自动化流程紧密结合,在整个软件生命周期中对指定的度量指标进行实时监控,进而评估、控制和改善整个软件开发过程.Prates[21]在研究中总结了多个 DevSecOps 的相关度量指标,但组织在实际制定度量标准时,仍然需要结合组织实际情况进行多方面的考量.

共享(Sharing): DevOps 指出开发团队和运维团队需要共享知识、工具和技术来促进团队之间的交流合作[20].同样的,DevSecOps 提倡在 DevOps 建立的共享氛围中加入安全团队[10].安全团队需要了解开发团队和运维

团队所面临的安全挑战,并帮助他们改善原有的工作流程.反之亦然,开发团队和运维团队能够了解到安全团队注重的安全问题,在早期就加以防范,进而促进不同团队之间的理解与沟通.此外,安全团队还需要为开发和运维团队统一安全工具集,并将这些自动化的安全工具集成到 DevOps 流水线中,防止因为工具不一致而导致的额外工作.公司内部也需要定期组织技术交流和技能培训活动,来帮助开发和运维人员提高自身的安全意识,能够主动去避免一些常见的安全问题.安全团队在和其他团队在交流过程中,也能够了解到他们常用的技术手段,并提出一些针对性的安全改进建议.倘若在组织内成功建立这样一种共享氛围,安全团队与其他团队之间的交流壁垒将会被进一步打破,组织的凝聚力和安全文化也会得到进一步的提升.

上述列举的四个方面的特征之间也存在着一定的联系.Tomas 等人[7]提出 DevSecOps 金字塔(如图 5 所示)来表示它们之间的关系.在一个公司内部拥有坚实的安全文化是至关重要的,安全文化位于整个 DevSecOps 金字塔的底部,它是支撑 DevSecOps 建立的基石.在安全文化的感染下,开发人员和运维人员能够主动地去学习安全技能、提高自身的安全意识,并主动承担日常开发和运维活动中的安全责任,更好地落实"人人对安全负责"的理念.同时,为了支持这种安全文化的建立,公司的组织结构也需要进行一定的调整,进而促进安全团队与其他团队之间更好的交流和协作.例如,组织需要聘请更多的安全专家,安全团队也需要为开发和运维团队进行相应的安全培训等.开发人员和运维人员在掌握了一定的安全知识之后,才可以使用既定的安全自动化工具来实现自动化安全.但 100%安全的自动化通常是难以实现的,很多时候仍需要进行必要的人工配置和检查.此外,对安全度量指标的跟踪贯穿了整个 DevSecOps 金字塔,公司需要使用不同的度量指标来反映系统的实际运行状况,并持续改善 DevSecOps 的安全工作流程.

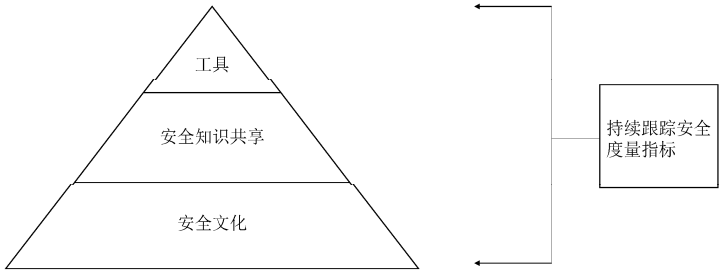

Fig.5 The pyramid of DevSecOps proposed by Tomas[7]

图 5 Tomas 提出的 DevSecOps 金字塔[7]

4 DevSecOps 典型实践

DevSecOps 核心特征揭示了 DevSecOps 的基本思想,但如何将这些原则落实到日常的开发和运维活动中,目前业界仍然缺少公认的最佳解决方案[7],大部分企业也仍处于对 DevSecOps 实践的探索阶段.现有的一些研究试图对能够在 DevOps 中集成的安全措施加以总结[8~10],但由于受访的组织数量有限、相关的学术研究数量不足、研究时间较早等多种因素影响,这些研究给出的安全实践数量相对较少且粒度较粗,对业界实际的指导意义有限.因此,本节基于现有的学术文献和灰色文献,综合了国内外企业实践探索经验,整理总结了一些常用的 DevSecOps 典型实践,试图弥补这一不足.根据我们整理得到的典型实践的适用范围,进一步将这些实践分为阶段实践和通用实践两类,接下来将分别从这两个方面对 DevSecOps 中的一些典型实践进行阐述.需要指出的是,本节我们并未罗列 DevSecOps 实践的全部内容,同一阶段内的不同实践也没有明确的先后之分.组织在具体开展 DevSecOps 相关实践时,需要综合考虑自身的组织结构、开发/生产环境、DevOps 成熟度[30]等多个方面的因素,进而更好地将这些实践应用于实际的工业生产中.

Fig.6 The flow chart of DevOps

图 6 DevOps 流程图

4.1 阶段实践

DevSecOps 实践试图在 DevOps 现有工作流中无缝接入安全相关实践,以提高 DevOps 整体流程和最终交付的安全性.DevOps 工作流及其相关实践已经在多篇研究[6,13]中进行了详细的介绍,这里便不再赘述.图 6 展示了一种典型的 DevOps 流程,它根据软件开发各个阶段主要工作内容的差异,将 DevOps 流程分为 8 个阶段.但是在实际生产中这些阶段之间并不会存在明显的界限.本文为方便展开,仍按照这八个阶段分别进行叙述,并通过各个阶段交付的制品产生联系.如图 7 所示,在本阶段中,工作人员需要对上一阶段交付的制品进行操作,从而产出新的制品并交付给下一个阶段.开发人员需要持续将自己编写完成的代码提交至指定仓库的分支中去进行自动化构建、测试等相关操作.运维阶段和监控阶段都是对生产环境中的产品进行操作,在这两个阶段之间不会有新的制品产生.一旦产品被正式部署到生产环境之中,就需要进行监控和日常维护.下面,本文将分阶段对部分典型的 DevSecOps 安全实践进行具体的阐述,并详细说明相邻阶段之间的内在联系.

4.1.1 计划阶段

计划阶段是进行软件开发前的准备阶段,在这一阶段中,开发人员需要确定需求、明确交付目标,这些需求可能由客户直接提出、是来自产品部门的新想法,或者是监控阶段反馈的安全问题.计划阶段位于 DevOps 生命周期的起始位置,倘若在该处就能够充分考虑安全性需求,尽可能多地识别出软件所面临的安全风险并加以处理,那么无疑会缩小可能的攻击面,也能够避免在开发后期发现安全问题增加的修复成本.在软件开发早期加入安全实践,对保证整个项目的安全性至关重要[14].它是"安全左移"的内在要求,也是更加注重安全团队的重要体现方式.在这个阶段,我们可以考虑以下安全实践:

安全需求工程:确定需求通常是开启项目的第一步,但在具体讨论时却通常又不会涉及软件的安全性需求[14],这就很容易导致所交付的产品或服务存在一定的安全隐患.Mead 和 Stehney 等人[15]提出使用安全需求工程来将安全性需求合并到具体的项目开发中.它包含了 9 个步骤:统一定义、识别安全目标、选择启发式技术、开发工件、引出安全性需求、对需求进行分类、执行风险评估、优化需求和需求检测.由于 DevOps 对安全需求工程提出了更高的响应速度上的要求,因此在当前的开发场景下,我们需要对安全需求工程进行一定的精简.例如,可以开发出能够直接将安全需求映射成为可运行代码的框架,进而简化安全需求工程的额外负担.

固有风险评估:系统的设计需要规避固有的安全问题,固有风险评估便是进行软件安全设计中的一项重要步骤,它需要归纳系统所面临的固有安全风险、判断这些安全风险可能带来的影响以及风险转变为现实的可能性.固有风险评估是一个需要不断重复的过程,在不同的阶段进行风险评估也有着不同的目的[16].在软件开发周期的早期进行固有风险评估主要是为了对产品进行安全加固设计,而在后期则主要为了检验安全设计的效果.

威胁建模:威胁建模是进行系统安全设计的另一个重要过程,它能够在软件开发生命周期的早期发现系统所面临的安全威胁,是一种用来识别、量化并正确应对系统所面临的威胁的结构化方法[17].使用威胁建模方法,需要正确识别出重要的数据信息、正确描述攻击者可能的攻击行为以及系统所有可能存在的威胁和漏洞[16],并将这些识别出来的威胁进行加固或者重新设计.Microsoft 公司将威胁建模作为保障 DevOps 安全的最佳实践

之一,并设计了威胁建模工具以帮助开发人员分析和识别可能的威胁,使之成为标准开发过程的一部分[48].为适应 DevOps 的迭代速度,威胁建模应当也更加高效、更加轻量级,在有限的时间内更准确地去识别出更多的安全威胁.

由于安全实践的引入,组织在整个计划阶段需要经历一个从初步讨论、设计到再讨论再设计的迭代过程.在迭代过程中,逐步完善系统的安全性需求,处理识别出来的安全威胁,从理论上保证整个系统的安全性.开发团队在得到需求之后,便开始了相应的编码工作.

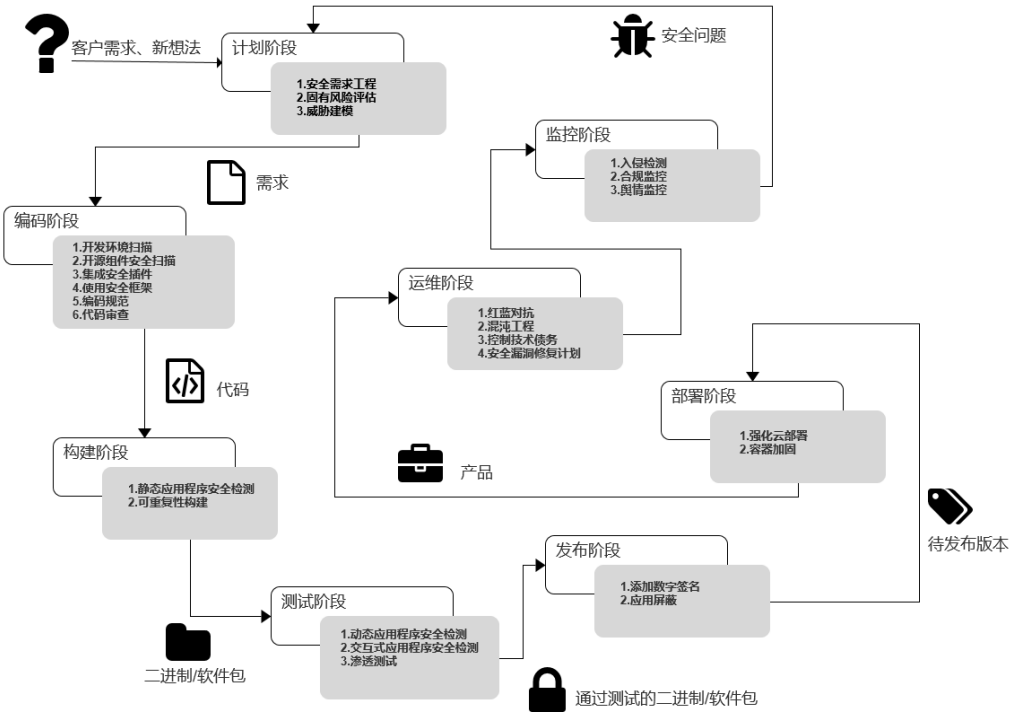

Fig.7 DevSecOps phase practices workflows

图 7 DevSecOps 阶段实践工作流

4.1.2 编码阶段

编码阶段是开发人员使用编程语言实现需求的过程.在这个过程中,开发人员需要高频率地将自己开发的代码提交至指定仓库.受需求的难易程度、开发人员的技术水平、组织的实际管理等因素影响,这些提交的代码质量通常难以得到有效保障.但是这些代码的质量却又直接影响测试和维护的过程,也对整个软件应用的安全性有着举足轻重的作用.结构良好、安全性能高的代码能够显著地减少后期的返工问题.为提高编码阶段产出的代码的质量,并加强整个开发过程中的安全规范,我们总结了以下几种具体实践.

开发环境扫描:在进行具体软件开发工作前,组织一般需要为开发人员提供开发环境.如果开发环境本身就不够安全,那么就很难保证在这样的环境中产出的代码的安全性.为了始终保障开发环境的安全可靠,英国国家网络安全中心指出需要在整个编码阶段持续对开发环境进行安全扫描[46],而对开发环境的扫描需要合适的安全自动化工具的支持.开发人员使用这些工具来识别并解决开发环境中潜在的安全隐患,保障开发环境和设备的安全性,进一步提高交付的代码的可信度.但持续对开发环境进行扫描并不意味着阻止开发人员的正常工作,组织可以在合适的地方应用安全控制技术,如身份验证、敏感点检测等方式,来为开发人员提供一个安全可靠的开发环境.

开源组件安全扫描:开发人员在开发过程中使用开源组件,能够节约大量的时间和成本,进而满足愈发快速的产品发布要求.但使用开源组件进行开发时,PhoenixNAP 公司指出目前开发者对开源软件中的缺陷的识别意识不足这一问题[85],并提示我们需要注意其可能带来的安全风险.首先,当开源组件没有得到很好的维护时,通常会存在安全漏洞.一旦这些漏洞被不法分子发现并加以利用,则会威胁到整个系统的安全状况;其次,组织在使用开源组件进行开发时,必须满足该组件许可证中规定的行为要求[43],否则组织则极有可能存在侵权行为,会面临被起诉甚至进行赔偿的风险.例如,2008 年 Jacobsen v. Katzer 案件[40]中,Katzer 在一项专利中使用 JMRI(Java Model Railroad Interface)项目中的文件,却没有遵守 Artistic License 1.0 协议.据判定,Katzer 这一行为属于侵权,需要赔偿 Jacobsen 10 万美元,并永远禁止通过下载或其他方式复制、修改或发行 JMRI 材料[40].在实际开发中使用开源组件时,我们需要选择合适的安全扫描工具对相关的开源组件进行扫描,如 FOSSID、BlackDuck 等,避免使用含有安全漏洞的开源组件或开源代码片段,使用时也需要符合其规定的行为规范.

集成安全插件:为方便开发人员在编码过程中及时发现代码中存在的安全问题,组织应当在开发人员的开发环境中安装统一的安全插件[47].这些安全插件在开发人员进行编码时开始工作,对开发人员编写的代码进行主动的安全扫描.倘若在扫描的过程中发现安全问题,便会进行提示,开发人员需要及时进行验证和修复工作.通过集成安全插件,开发人员能够在开发的早期对代码中存在的安全问题进行修复,提高进入到 DevOps 流水线中的代码的安全性.但需要保证的是,安全插件的扫描过程不会过度影响开发人员的正常开发.因此,为了与 DevOps 的速度相匹配,集成的安全插件需要由安全团队指定,这些插件需要做到足够的轻量级和较高的准确率.

使用安全框架:安全框架的内容较为宽泛,认证处理、授权管理、会话管理、加密处理、线程安全等都可以交由安全框架进行统一处理.开发人员只需要进行一定的配置,即完成对常见安全问题的管理,可以更加专注业务代码的开发.安全框架能够帮助开发人员处理常见的安全问题[47],极大地提高开发效率和编码质量,是 DevSecOps 实践中不可多得的一项实践.但安全框架的选用仍然需要综合考虑诸如具体的业务场景、框架实际性能等多方面的因素.

编码规范:编码规范涉及文件组织、编码风格、注释写法、命名规则等内容,良好的编码规范不会增加过多的工作负担,也不会降低开发人员的开发效率[48].趋于一致的代码风格反而有助于组织进行代码审查,更易于发现隐藏在高复杂度代码背后的安全漏洞,还能够提高代码的可读性和健壮性[46],进而降低二次开发和维护的成本.但适合公司的编码规范需要在实践中不断总结,仔细权衡编码效率和实际收益的关系,进而打造适应公司的编码规范.

代码审查:代码审查一般是通过人工审阅的方式来查看提交的代码是否合乎公司的编码规范,是否功能齐全,是否使用了危险函数,是否考虑到安全编码等[31],它是一项提高开发人员的规范意识、促进开发人员之间的技术交流的重要举措.Dynatrac 公司的开发人员 Plank 指出代码审查已成为公司安全开发的基本实践[86],以确保每一段代码在提交之后都能够得到别人的审查.它能够充分规避个人在编写过程中出现的疏漏,还能够使得整个团队的代码风格趋于一致并符合安全规范要求[48],进一步降低可能的安全风险.

编码阶段产出的代码是最终交付的产品或服务的核心部分,其重要程度不言而喻.如何保护整个开发过程的安全、如何提高产出的代码质量都应该是组织需要考虑的重点问题.为处理好这些问题,我们介绍了一些行之有效的安全实践,它们能够在不严重拖慢 DevOps 开发速度的情况下,对产出的代码进行严格的安全控制.编码阶段是一个持续的过程,并与构建阶段结合紧密.开发人员需要不断将自己编写的代码提交到指定的代码仓库中,以便 DevOps 流水线进行持续的集成和构建.

4.1.3 构建阶段

构建阶段的主要内容是对源代码进行集成、编译、单元测试、打包、生成可运行的二进制等自动化操作.整个构建阶段依赖 DevOps 流水线,是一个持续的过程.当开发人员将自己编写的代码提交到 DevOps 流水线中时,集成的工具就能够自动化地完成对新代码的集成、测试和打包工作.倘若在这个自动化的过程中出现问题,DevOps 流水线也会触发相应保护机制,拒绝本次集成,并将检测到的问题及其原因反馈给开发人员处理,直

至完成本次集成为止.在此基础上,DevSecOps 提倡将安全措施无缝集成到 DevOps 流水线中去.为此,我们总结了以下几种相关实践.

静态应用程序安全检测:静态应用程序安全检测(SAST, Static Application Security Testing)是提高代码安全的有效手段,也是目前较为常见的安全实践方式之一.该方法不会实际运行代码,而是通过扫描工具自动化分析源代码的词法、语法、语义、结构等多个要素,进而判断源代码是否违背既定的安全规则[18].SAST 的检测效果依赖于工具的本身性能,速度过慢、准确率过低的安全扫描工具无疑会拖慢 DevOps 的进程,也会增加开发人员验证告警的工作量.因此,为保证 DevOps 流水线的集成效率不会受过多影响,安全部门需要仔细甄别集成到 DevOps 流水线中的安全工具,这些工具应当足够的快速并拥有较高的准确性,以符合组织的期望.

可重复性构建:可重复性构建是指相同的源代码在经过多次编译后,始终输出比特位相同的二进制文件,这是组织确保源代码到二进制文件的过程中没有被篡改的重要手段之一[49].但实际操作时,这种二进制文件的一致性却难以得到保障.例如,开发人员在源代码中添加了诸如获取时间戳、使用随机数等操作时,每次编译后的输出就很难完全相同.为此,我们更需要在编码和构建这两个持续的过程中,对这些可能发生变化的因素进行控制,如配置自动化流程、避免使用不可控函数等,以实现这种一致性.

构建阶段的操作以自动化的方式运行,它与编码阶段共同形成的持续过程推动着整个项目的进展.在持续集成的过程中加入自动化的安全实践,DevOps 流水线就能够较好地对进入流水线中的内容进行验证而不浪费过多的时间,以此保障代码的安全性.完成打包的二进制或软件包会进入到测试阶段进行测试,从而进一步提高最终交付的产品质量.

4.1.4 测试阶段

尽管 DevSecOps 已经将部分测试工作提前,由开发人员负责完成,如单元测试等,但仍有部分测试需要由专业的测试人员进行.在测试阶段中,他们需要对构建完成的软件包/二进制文件进行验证工作.在一定的时间内,需要发现其中所存在的安全问题,并反馈给开发人员进行修复.在测试过程中,为保证测试的准确率,DevSecOps 期望在 DevOps 中无缝衔接安全,鼓励结合多种测试手段、使用不同的自动化测试工具来协助完成测试.企业需要充分考虑自身条件和要求,选择最合理的安全测试方案.

动态应用程序安全检测:动态应用程序安全检测(DAST, Dynamic Application Security Testing)是使用故障注入技术来识别常见的风险和漏洞的一种测试方式[52],它也是最为常见的测试方式之一.与静态检测不同,它不需要分析程序源代码的内部逻辑结构,而是为正在运行的应用程序提供数据输入,并判断程序的动态行为和输出是否符合预期,进而分析应用程序的健壮性和安全性.由于用户的输入是不可控的,因此测试人员提供的测试用例必须足够全面,并针对输入的内容做好防范和处理.

交互式应用程序安全检测:交互式应用程序安全检测(IAST, Interactive Application Security Testing)分为主动 IAST 和被动 IAST[51],它们不需要额外的配置就能够自动运行.主动 IAST 的检测功能基于外部源,该外部源能够触发应用程序中检测到的代理,这一类的 IAST 需要 DAST 工具才能激活.而在被动 IAST 中代理是独立的,并且在应用程序运行时对代码进行监视和分析,进而寻找安全漏洞.它能够与现有的测试自动化并行工作,并提供即时结果.应用 IAST 可以提供更广的代码覆盖率,可以发现更敏感的漏洞,并及时反馈检测到的结果.

渗透测试:渗透测试是从攻击者的视角出发,通过模拟恶意黑客的进攻手段对系统进行攻击,主动去发现系统中存在的安全隐患的一种测试方法[19].它能够独立地检查网络和系统的安全机制,是对常规安全扫描结果的验证和补充.除了利用自动化的渗透工具之外,渗透测试还需要安全专业人士的参与.在对工具的结果进行分析之后,通常安全人员也需要进行手动测试,但这一过程一般较为耗时.因此,在 DevSecOps 场景下,目前很难做到每一次发布前都进行渗透测试,组织需要在测试策略上进行一定的调整,例如当产品积累到足够多的新特性时,可以进行一次渗透测试.

测试阶段的安全实践集中于对二进制文件/软件包的安全检测.在产品正式发布前,通过实际运行程序来发现这些二进制文件/软件包中的漏洞和缺陷,进而完成及时的修复工作.测试方法和测试工具多种多样,为避免过度的测试影响组织发布产品的速度,组织在进行 DevSecOps 安全实践时,应当充分考虑自身实际情况,由安全

主管或安全测试专家为公司打造最合理的测试工具和测试方法的组合.在通过全面的安全测试后,这些二进制文件/软件包即可进行正式发布.

4.1.5 发布阶段

发布阶段中,版本的发布可以由相关负责人审核或者通过 DevOps 流水线自动完成.考虑到这些二进制/软件包在发布后会面临被私自篡改、逆向工程等安全威胁,组织在进行正式版本发布前,需要对这些待发布的软件产品采取不同程度的防范措施,以确保用户使用真实可靠的产品版本并维护企业的合法权益.为此,我们总结了以下相关安全实践.

添加数字签名:数字签名是进行身份验证、保障数据完整性的重要手段之一.它提供了以下四点保障:真实性、完整性、不可否认性以及特定情况下的公证效力.一般而言,组织的数字签名无法被冒充,因此在发布软件包之前,都应该为其添加数字签名,以增加该软件的可信度和公信力.一旦已经添加了数字签名的软件包被发布,它就不会被轻易修改.倘若软件在发布后被私自篡改,数字签名中的消息摘要就会发生改变,用户在实际使用这些被篡改的软件时就会收到告警.正是利用这种手段,我们可以实现保障软件安全的目标.也正因如此,添加数字签名这一安全实践被广泛用于现实的产品发布流程中.

应用屏蔽:应用屏蔽技术是一项修改应用程序的二进制代码的技术,用以加强应用程序抵御入侵、篡改和逆向工程的能力[53].它利用了代码混淆技术,让应用程序中的代码变得混乱和复杂,使应用程序的二进制文件更加难以分析,进而增强了应用程序的防护能力.此外,应用屏蔽技术也能够保护应用软件的资产安全,它大大提高了逆向工程所需的工作量,能够有效阻止盗版、数字信息盗窃等情况的出现,从而让企业在同类产品的竞争中占据一定优势.

为缩短产品的发布周期,让用户尽早享受到新的产品和功能,发布阶段的相关安全实践也应当以自动化的方式进行.在保证产品质量和组织利益的前提下,通过对产品添加额外的安全保护,保证高质量产品的发布过程.在完成产品发布后,组织将会在实际生产环境中部署上线该版本的产品.

4.1.6 部署阶段

部署阶段会将待发布的版本部署到生产环境中,并让其持续对外提供服务.DevOps 提倡的是持续部署,它需要以自动化的方式简化整个部署过程并完成上线.因此,对即将部署上线的产品和基础设施的安全保护也应当集成到这个自动化的过程中,这些实践能够切实消除运维人员的操作风险,实现对交付的产品的进一步加固.为此,我们总结了以下几点实践.

强化云部署:由于 DevOps 与云技术结合得十分紧密,运维人员在多云环境中进行安全部署时更需要考虑上线后的诸多问题.比如,在确保系统安全的情况下,如何保障计算资源的高可用、如何实现相对均衡的负载、如何对存储资源进行合理分配等.为解决好上述问题,在对基础设施中的众多组件进行安全配置时[16],有时需要依赖运维人员的个人经验,但这无疑会增加基础设施的安全风险.强化云部署将通过验证的部署准则和经过测试的配置集成到一套框架中,并依赖自动安全配置检测机制来加快部署进程,避免发生错误的部署[50].这大大简化了运维人员的操作,也能够强化对基础设施的集中式管理.

容器加固:容器技术的出现极大地方便了软件的开发和部署,对容器进行安全加固,是 DevSecOps 中必要的一项安全措施[63],Taylor 也强调了在企业在转向 DevSecOps 过程中保障容器安全的重要性[87].容器技术具有轻量级、灵活性高、低成本等优势[54],但由于容器本身的权限较大,将容器部署到基础设施中时,必须对它采取安全加固措施,以减小潜在的攻击面.常见的加固措施相对较多,可以通过容器原生的安全功能设置访问控制组、清除不必要的 root 权限等方式进行加固,也可以使用外部工具,如 Bench Security 脚本,对容器安全进行测评和必要的修补,还可以对容器镜像进行安全扫描以保证镜像源的完整性等.

DevOps 强调自动化软件部署过程,但这一过程仍离不开运维人员的参与.运维人员需要根据组织的基础设施、可能的部署方式等实际情况,制定最佳的部署方案,完成线上的部署工作.当待发布版本的部署上线后,产品便运行在生产环境下对外提供服务,组织就开始了对它的日常运维工作.

4.1.7 运维阶段

在日常的运维过程中,线上的产品时常会受到来自外部或内部的恶意攻击,从而丧失正常提供服务的能力或造成大量信息泄露等安全问题,可能给企业造成巨大的负面影响.为保障线上产品的稳定和安全,运维团队应当与安全团队紧密配合,在实际的生产中去检验产品和服务的安全性能,积极提高组织处理突发安全问题的能力.为此,我们总结了以下几种典型实践.

红蓝对抗:红蓝对抗是一种在生产环境下模拟网络持续攻防场景,提高企业的安全防御能力的一项实践[10].它能够在了解企业实际安全状况的基础上,针对企业重要的核心业务模拟真实的网络攻防,从而揭示系统实际存在的脆弱环节,帮助业务提升安全能力.红蓝对抗的范围较广,涉及外网安全、内网安全、数据库安全等多个方面,并与系统的实际业务场景紧密结合.在持续的对抗中,组织内部也需要不同团队之间的互相沟通、配合,共同抵御来自内部或者外部的网络攻击.

混沌工程:混沌工程是一项在生产环境中对软件系统进行试验的方法,其目的是为了建立系统能够成功抵抗意外情况的信心[55].该方法允许在生产环境下随机关闭服务、模拟网络故障等方式来测试系统的安全性和弹性恢复能力,进而保证系统在部分节点故障的情况下,仍然能够提供高质量的服务.在具体实践时,我们可以选择使用Netflix开发的Simian Army工具集中的部分工具,来协助我们对基础设施的可靠性和安全性进行弹性测试,进而提高组织自身的安全信心.

控制技术债务:技术债务是指在进行软件开发时,开发团队从短期效应的角度选择了易于实现的解决方案而导致的额外的返工成本.技术债务的不断积累,会严重阻碍系统长期的可扩展性和安全性,并使得公司在未来需要花费更多的时间来偿还这些债务.譬如,如果企业不注意对原有的技术进行维护和升级,那么当旧技术本身存在重大的安全漏洞且难以修复时,执意使用原有组件无疑会给企业带来严重的安全威胁.为此,我们就不得不通过重构或者其他方式来偿还之前所有的技术债务. DXC公司指出了技术债务可能带来的不利影响,并将控制技术债务的实践应用于 DevSecOps 中[88].正因如此,我们需要控制技术债务的回报高于债务本身[56],在前期可以选择通过技术债务来快速的扩张,但我们也必须考虑定时偿还这些债务,不让它成为阻碍公司进步的障碍.

安全漏洞修复计划:组织内应该具备这样一个流程,它用于支持安全漏洞的验证并采取相应的补救措施,从而持续保护整个应用系统的安全.组织在验证漏洞报告中注明的潜在漏洞的真实性时,需要根据波及的用户数量和高危程度等因素综合确定这些漏洞修复的优先级.针对不同等级的漏洞确定具体的不同修补方案,努力减少漏洞的影响范围,直至彻底解决这些漏洞为止.在产品发布前甚至发布后,我们都需要不断地重复上述过程.通过具体而全面的安全漏洞修复计划,来大幅提高组织处理安全威胁的效率和能力,同时这也是判断DevSecOps在组织内执行效果的重要表现之一.

很多意料之外的安全问题只有在实际的生产环境中才会被暴露出来.当这些问题被暴露、组织受到实际的恶意攻击时,在 DevOps 运维过程中加入安全实践便能够很好地进行安全防卫,这对于促进组织中各个团队进行有效的交流合作、提升组织自身的安全防卫能力、改善具体的安全流程具有着重要意义.此外,运维阶段与监控阶段通常密不可分,维护和监控的对象都是生产环境中的实际产品,它们通过具体行为来产生相互联系.为此,组织常常也会在日常运维过程中加入有效的安全监控手段来进行安全保障活动.

4.1.8 监控阶段

在生产环境中,软件服务以及开发、运维中的每个环节都应当实施有效的安全监控.一旦发现问题,就需要及时进行反馈和处理.根据反馈的问题的严重程度及其具体原因的不同,运维人员也需要采取不同的应对措施,以形成一个完整的安全修复过程.DevSecOps 鼓励在监控阶段加入自动化的安全实践并建立类似的安全反馈机制,加速安全问题的发现、定位和修复工作,从而加强日常的安全规范.为加强安全监控的有效性,我们总结了以下几点常见的安全实践.

入侵检测:入侵检测系统是一种检测网络流量以发现可疑活动,并在发现此类活动时发出警报的检测系统.它能够扫描网络或系统中违反规定的行为[57].常见的入侵检测系统分为网络入侵检测系统和基于主机的入侵检测系统.网络入侵检测系统能够检测进出网络中所有设备的流量,并对这些流量进行分析.一旦识别出恶意攻

击或者异常行为,则发出警告信息,它通常与机器学习技术相结合以提高预测的准确率.主机入侵检测则需要获取当前系统文件的快照并与先前快照进行对比,以防止关键的系统文件被修改或者删除.完善的入侵检测系统机制能够从内部和外部两个方面对系统进行保护,在识别出入侵行为后,及时提醒运维人员应当采取的防御措施,实现对系统的安全保障.

合规监控:合规监控是评估开发、运维等工作人员对国家政策和公司制度的遵守程度的一种有效方法[27],这些政策和制度包括了对实际业务构成合规风险的所有活动.在日常的开发和运维过程中,工作人员必须要遵守这些既定的规约,帮助企业在业务持续增长的基础上有效地管理风险.合规监控体现了企业对维护合规计划有效性的承诺,它应当以规约即代码(Compliance as Code)的形式[27]作为 DevSecOps 日常自动化监控不可或缺的一部分.

舆情监控:软件正式部署上线后,很多安全漏洞通常是在生产环境下被发现并进行漏洞信息分享的.因此企业需要密切关注各大网站、论坛中关于安全问题和安全动态的讨论,在未发现的安全漏洞被利用之前,尽快完成自检和修复工作.企业也应当积极加入诸如 FIRST(Forum of Incident Response and Security Teams)这样的全球安全事件响应组织,通过社区间的安全信息共享和相互协作,进一步提高自身服务的安全性能.

监控阶段是一个持续的过程,在这个过程中应用安全实践是确保产品稳定运行的必然要求.通过对提供服务的产品进行自动化的监控,运维人员可以及时了解产品的运行状态,并对发现的问题进行及时的修复或整理,进而产生新的需求并反馈到 DevOps 流程的计划阶段,开始新的一轮迭代过程.

4.2 通用实践

在 DevOps 各个阶段的早期集成安全实践是"安全左移"的重要表现形式,它能够帮助组织更早更快地发现并解决安全问题.然而 DevSecOps 的落地并不只是体现在对原有 DevOps 流程上的改进,它还要求组织在内部建立一种安全文化,使得开发、运维和安全团队能够高效地协同工作,实现快速、安全的产品交付.为实现这一目标,组织需要落地一系列相关实践来调整自身的组织结构和管理流程,由于这些实践的适用范围并不局限于软件开发生命周期中的某个特定阶段,我们将其统称为通用实践.如图 8 所示,根据实践性质的差异,我们将通用实践又进一步分为协作实践和过程实践.

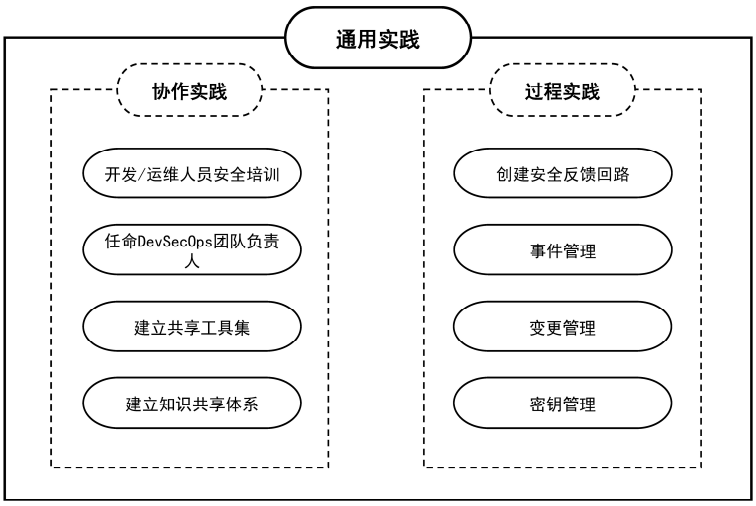

Fig.8 DevSecOps general practice

图 8 DevSecOps 通用实践

4.2.1 协作实践

协作实践旨在消除团队之间的沟通壁垒,促进团队及其成员之间的相互协作[24].它强调组织在自身的结构和文化上进行调整.实施协作实践能够帮助团队成员更好地进行经验交流、充分理解来自其他团队的问题并协同解决这些挑战,也是组织提高内部凝聚力的重要体现.我们整理总结了以下几个典型的协作实践:

开发/运维人员安全培训: DevSecOps 要求团队中的每一个成员都需要具备一定的安全意识.Microsoft[83]、Checkmarx[84]等公司将安全培训认为是 DevSecOps 实践中的重要一环,但遗憾的是,目前大多数软件公司在招收开发/运维人员时并不会将安全知识作为考察内容,因此有必要对他们进行安全方面的培训以提高安全意识[8].安全人员通过安全培训帮助其他团队成员熟悉安全编码原则、安全变量等基本安全知识,让其他团队成员能够在平时的工作中主动避免引入常见的危险操作.

任命 DevSecOps 团队负责人: DevSecOps 在软件企业中的落地通常需要多个团队的沟通与协作,在进行讨论之后共同进行决策.此外,DevSecOps 鼓励"人人对安全负责",但在具体实施时每个人的安全责任又难以具体认定,这无疑会大大降低 DevSecOps 实施的有效性.因此组织内需要任命一些 DevSecOps 负责人来具体承担安全责任,并负责推动项目的进行,这一角色的职能类似于 OWASP 提出的"安全代言人"(Security Champions)的角色[58].他们来自于各个不同的团队,帮助各自团队决定与安全团队进行合作的时机并从中进行协调[59].

建立共享工具集:在大多数情况下,开发、运维和安全人员很少使用相同的工具,这使得他们相互之间难以高效地进行协作.DevOps 通过自动化工具建立了 CI/CD 流水线,打破了开发和运维团队之间的沟通隔阂,但安全团队并不能直接通过这个方式解决沟通问题.安全团队所采用的传统安全信息和事件管理工具适用于检测异常并向安全团队发出警报,它们并不能够直接帮助开发人员增强代码的安全性.因此,安全人员需要在组织内统一合适的自动化安全工具(如 Checkmarx),并将它们集成到 DevOps 流水线中去,避免由于工具不一致带来的额外工作量,并进一步提升开发流程的效率和安全性.但由于开发和运维人员的安全知识可能较少,因此通常需要在组织内建立一个完善的安全培训和知识共享体系来共同解决这一问题.

建立知识共享体系:知识共享是消除不同部门间分歧的有效手段之一.一般而言,知识只有在被具体要求时才会被分享.DevSecOps 对知识共享的要求更多地体现在安全团队向开发和运维团队分享一些常见安全问题的预防和处理知识,同时开发和运维团队为安全团队提供产品信息以及相关的技术和设计理念,帮助安全团队理解产品功能和系统架构,进而更容易找出其中的安全隐患.

4.2.2 过程实践

过程实践旨在从一个全局的视角对整个 DevOps 流程进行安全控制,它不局限于 DevOps 流程中的某个具体阶段,是一种长期的、跨阶段的实践类型.借助 DevSecOps 过程实践,组织能够对软件开发的整个工作流程进行有效的统一管理,我们总结了以下几个典型实践:

创建安全反馈回路:安全反馈回路的建立有助于软件开发和运维团队高效地处理安全问题.在传统的软件开发流程中,安全人员往往在项目的后期才对产品的安全性进行评估,此时修复安全漏洞所需的时间和人力成本往往较高,正因如此,安全很多时候被视为项目上线的阻碍.但这一现象会在 DevSecOps 模式下有所改善,安全活动被嵌入到软件开发生命周期的各个阶段中,开发人员和运维人员可以在各自的工作过程收到安全反馈回路连续反馈的安全问题,进而进行相应的修复工作.这种不断更新的信息流可以让整个团队了解当前产品的安全状态,高效地完成必要的安全补丁和更新.

事件管理: DevOps 中提出了智能运维的理念,使用脚本化、自动化的方式来处理一些重复的常见运维问题.DevSecOps 对这一理念在安全领域进行了拓展,即对于安全事件进行标准化响应.它需要提前创建工作流、工作计划、运行脚本等,实现对安全事件发出一致的、可重复的、可度量的响应.这些脚本可以集成到 CI/CD 中,进而配合 DevSecOps 进行主动的、持续的威胁和漏洞检测,减少重大安全事故的发生.

变更管理:在代码、配置中进行更新,可能会对系统的稳定性造成一定的影响,因此良好的变更管理可以为安全人员的工作提供完备的信息支持.变更管理的范围主要包括了所有与应用程序和平台本身的版本控制相关的标准和规范[60].例如,在部署到生产环境之前,任何对代码的更改都需要通过变更管理进行审查和批准.许

多 DevOps 实践者已经使用 Git 等工具对系统版本进行控制,运维团队可以通过版本控制工具确认当前正在使用的应用程序及其代码的版本.同样的,在 DevSecOps 模式下,安全团队可以通过变更管理工具查询当前的应用程序的安全配置情况,以防证书或密钥等机密信息的泄露.当组织批准开发建议时,不同人员能够在最终的变更发布前查看工作流状态和性能指标,安全人员也能够确切地知道任何他们需要的信息[61].

密钥管理:在企业中进行诸如令牌、密钥、证书、密码等敏感信息管理和保护[60],是保障信息安全、防止企业安全威胁的有效手段之一.由于 DevOps 的敏捷性要求,很多敏感信息会对团队中的所有成员可见,这就不可避免地提高了敏感信息泄露、面临内部攻击等风险.Microsoft[83]、CloudBees[89]、IBM[90]等公司均将密钥管理纳入 DevSecOps 安全实践编排中的重要一环,以保障敏感信息的安全性.但对敏感信息的管理和保护并非易事.如果管理过于严格,会增加项目实施的复杂度,进而拖慢项目的进展;如果管理过于宽松,则可能会增加相应的安全风险.为了对敏感信息进行必要的管理,DevSecOps 可以借助集中化的敏感信息管理工具,如 Vault 等工具来对这些信息进行必要的管理和权限控制,让不同权限的账户对不同的敏感信息有不同程度的可见性[61].

5 DevSecOps 的裨益与挑战

DevSecOps 的应用会为企业带来多方面的影响,我们基于经验对整理得到的裨益和挑战进行了归纳总结,见图 9.具体阐述如下:

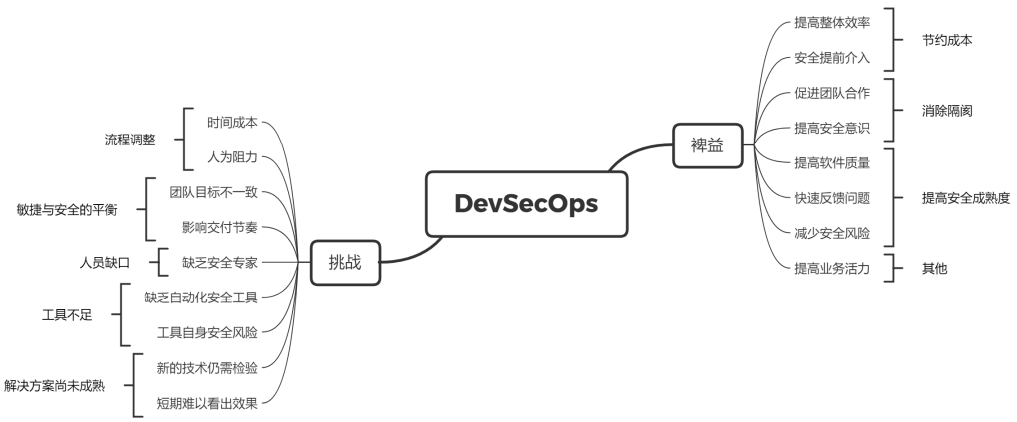

Fig.9 The map of DevSecOps benefit and challenge

图 9 DevSecOps 裨益与挑战导图

5.1 DevSecOps裨益

企业落地 DevSecOps 安全实践可以为企业带来多方面的好处.

首先,应用 DevSecOps 可以提高企业的安全开发效率,让他们有更多的时间和预算来实现安全生产的目标[10].在 DevSecOps 中,安全性被看作是 DevOps 过程的一个组成部分,增强了整个软件交付流水线的安全自动化,确保了软件发布的质量.同时,在这样的流水线中,安全漏洞也被视作软件缺陷,开发人员和测试人员都可以在早期介入安全漏洞发现和消除的过程,这种消除时机的提前可以为企业节省资金和时间成本[62].

其次,DevSecOps 将安全概念融入 DevOps 中,它提倡一种安全文化[7],成功在开发人员、运维人员和安全人员之间架起了沟通的桥梁[63],以消除不同团队之间的交流隔阂,促使达成团队之间更多、更好的合作[64].在这样的文化氛围中,更有助于提升不同团队成员的安全意识和安全能力,让他们能够主动地进行安全防范和安全控

制,而不仅仅是被动地进行漏洞的修复[10].

第三,DevSecOps 的应用可以提升软件组织的安全成熟度以及它们所开发的软件质量.DevSecOps 具有快速响应安全需求变化的能力.它可以让开发人员专注于编写高质量安全代码,使得团队可以发布更安全的应用程序.同时 DevSecOps 也支持更多类型的自动化构建、自动化安全测试和自动化安全监控[65],这些自动化的方法让软件开发的整个流程更加安全高效.

此外,应用 DevSecOps 还有其他的一些方面的好处.譬如使软件组织的业务更活跃等.在 DevSecOps 所营造的开发、安全、运维一体化的文化氛围下,各个团队紧密合作,形成支持快速反馈和修复的安全工作流.开发团队在专注于业务开发的同时兼顾考虑常见的安全问题,运维和安全团队为实现快速的业务开发迭代提供必要的支持,进一步加强公司内各个组织的活力,从而推动业务的快速、积极发展.

5.2 DevSecOps挑战

但组织应用 DevSecOps 相关实践也并非易事,它需要在改变组织内现有的流程和结构,面临着诸如组织、文化、流程与技术等多方面的挑战[10].

首先,DevSecOps 可能会改变现有的组织结构和工作流程.如果企业没有理解在 DevOps 环境中注重安全的必要性[66],其组织结构就难以发生变化.在企业转向 DevSecOps 的过程中,团队和专家也需要对自身进行调整以适应新的组织结构,他们需要掌握新的各项技能,需要集成新的工具.这些全新的内容需要组织成员花费一定的时间去熟悉和适应,无疑增加了他们的工作负担[44].

其次,如何掌握 DevOps 的敏捷性与安全性之间的平衡是一个需要解决的重要挑战.DevSecOps 的目标是打破三个不同团队之间的沟通壁垒,在公司内部营造一个协作的环境.然而,由于这些团队的核心目标不同[62],安全团队在融入 DevOps 的过程中,通常会遭到开发团队的抵制.开发人员认为安全不利于 DevOps 的敏捷性,会严重拖慢开发节奏,影响产品的交付速度[67],他们追求快速交付和高频率部署,而安全团队期望产品稳定,这常常导致两个团队之间的冲突.

第三,软件组织中缺乏 DevSecOps 安全专家.这样的角色在 DevOps 与传统安全之间架起了一座桥梁,他需要理解组织内部当前的工作流程,也需要了解如何在目前的流程中增加安全方法[59],还需要比较不同安全方法的优劣并做出最合理的判断.但可惜的是现在的软件组织中这样的角色很少见.据一份报告指出[68], 在 DevSecOps 中熟悉安全的从业人员的数量仍然很少,并预计到 2021 年将有 350 万个网络安全职位空缺.

第四,缺乏自动化的安全工具也是另一重要挑战[44].一部分组织不愿意应用 DevSecOps 的重要原因就是许多安全性监控工具的性能、自动化程度没有跟上 DevSecOps 的要求[70],这使得它们难以被集成到 DevOps 流水线中.尚未成熟、可视性较差的安全工具也难以帮助安全人员在安全方面做出明智的决策.此外,工具本身是否安全也是另一个需要考虑的事情[69],在将自动化安全工具集成到 DevSecOps 流水线中时,这是无法忽视的重要问题.

第五,DevSecOps 解决方案尚未成熟,它还需要更多的实践探索.组织成功转型到 DevSecOps 也可能需要一定的时间,并在短期内也难以看到其巨大优势.在这种情况下,一些小型企业不愿意尝试转向 DevSecOps[70].此外,DevOps 是云的支持者,使用容器是一个普遍现象,这使得安全团队需要有更多的安全方面的考虑.新的安全实践必须适应这样新的环境和新的技术,并与特定的安全准则保持一致[71].

6 未来展望

从上述介绍的 DevSecOps 特征和相关实践现状来看,目前对 DevSecOps 的研究和应用还处于起步阶段,仍然有诸多问题亟待解决,也需要进一步的发展和完善.基于本文内容,本节将 DevSecOps 未来可能的研究方向归纳如下:

(1) 组织结构和文化转型

组织在实际进行 DevSecOps 转型过程中不得不考虑如何调整现有的组织结构,如何进行团队划分,如何接纳安全文化,如何促进团队之间的交流等诸多问题.现有的 DevOps 研究已经在指导组织的转型方面有所进展

[74,75],协助组织解决在 DevOps 转型过程中遇到的各种挑战,但 DevSecOps 在这一方面还有所欠缺.尽管部分研究指出了组织在 DevSecOps 转型过程中确实存在管理层对安全文化重视不够、安全责任难以落实等挑战[7],但遗憾的是,现有的研究却鲜有在这方面的深入讨论,难以帮助组织科学合理地完成 DevSecOps 转型,这在一定程度上制约了 DevSecOps 的发展和应用.因此,产生一份科学合理的解决方案和指导办法,来为组织进行 DevSecOps 转型提供理论基础,值得研究者们展开更深层次的讨论和探索.

(2) 自动化安全工具改进

DevSecOps 实践离不开自动化安全工具的支持.在 DevSecOps 流水线中集成合适的自动化安全工具能够显著减少人工干预,让整个项目以自动化的方式向前推进,实现对整个开发和运维过程的安全保障[10],而这些集成的安全工具自身的性能和安全性对 DevSecOps 实践的效果有着举足轻重的作用.例如,使用误报率低、漏报率低的静态代码分析工具进行代码扫描,能够显著减少误报和漏报的数量[76],安全人员不需要在无意义的误报上浪费太多的时间和精力,而是专注于解决真正的安全问题.因此,持续改进自动化安全工具的性能也将会作为未来的发展趋势之一,进一步受到关注.

(3) DevSecOps 流水线管理

DevSecOps 流水线是组织进行自动化安全实践的基础.在这样一条流水线中应当集成哪些安全工具,这些安全工具需要怎样编排,如何保障流水线的安全运转等都是组织进行 DevSecOps 实践时需要考虑的问题.部分 DevOps 研究提出了可以通过创建信赖机制来保护流水线[78],但融入了安全元素之后,DevSecOps 流水线的管理更加复杂,在这样的新环境下,如何评估 DevSecOps 流水线的安全性,如何提高 DevSecOps 流水线中的工作效率,如何给予 DevSecOps 流水线更全面的安全保护等相关问题都还有待展开深入讨论.

(4) 传统安全实践改进

DevSecOps 对集成的安全实践的敏捷性提出了新的要求,云技术、容器技术的广泛应用也给传统安全带来了不小的挑战,很多耗时的传统安全实践已经难以适用于追求快速交付的 DevOps 开发模式中,我们亟需对现有的传统安全实践进行改进,打造出更高效、更轻量级的安全实践方法.例如,现有的威胁建模方法相对复杂,会在一定程度上阻碍开发的进度,打造轻量级的威胁建模技术更符合 DevSecOps 的内在要求,也正逐渐引起研究者的重视[79].此外,DevSecOps 的目标是在软件开发的早期阶段发现并解决安全问题,因此,安全实践需要尽可能地左移,但具体有哪些安全实践能够左移,这些安全实践又应该具体左移到什么位置,安全实践的左移又会给软件工程带来怎样的发展等相关问题也有待深入研究.

(5) DevSecOps 评价体系构建

DevSecOps 实践应当能够根据一套可量化的度量体系进行标准化评估,这些指标的变化能够客观地反映出 DevSecOps 实践的具体效果,组织也能够根据指标所反映的 DevSecOps 实践执行的具体情况采取不同的应对措施.目前已经有部分研究就这一方向展开讨论,并总结了部分可用于 DevSecOps 中的度量指标[21],但这些指标仍然需要实践的检验,也难以构建出一套完整的评价体系.尽管构建一份客观的全面的 DevSecOps 评价体系较为困难,但一旦构建成功,将进一步提高其科学性和普适性,并加速 DevSecOps 实践的实际落地.

(6) 工程师个人能力提升

工程师个人能力相关问题研究一直是软件工程领域的研究热点之一[77].随着 DevSecOps 理论和实践的不断发展,对工程师的个人能力也提出了新的要求.例如,DevSecOps 要求开发人员具备一定的安全意识和基本的安全技能,而这些是传统要求中所不需要具备的.工程师们在适应这些新要求的同时,会给 DevSecOps 的发展带来哪些机遇,这样的相互作用又会给软件工程的发展带来怎样的影响,这些问题在现有的研究中都尚未展开深入讨论.在未来,对提升工程师的个人能力等相关问题的探索也会成为一个可能的研究方向.

7 总结

DevOps 作为一种新型的软件开发和运维模式,已广泛应用于国内外的各大软件企业中.它显著提升了团队的 IT 服务能力,为企业带来了高效的发布和部署能力,让企业能够以最快的速度去完成产品的交付.但随着部署

频率越来越高、迭代周期越来越短,交付的软件质量难以得到有效保障.在这样情况下,DevSecOps 为在 DevOps 开发模式下进行安全拓展,提供了新的方法和思路,它成功将安全与 DevOps 紧密结合,兼顾了软件开发的敏捷性和安全性,发展并丰富了 DevOps 相关理论研究内容,进一步提高了 DevOps 相关理论的有效性和普适性.

本文从背景、特征、实践、裨益以及挑战五个方面系统地阐述了 DevSecOps 的理论和实践现状,弥补了国内 DevSecOps 领域的研究空白,有助于研究者了解 DevSecOps 的意义和内容,也为企业实际落地 DevSecOps 提供了理论和实践上的指导.在此基础上,本文也提出了一些 DevSecOps 未来可能的研究与实践的发展方向.具体而言,本文以 CAMS 理论模型为基础,从文化、自动化、度量、共享四个方面比对了 DevSecOps 与 DevOps 的差异,详细阐述了 DevSecOps 中增添的新内容,并比较了 DevSecOps 中这四个方面特征之间的关系.此外,本文还对能够应用于 DevOps 流程中的部分典型安全实践着重进行了总结介绍,并将这些实践分为阶段实践与通用实践两个类别,系统且全面地介绍了 DevSecOps 实践现状,在一定程度上能够指导工业界实际落地 DevSecOps.阶段实践基于 DevOps 流程图,详细总结了各个阶段之间的关系,并基于各个阶段的特点,给出了可用于各个不同阶段的具体安全保障措施;通用实践则更专注于改进企业的安全文化和安全管理,两种类型的实践相辅相成,相互促进,对改进企业实际的工作流程具有较强的实际指导意义.最后我们也总结了实际落地 DevSecOps 可能带来的益处与挑战,并展望了未来的发展趋势和研究方向.尽管现如今 DevSecOps 解决方案还不够成熟,但随着技术的发展和研究的深入,DevSecOps 也终将会克服这些阻碍因素,形成一个更具普适性的科学理论体系,广泛地应用于实际的企业生产中.

参考文献:

[1] Cohen D, Lindvall M, Costa P. An introduction to agile methods. Advances in Computers, 2004,62:1–66.

[2] Schwaber K, Beedle M. Agile Software Development with Scrum. Upper Saddle River: Prentice Hall, 2002.

[3] Beck K. Extreme Programming Explained: Embrace Change. Addison-Wesley, 2000.

[4] Ahmad MO, Markkula J, Ovio M. Kanban in software development: A systematic literature review. In: Proc. of the 39th Euromicro Conf. on Software Engineering and Advanced Applications. IEEE, 2013. 9–16.

[5] Lwakatare LE, Kuvaja P, Oivo M. Dimensions of DevOps. In: Proc. of the Int'l Conf. on Agile Software Development. Springer-Verlag, 2015. 212–217.

[6] Kim G, Humble J, Debois P, Willis J. The DevOps Handbook: How to Create World-Class Agility, Reliability, and Security in Technology Organizations, 2016.

[7] Tomas N, Li J, Huang, H. An Empirical Study on Culture, Automation, Measurement, and Sharing of DevSecOps. In 2019 International Conference on Cyber Security and Protection of Digital Services (Cyber Security). IEEE. 2019. 1-8.

[8] Rahman AAU,Williams L. Software security in DevOps: synthesizing practitioners' perceptions and practices. In 2016 IEEE/ACM International Workshop on Continuous Software Evolution and Delivery (CSED). IEEE. 2016. 70-76.

[9] Mohan V. and Othmane L.. SecDevOps: Is it a marketing buzzword? Mapping research on security in DevOps. In 2016 11th International Conference on Availability, Reliability and Security (ARES). IEEE. 2016. 542-547.

[10] Myrbakken H, Colomo-Palacios R.. DevSecOps: a multivocal literature review. In International Conference on Software Process Improvement and Capability Determination. Springer,2017. 17-29.

[11] Jaatun M.G. Software Security Activities that Support Incident Management in Secure DevOps. In Proceedings of the 13th International Conference on Availability, Reliability and Security. ACM. 2018. 1-6

[12] Leite L, Rocha C., Kon, F., Milojicic D. and Meirelles P. A Survey of DevOps Concepts and Challenges. ACM Computing Surveys (CSUR), 2019. 52(6), 1-35.

[13] Vadapalli S.. DevOps: Continuous Delivery, Integration, and Deployment with DevOps: Dive Into the Core DevOps Strategies. Packt Publishing Ltd. 2018.

[14] Yasar H, Kontostathis K. Where to Integrate Security Practices on DevOps Platform. International Journal of Secure Software Engineering (IJSSE). 2016, 7(4):39-50.

[15] Mead N R, Stehney T. Security quality requirements engineering (SQUARE) methodology. ACM SIGSOFT Software Engineering Notes, 2005, 30(4):1-7.

[16] McGraw G. Software Security: Building Security In. Addison-Wesley Professional. 2006.

[17] Shostack A.. Threat modeling: Designing for security. John Wiley & Sons. 2014.

[18] Chess B. West J.. Secure programming with static analysis. Pearson Education. 2007.

[19] Humble J. Molesky J.. Why enterprises must adopt DevOps to enable continuous delivery. Cutter IT Journal, 2011. 24(8).6-12.

[20] Fitzgerald B. Stol K J. Continuous software engineering: A roadmap and agenda. Journal of Systems and Software, 2017. 123:176-189.

[21] Prates L, Faustino J, Silva M, Pereira R. DevSecOps Metrics. EuroSymposium on Systems Analysis and Design. Springer Cham, 2019: 77-90.

[22] Smeds J, Nybom K, Porres I.. DevOps: A Definition and Perceived Adoption Impediments. International Conference on Agile Software Development. Springer Cham. 2015: 166-177.

[23] Erich F. DevOps is Simply Interaction Between Development and Operations.International Workshop on Software Engineering Aspects of Continuous Development and New Paradigms of Software Production and Deployment. Springer, Cham. 2018: 89-99.

[24] de França B, Jeronimo H. Travassos GH.. Characterizing DevOps by hearing multiple voices. In Proceedings of the 30th Brazilian Symposium on Software Engineering. ACM. 2016. 53-62.

[25] Senapathi M, Buchan J, Osman H. DevOps capabilities, practices, and challenges: Insights from a case study. In Proceedings of the 22nd International Conference on Evaluation and Assessment in Software Engineering 2018. ACM. 2018. 57-67.

[26] Jabbari R, Ali N, Petersen K, Tanveer B. What is DevOps? A systematic mapping study on definitions and practices. In Proceedings of the Scientific Workshop Proceedings of XP. ACM. 2016. 1-11.

[27] Bird J. DevOpsSec: Securing software through continuous delivery. Sebastopol. O'Reilly Media. 2016.

[28] Erich F, Amrit C, Daneva M. A qualitative study of DevOps usage in practice. Journal of Software: Evolution and Process. 2017. 29(6): e1885.

[29] Leau Y, Loo W, Tham W, Tan S. Software development life cycle AGILE vs traditional approaches. International Conference on Information and Network Technology. 2012, 37(1): 162-167.

[30] McCarthy M, Herger L, Khan S,Belgodere B. Composable DevOps: Automated Ontology Based DevOps Maturity Analysis. In 2015 IEEE International Conference on Services Computing. New York. IEEE. 2015. 600-607.

[31] Baum T, Liskin O, Niklas K, Schneider K. A Faceted Classification Scheme for Change-Based Industrial Code Review Processes. 2016 IEEE International Conference on Software Quality, Reliability and Security (QRS). Vienna. IEEE. 2016. 74-85.

[32] Forsgren N, Smith D, Humble J, Frazelle J. 2019 Accelerate State of DevOps. Technical Report. DORA & Google Cloud. 2019.

[33] Feitelson D, Frachtenberg E, Beck K. Development and Deployment at Facebook. IEEE Internet Computing. 2013, 17(4): 8-17.

[34] Klijnsma Y. Inside the Magecart Breach of British Airways: How 22 Lines of Code Claimed 380,000 Victims. 2018. https://www.riskiq.com/blog/labs/magecart-british-airways-breach/.

[35] MacDonald N. DevOps Needs to Become DevOpsSec. 2012. https://blogs.gartner.com/neil_macdonald/2012/01/17/devops-needs-to-become-devopssec/.

[36] MacDonald N. Reimagining Security and IT Resilience for a Cloud-Native DevSecOps World. Technical Report G00350812. Gartner. 2018.

[37] Hüttermann M.. DevOps for developers. Apress. 2012.

[38] Katal A, Bajoria V, Dahiya S. DevOps: Bridging the gap between Development and Operations. In 2019 3rd International Conference on Computing Methodologies and Communication (ICCMC). IEEE. 2019. 1-7.

[39] Liu BH, Zhang H, Dong LM. Survey on state of DevOps in China. Journal of Software,2019,30(10):3206-3226 (in Chinese).

[40] United States Court of Appeals. Federal Circuit. JACOBSEN v. KATZER. 2018. https://www.leagle.com/decision/infco20080813083.

[41] Auger P. Information sources in grey literature. 4th ed. Bowker Saur.2017.

[42] Salleh N, Mendes E, Grundy J. Empirical studies of pair programming for CS/SE teaching in higher education: A systematic literature review. IEEE Transactions on Software Engineering. 2010. 37(4): 509–525.

[43] Ferrante D. Software licensing models: What's out there?. IT Professional, 2006, 8(6): 24-29.

[44] Shackleford D.A DevSecOps Playbook. Technical Report. SANS Institute. 2016.

[45] Willis J. What devops means to me. 2010. https://www.chef.io/blog/2010/07/16/what-devops-means-to-me/.

[46] National Cyber Security Centre. Secure development and deployment guidance. 2018. https://www.ncsc.gov.uk/collection/developers-collection.

[47] Million T Adatia R, McCann A, Vinen N. Secure flexible plugin software architecture. U.S. Patent No. 6,742,176. 2014-05-15.

[48] Maruping M, Zhang X, Venkatesh V. Role of collective ownership and coding standards in coordinating expertise in soft ware project teams. European Journal of Information Systems. 2019,18(4): 355-371.

[49] Ratliff E. Establishing Correspondence Between an Application and its Source Code. 2016. https://www.securityweek.com/establishing-correspondence-between-application-and-its-source-code.

[50] Poolsappasit N, Dewri R, Ray I. Dynamic security risk management using bayesian attack graphs. IEEE Transactions on Dependable and Secure Computing/ 2011,9(1): 61-74.

[51] Nuseibeh R. An Introduction to IAST. 2017. https://www.checkmarx.com/2017/07/13/an-introduction-to-iast/.

[52] Mansfield-Devine S. DevOps: finding room for security. Network Security. 2018(7): 15-20.

[53] Newman H. Hacker Lexicon: What Is Application Shielding? 2019. https://www.wired.com/story/what-is-application-shielding/.

[54] Combe T, Martin A, Pietro R. To docker or not to docker: A security perspective. IEEE Cloud Computing, 2016, 3(5): 54-62.

[55] Basiri A, Behnam N, Roogi De, et al. Chaos engineering. IEEE Software. 2016, 33(3):35-41.

[56] Brown N, Cai Y, Guo Y, et al. Managing technical debt in software-reliant systems. Proceedings of the FSE/SDP worksh op on Future of software engineering research. 2010. 47-52.

[57] Liao HJ,Lin CHR, Lin YC, et al. Intrusion detection system: A comprehensive review. Journal of Network and Computer Applications. 2013,36(1): 16-24.

[58] OWASP. Security Champions. 2019. https://www.owasp.org/index.php/Security_Champions.

[59] Cardoza C. DevSecOps: Baking security into development. SDTimes.2017. https://sdtimes.com/collabnet/devsecops-baking-security-devops/.

[60] Hornbeek M. 9 Pillars of Continuous Security Best Practices. 2019. https://devops.com/9-pillars-of-continuous-security-best-practices/.

[61] Wicket J. The DevOps RoadMap for Security.. Technical Report. Signal Sciences. 2016.

[62] Chaudhry A. What is DevSecOps? 2018. https://dev.to/aditichaudhry92/what-is-devsecops-gge.

[63] Vernon M. DevSecOps: The Intersection of DevOps and Security. 2019. https://victorops.com/blog/devsecops-the-intersection-of-devops-and-security.

[64] Crouch A. DevSecOps: Incorporate Security into DevOps to Reduce Software Risk. 2017. https://www.agileconnection.com/article/devsecops-incorporate-security-devops-reduce-software-risk.

[65] Sumo Logic. The State of Modern Applications & DevSecOps in the Cloud.Technical Report. 2018.

[66] Ghosh S. Time to Move from DevOps to DevSecOps, Finds Latest CIO Survey. 2019. https://www.aithority.com/ait-featured-posts/time-to-move-from-devops-to-devsecops-finds-latest-cio-survey/.

[67] Henry J. Why Isn't Secure DevOps Being Practiced? 2018. https://securityintelligence.com/why-isnt-secure-devops-being-practiced/.

[68] Robinson M. DevSecOps: A Complete Guide to What, Why, and How. 2019. https://www.plutora.com/blog/devsecops-guide.

[69] Scalyr. DevOps Security Challenges and How to Deal with Them.2019. https://www.scalyr.com/blog/devopssec-challenges/.

[70] Drinkwater D. What is DevSecOps? Developing more secure applications. 2018. https://www.csoonline.com/article/3245748/what-is-devsecops-developing-more-secure-applications.html.

[71] Woods J. Cloud, Automation and the Future of DevSecOps. 2019. https://www.symantec.com/blogs/feature-stories/cloud-automation-and-future-devsecops.

[72] Huang H, Zhang H, Shao D. Practical impacts of automation tools in support of DevOps in China. Journal of Software, 2019,30(10):3056-3070 (in Chinese).

[73] Jin ZF, Zhang YW, YE WH, Zhang H, Shao D. Research on application of DevOps in documentation towards full value delivery. Journal of Software, 2019,30(10):3127-3147 (in Chinese).

[74] Sharma S. The DevOps Adoption Playbook: A Guide to Adopting DevOps in a Multi-Speed IT Enterprise. John Wiley & Sons, 2017.

[75] Hasselbring W, Henning S, Latte B, et al. Industrial DevOps.2019 IEEE International Conference on Software Architecture Companion (ICSA-C). IEEE, 2019: 123-126.

[76] Johnson B, Song Y, Murphy-Hill E, et al. Why don't software developers use static analysis tools to find bugs? /2013 35th International Conference on Software Engineering (ICSE). IEEE, 2013: 672-681.

[77] Rivera-Ibarra J G, Rodríguez-Jacobo J, Serrano-Vargas M A. Competency framework for software engineers. 2010 23rd IEEE Conference on Software Engineering Education and Training. IEEE, 2010: 33-40.

[78] Bass L, Holz R, Rimba P, et al. Securing a deployment pipeline[. 2015 IEEE/ACM 3rd International Workshop on Release Engineering. IEEE, 2015: 4-7.

[79] Khan R, McLaughlin K, Laverty D, et al. STRIDE-based threat modeling for cyber-physical systems. 2017 IEEE PES Innovative Smart Grid Technologies Conference Europe (ISGT-Europe). IEEE, 2017: 1-6.

[80] Garousi V, Felderer M, Mäntylä Mika V. Guidelines for including grey literature and conducting multivocal literature reviews in software engineering. Information and Software Technology, 2019, 106: 101-121..

[81] Che X. New ideas of network security from Devops to DevSecOps. Communications World, 2019(25):45-48 (in Chinese).

[82] Liu CJ. Some thoughts on enterprise security of DevSecOps. Computer & Network, 2017, 043(019):54-55 (in Chinese).

[83] Microsoft. Threat Modeling. https://www.microsoft.com/en-us/securityengineering/sdl/threatmodeling

[84] Rose M. Shifting to DevSecOps, with Software Security Testing Built In. 2019. https://www.checkmarx.com/blog/devsecops-software-security-testing.

[85] Dobran B. How DevOps Security Best Practices Delivers More Secure Software. 2019. https://phoenixnap.com/blog/devops-security-best-practices.

[86] Plank M. DevOps+Security=DevSecOps. 2017. https://www.dynatrace.com/news/blog/devops-security-devsecops/.

[87] Twain T. From Agile to DevSecOps. 2019. https://thenewstack.io/from-agile-to-devsecops/.

[88] DXC. Take a risk-based approach to DevSecOps: Embedding cyber security in application development. https://www.dxc.technology/security/insights/144315-take_a_risk_based_approach_to_devsecops_embedding_cyber_security_in_application_development.html

[89] Chicoski B. Orchestrating DevSecOps: Security at Speed. 2018. https://www.cloudbees.com/blog/orchestrating-devsecops-security-speed/.

[90] GiladMaayan. DevSecOps: Security and DevOps Working Together. 2019. https://developer.ibm.com/recipes/tutorials/devsecops-security-and-devops-working-together/.

[91] Fesenko I. DevOps in Practice. 2019. https://docs.microsoft.com/en-us/archive/blogs/uktechnet/devops-in-practice.

附中文参考文献:

[39]刘博涵,张贺,董黎明.DevOps 中国调查研究.软件学报,2019,30(10):3206-3226.

[72]黄璜,张贺,邵栋.自动化工具对中国 DevOps 实践的影响.软件学报,2019,30(10):3056-3070.

[73]金泽锋,张佑文,叶文华,张贺,邵栋.面向完整价值交付的文档 DevOps 应用研究.软件学报,2019,30(10):3127–3147.

[81]车昕. 网络安全新思路 从 DevOps 到 DevSecOps. 通信世界, 2019(25):45-48.

[82]刘长建. DevSecOps 的一些关于企业安全的思考. 计算机与网络, 2017, 043(019):54-55.