DevSecOps的六大支柱：自动化

SAFECode
Software Assurance Forum for Excellence in Code
Driving Security and Integrity

CSA GCR cloud security alliance®
GREATER CHINA REGION

CSA cloud security alliance®

DevSecOps 工作组的官方网址是：

https://cloudsecurityalliance.org/research/working-groups/devsecops/

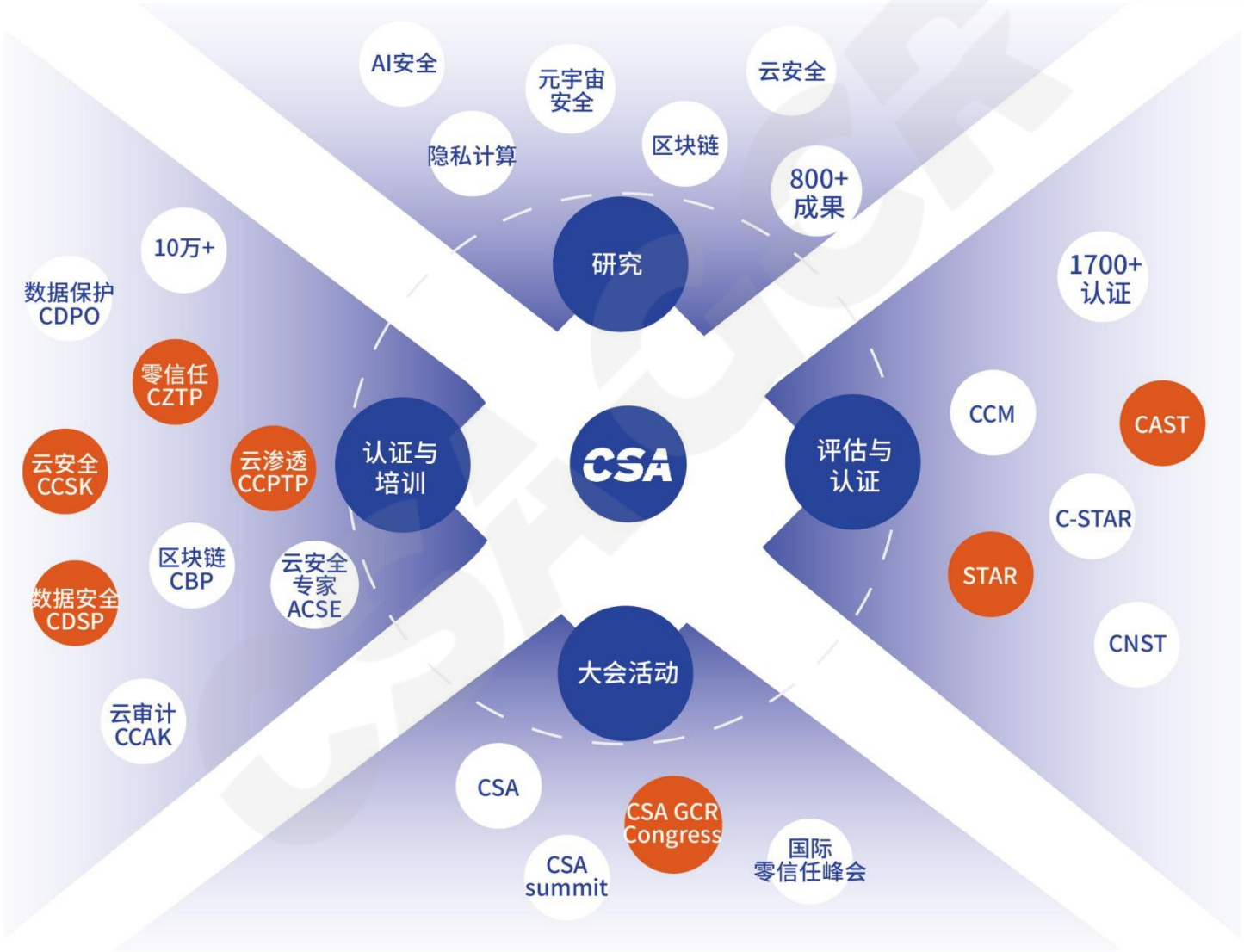
联盟简介

云安全联盟(Cloud Security Alliance,CSA)是中立、权威的全球性非营利产业组织,于2009年正式成立,致力于定义和提高业界对云计算和下一代数字技术安全最佳实践的认识,推动数字安全产业全面发展。

云安全联盟大中华区(Cloud Security Alliance Greater China Region, CSA GCR)作为CSA全球四大区之一,2016年在香港独立注册,于2021年在中国登记注册,是网络安全领域首家在中国境内注册备案的国际NGO,旨在立足中国,连接全球,推动大中华区数字安全技术标准与产业的发展及国际合作。

我们的工作

联盟会刊下载地址
了解联盟更多信息



加入我们

CSA大中华区官网
(https://c-csa.cn)

点击会员

加入联盟

填写相关申请信息

成为CSA会员

JOIN US

致谢

《DevSecOps-的六大支柱：自动化（The Six Pillars of DevSecOps: Automation ）》由 CSA 工作组专家编写，CSA 大中华区秘书处组织翻译并审校。

中文版翻译专家组（排名不分先后）：

组　长：李岩

翻译组：

伏伟任　　何国锋　　何伊圣　　贺志生　　江楠　　江泽鑫

陆琪　　余晓光

审校组：

何国锋　　江楠　　李岩

研究协调员：

卜宋博

感谢以下单位的支持与贡献：

中国电信股份有限公司研究院　　华为技术有限公司

腾讯云计算（北京）有限责任公司

英文版本编写专家

主要作者：

Souheil Moghnie　　Theodore Niedzialkowski　　Sam Sehgal

贡献者：

Michael Roza

CSA 分析师：

Sean Heide

特别感谢：

Ankur Gargi　　Raj Handa　　Manuel Ifland　　John Martin

Kamran Sadique　　Charanjeet Singh　　Altaz Valani

在此感谢以上专家。如译文有不妥当之处，敬请读者联系 CSA GCR 秘书处给予雅正！ 联系邮箱 research@c-csa.cn； 国际云安全联盟 CSA 公众号。

序言

DevSecOps 是基于 DevOps 安全敏捷化的一场变革，DevSecOps 的出现也改变了安全解决方案及安全合规的新思维。CSA 针对 DevSecOps 提出了六大支柱，分别为集体责任、培训和流程整合、实用的实施、建立合规与发展的桥梁、自动化、度量、监控、报告和行动等内容。

DevSecOps 的核心意味着要把安全管理思想全面地覆盖到整个框架中软件开发生命周期中，实现基于安全性的自动化是关键的核心问题。在 DevOps 名著《持续交付》中就指出应将几乎所有事情自动化，

本白皮书以自动化为核心，提出了 CSA DevSecOps 软件交付流水线，将安全传递给 DevOps 团队，权衡软件开发和安全需求，从而通过自动集成提高交付效率。通过学习 CSA DevSecOps 自动化的交付模式，可以帮助企业提升数字化业务的交付能力，实现基于风险的安全自动化的新方案。

DevSecOps 也是 CSA 高级云安全专家课程（CSA ACSE）的核心内容，DevSecOps 是践行共享安全责任的协作表现，DevSecOps 意味着从一开始就考虑应用程序和基础设施安全，实现自动化安全质量门禁。只有基于 DevOps 整合 IT 的合作文化，才能构建 DevSecOps 集成安全性、开发和运营实现自动化流水线，帮助企业实现其数字化转型的业务目标。

李雨航 Yale Li

CSA 大中华区主席兼研究院院长

目录

前言

云安全联盟（CSA）和 SAFECode 都致力于提高软件安全成果。2019 年 8 月发布了文章《DevSecOps 的六大支柱》提供了一组高阶方法，成功实施了作者使用的解决方案，以快速构建软件并将与安全相关的错误降至最低。这六大支柱是：

- 支柱 1：集体责任（2020.02.20 发布）
- 支柱 2：培训和流程整合
- 支柱 3：实用的实施
- 支柱 4：建立合规与发展的桥梁（2022.02.03 发布）
- 支柱 5：自动化（2022.07.06 发布）
- 支柱 6：度量、监控、报告和行动

为支持六大支柱，云安全联盟和 SAFECode[1]联合发布了一系列更详细的成功解决方案。本文是此系列出版物的第二篇。

介绍

0.1 背景

DevSecOps 自动化支柱提倡使用安全自动化来实现反思性安全。具体而言，实施安全自动化和程序化执行框架以及安全控制监控，以识别、保护、检测、响应、并从网络威胁中恢复。[2]

自动化是 DevSecOps 的一个重要组成部分，它能够提高流程效率，使开发人员、基础设施和信息安全团队专注于交付价值，而不是重复人工工作和繁复的错误。

[1] SAFECode 是一家全球性的非营利组织，定期组织商界领袖和技术专家聚集在一起，交流创建、改进和促进有效和可扩展的软件安全议题

[2] 反思性安全是一种动态的、交互式的、整体的、有效的信息安全管理策略。它代表了从现有协作概念和实践中推断出来的文化实践，并提供了一套影响组织网络安全态势的广泛含义和易于理解的原则

本文注重介绍一种基于风险的安全自动化方案，该方案在整个持续的软件开发部署周期中串联自动化安全操作。可以自动化的活动包括应用、主机和容器漏洞扫描等。

DevOps 团队灵活利用持续集成、持续交付和基础设施即代码在内的最佳实践，能够动态满足用户需求并更快速地交付。

为了将信息安全控制集成到这个过程中，自动化安全能力对于提供及时且有意义的反馈至关重要。

当今云基础设施的复杂性意味着即使微小的代码变更也可能对下游产生很大的影响。因此，安全检查需要集成到整个软件开发和部署生命周期中，贯穿设计、实施、测试和发布，并在生产中保持监控。

与反思性安全的实用的实施支柱步调一致，采用适当的自动化安全能力就能够实现快速反馈，并有可能消除整类风险。例如，容器扫描以确保加固操作系统或对已知的通用漏洞披露（CVE）进行软件组成分析。

0.2 目的

本文提供了一个框架，将自动化能够通过以下方式将安全性透明集成到软件开发生命周期中：

- 将安全相关的信息快速、互惠地传递给 DevOps 团队，通过设计连续的方式创建和验证安全代码，避免将安全问题留到交付时的单一阶段解决。
- 权衡软件开发和安全需求，从而通过自动集成提高交付效率。

0.3 读者群体

本文的目标读者包括涉及安全风险、信息安全和信息技术的管理和运营岗位工作人员。包括最高管理层（CISO、CIO、CTO、CRO、COO、CEO）尤其是涉及以下职能领域的个人：DevSecOps、自动化、DevOps、质量保证、信息安全、治理、风险管理、变更管理与合规。

1. 适用范围

本文描述了 DevSecOps 的自动化支柱：安全自动化的必要性、安全测试自动化技术以及实现机制。

本文专门提供了一个整体框架，用于促进 DevSecOps 中的安全自动化和安全控制自动化的最佳实践，并解释了有关 DevSecOps 安全测试中的常见误解。

2. 规范引用文件

部分或全部内容符合本文要求时，可在文中引用下列文件。对于注明日期的参考文献，仅版本适用的文章参考使用。对于未注明日期的引用，适用引用文件的最新版本参考使用。

- ISO/IEC 27000:2018，信息技术—安全技术—信息安全管理系统—概述和词汇；
- CSA 通过反思性安全进行信息安全管理；
- CSA DevSecOps 的六大支柱；

3. 术语和定义

出于本文档目的，ISO 27000、通过反思性安全的 CSA 信息安全管理以及以下内容中给出的术语和定义均适用。

3.1 软件组成分析（SCA）

分析具有已知漏洞的软件组件的应用或编译代码的安全测试。

注释 1：软件组成分析中的软件组件可能包括开源、库和公共代码。

注释 2：已知漏洞可通过漏洞库（如 CVE）发现。

3.2 静态应用安全测试（SAST）

分析应用源代码中软件漏洞和最佳实践差距的安全测试。

注释 1：静态分析可以在多种环境中执行，包括开放人员的 IDE、源代码和二进制文件。

注释 2：也称为“白盒测试”。

3.3 动态应用安全测试（DAST）

通过行使应用功能并根据应用行为和响应检测漏洞来分析正在运行的应用的安全测试。

注释 1：也称为“黑盒测试”。

3.4 交互式应用安全测试（IAST）

与应用一起部署的软件组件，用于评估应用行为并检测在实际测试场景中运行的应用是否存在漏洞。

3.5 运行时应用安全保护（RASP）

生产过程中目标应用里部署的安全技术，用于检测、警报和阻止攻击。

注释 1：类似于 WAF，但是安装在应用中进行检测。

3.6 WEB 应用防火墙（WAF）

通过检查 HTTP 流量来监视、警报和阻止攻击的应用防火墙。

3.7 云安全态势管理（CSPM）

能够发现、评估和解决易受攻击的云基础设施错误配置的安全技术。

3.8 云安全监控和合规

监控虚拟服务器并评估数据、应用和基础架构安全风险的安全技术。

3.9 威胁建模

识别和理解影响资源或资源集的威胁的方法。

3.10 白盒代码审查

阅读源代码以识别安全问题的人工过程。

3.11 软件交付流水线

一组用于将软件从概念交付到部署的自动化过程。

3.12 软件交付流水线（CDDP）

一种符合 DevSecOps 原则以支持安全软件交付的流水线。

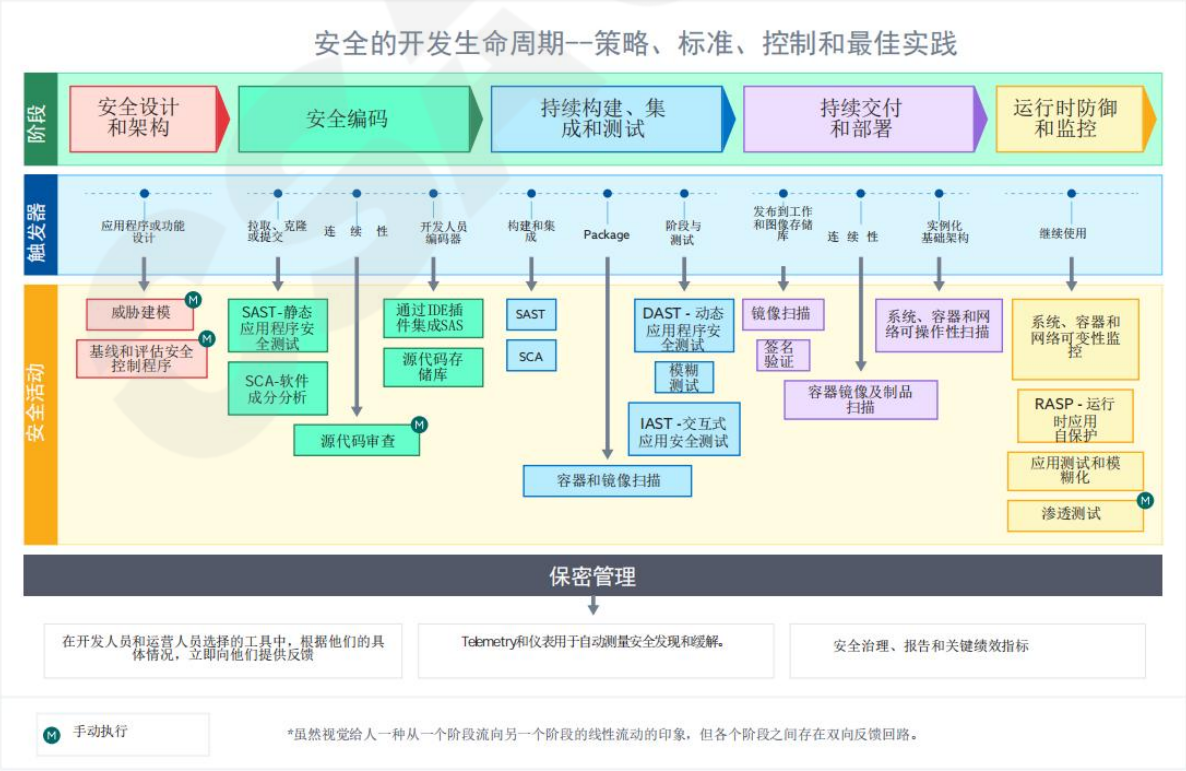
4.CSA DevSecOps 软件交付流水线

4.1 总则

在生产环境中部署之前，每个软件都要经历概念、设计、开发、构建和测试阶段。 在现代软件交付流水线中，阶段检查点的工具是高度自动化的。 在流水线的每个阶段都会执行自动检查，以防质量问题流入下一阶段。

软件交付流水线中涉及多个流程，包括软件开发、变更管理、配置管理和服务管理，可能会应用持续集成、持续交付和持续监控的概念。 开发和运维使用的工具包括源代码控制、构建、持续集成、容器化、配置管理、编排和监控。

在软件开发流水线中，集成安全测试要求安全活动完全自动化，并与交付流水线中团队已经使用的本地工具和过程充分衔接。检查点所必需的但其结果需要人工干预的活动，在本文档中称为异步测试，应特别考虑，以防止在安全自动化检查点把这些异步测试遗忘。



4.2 结构

4.2.1 总则

支持安全的软件交付流水线被认为具有三个粒度级别：

- 阶段
- 触发器和检查点
- 活动

4.2.2 阶段

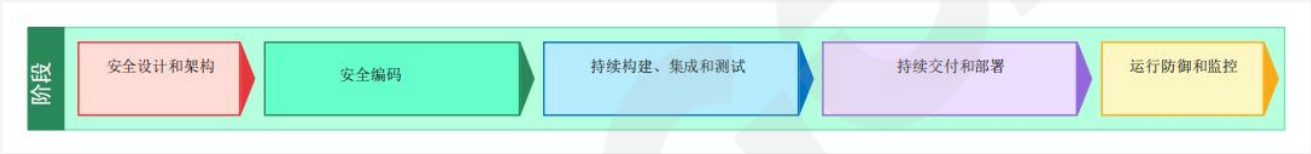


图 1-1：交付流水线中的阶段

阶段表示可交付产品的不同渐进成熟度。 在概念和架构设计阶段，软件成熟度较低。 当软件准备好持续交付和部署时，软件趋向成熟。

每个软件交付物都必须经过从设计、编码到测试的各个阶段，然后才能最终部署到生产环境中。 这与软件开发方法（瀑布或敏捷）无关。

每个阶段都有机会嵌入适当的自动化安全控制。 本文件识别了以下不同的阶段：

- 安全设计和架构
- 安全编码（开发者 IDE 和代码存储库）
- 持续构建、集成和测试
- 持续交付和部署
- 持续监控和运行时防御

4.2.3 触发器

图 1-2：交付流水线中的触发器

触发器和检查点（参见图 1-2）表示在阶段内的转换。 当满足触发条件时，将激活一个或多个安全活动，这些活动的结果决定是否满足检查点的预期要求。

如果安全活动的结果满足预期要求，则可交付成果将转移到下一个检查点（如果检查点是最后一个，则转移到下一阶段）。 否则，可交付成果将被暂停而不被允许进入到下一个检查点。

示例 1：构建新功能可以从练习构建威胁建模的开始，威胁建模可用于识别适用该功能的一般或特定安全控制。

示例 2：开发人员的代码提交或拉取/合并请求可能会自动触发源代码扫描和代码审核过程。

示例 3：将容器镜像推送到容器注册表可能会在镜像在注册表中可用之前触发漏洞扫描。

有两种类型的触发器：

- 基于变更的触发器：这些触发器由变更事件引发，例如代码推送；
- 循环触发器：这些触发由时间事件触发，例如例行计时器；

基于变更的触发器用于保护检查点的状态变更。 它们通常被用于可以以基于事件的方式运行的短期安全检查。

循环触发器则通常用于运行时间较长的安全检查，例如涉及批处理或组件之

间集成的安全检查。

示例 4：在批量代码仓中扫描误存的秘密凭据信息，循环触发器是首选。

4.2.4 活动

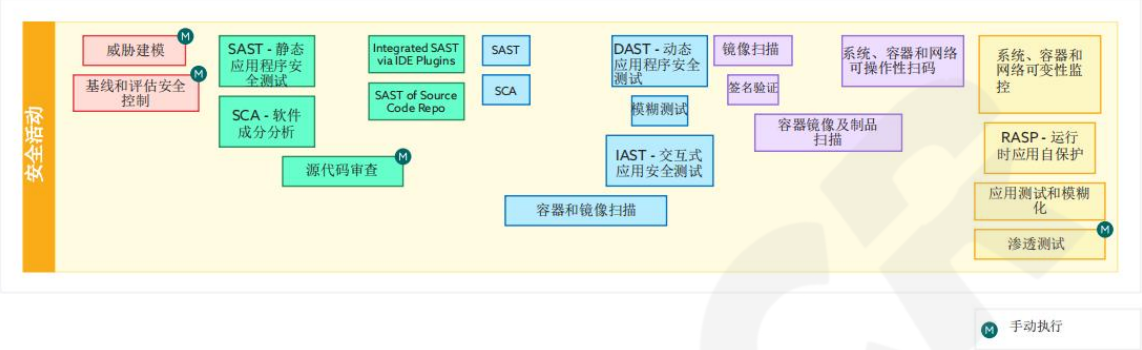


图 1-3：交付流水线中的安全活动

活动是将可交付成果作为输入并产生肯定或否定等结果的单个过程。 如果可交付成果符合活动的标准，则结果被认为是肯定的。 否则，结果是否定的。

具体来说，每个活动都代表安全控制的实施——为检测代码、库、应用程序或应用程序将要运行的环境中的安全漏洞而执行的测试或验证任务。

活动独立于阶段和触发器：一些活动可以在所有阶段运行，或具体到单个阶段运行。多个触发器可能依赖于同一个活动的结果。 参考文档中提供了每个安全活动的详细描述和应用指南。

4.3 流水线的设置和维护

为了支持 DevSecOps，交付流水线应该建立在 DevOps 团队已经在使用的工具和流程中。安全需要成为正常发布流程的一部分，以减少摩擦并支持高效、快速的发布周期。

从安全的实用角度来看，流水线安全的测试功能应该由高价值、低影响的功

能逐步引入。对已经使用代码质量管理工具或库管理[3]工具的安全测试可以快速产生效果，同时对操作或额外资源的产生影响最小（如识别供应商、获得新的技术和仪器）。

在引入例如静态应用安全测试（SAST）新的测试功能时，DevOps 团队应首先关注特定的发现类型，调整工具以防止误报，确保最大的代码覆盖率（消除假阴性）。DevOps 团队在自动化测试中增加新的发现类型或功能之前，应该对遗漏的漏洞和误报进行根本原因分析，以获得最佳效率。

大量的误报会给 DevOps 团队带来负担，损害安全工具的可信度。

一个好的策略是 “在向右加速的同时安全向左转移”。自动化的安全控制和行为不会取代一个组织现有的 SDLC 政策、标准和程序。相反，自动化增强了现有的能力，以确保流程和资产的安全。组织应该从手动执行的控制和活动开始向自动化转变。

5.风险优先的流水线

5.1 概述

不是所有的应用程序都有相同的风险状况。需要有一种机制，能对有稳定记录的高风险应用快速跟踪，并为其交付提供更多的审查，这将利于在确定资源优先级和可交付性方面取得平衡。

在部署构建时，应使用基于风险的方法来确定安全测试的总体严格程度。通常有如下三种类型的风险因素需要考虑：

3 https://www.csoonline.com/article/3398485/28-devsecops-tools-for-baking-security-into-the-development-process.html

5.2 风险因素

5.2.1 应用程序风险

对应用程序其固有的风险进行评估，包括考虑其处理的数据和使用环境。

例如：处理银行或医疗保健数据的应用程序可被视为 “高风险”，而处理 PII 和信用卡的应用程序可被视为 “中等风险”。那些处理静态内容或公共信息的应用程序可被视为 “低风险”。

5.2.2 变更请求的风险

因为变更而引入的风险，包括变更可能带来的安全影响、受变更影响的数据以及变更的性质。

例如：影响核心安全组件的变更（如身份验证或访问管理），涉及重构而引入新功能或技术的变更对应用程序的安全状况可能重大的影响。因此需要应进行更彻底的测试。

5.2.3 流水线及其交付物的可靠记录的风险

归因于交付流水线和其所涉及应用程序的可靠性和完整性历史的风险。

例如：一个历史稳定、具有安全发布记录的交付流水线比一个发布历史的不稳定的交付流水线的风险要求。具有稳定安全发布的应用程序比有一连串安全漏洞的应用程序更可靠。

5.3 流水线配置的优先级

交付流水线的配置可以根据风险因素的整体性进行优先排序。每个交付流水线和应用程序都是独一无二的；在实施过程中，需要对每一个用例区别考虑风险，

以满足是否其需求。

对交付流水线的差异化处理，为安全开发和部署应用程序提供了分级机制，从而缩短交付时间。

用交通灯的方式来解释，即有三个不同颜色的处理流水线：绿色、黄色和红色，每个流水线提供不同级别的安全审查：

- 绿色流水线：运行测试时间短，为持续交付而优化。
- 黄色流水线：可能涉及 DAST，处理时间短的小规模带外活动，可持续交付。
- 红色流水线：可能涉及 IAST，处理时间长的大规模带外活动，难以持续交付。

示例 1：在所有三个风险因素中排名较低的应用程序的交付可以视为是“整体风险低”，并可以沿着 “绿色流水线”继续处理。

示例 2：在低风险应用程序中，不影响处理个人数据共享的代码变更可以视为“中等风险”，可以沿着“黄色流水线”继续进行。

示例 3：对有重大漏洞记录的应用程序的交付，应按照“红色流水线”进行更严格的安全审查。

示例 4。如果提议的变更为安全处理数据引入了新的标准组件，或要求开发人员接受相关安全最佳实践的培训，则适用“红色流水线”。

当检测到新漏洞时，应实施根本原因分析，以确定原因是否与人员、流程或技术有关，并对相关内容进行重新设计或变更，防止漏洞再次发生。

下表提供了一个变化的评分样本，结合这三个因素来确定目标流水线配置。采用高（3）、中（2）、低（1）的数值，总分 5 分及以上被认为是在“黄色流水线”，7 分及以上被认为是在“红色流水线”。

应用风险		变更风险	可靠性风险	交付流程配置
变更 #1	中（2）	低（1）	低（1）	绿色流水线（4）
变更 #2	低（1）	高（3）	中（2）	黄色流水线（6）
变更 #3	高（3）	高（3）	高（3）	红色流水线（9）

表 1：生成影响和结果发布流水线

6. 交付流水线的活动框架

6.1 概述

本节介绍了一个框架（见图 2），它代表了交付流水线中使用的六类安全活动，以及这些活动的指导方针。

6.2 安全的设计

只有在设计中考虑了安全措施，才能实现安全。将事后才考虑应用安全措施，是一种灾难性的做法，已知设计缺陷占漏洞的很大一部分。

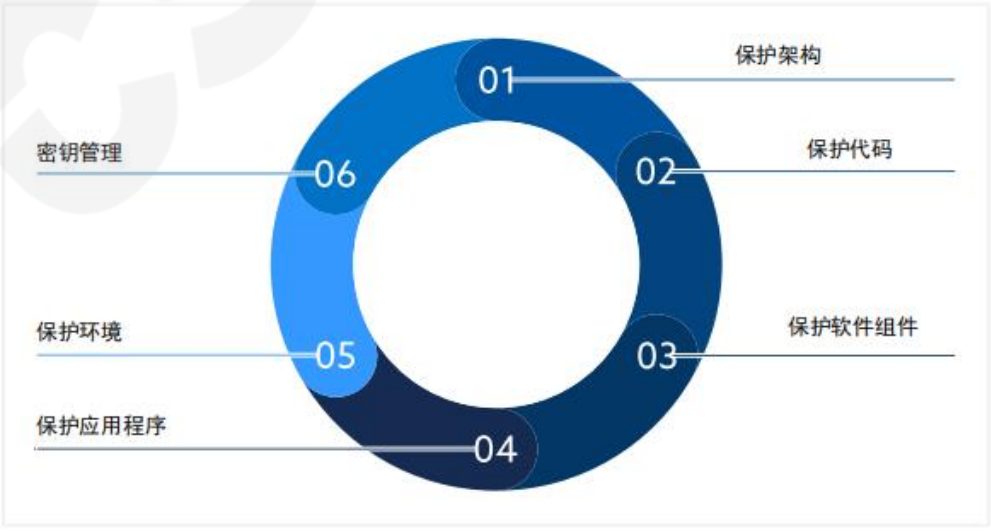


图2: 已启用安全功能的交付管道中的安全活动

虽然有人认为手动流程与 DevSecOps 实践不相容，但事实并非如此。有些核心的安全设计活动需要人类的智慧。例如威胁建模，必须手动执行，以确保建立一个安全的架构。

传统上，威胁建模只能手动进行，尽管现在有一些工具可以帮助实现这一过程的自动化。但除非应用程序中添加了新组件或安全敏感组件发生了变更，此外威胁模型通常不需要，所以手动更新仍然是一种可行的机制。

6.3 安全的编码

无论一个应用程序的安全架构是多么的周密和健全，漏洞都可能来自糟糕的编码。如果开发人员编写的代码包含安全缺陷，应用程序被攻破只是时间问题。

SAST 工具可以帮助扫描交付物的源代码，并报告安全（和通用）缺陷，防止它们表现为安全漏洞。

在自动化静态代码测试时，DevSecOps 的目标是尽可能早地通过上下文向开发人员提供代码安全警报，并规范其反馈循环。

在交付流水线中，很多时候会执行安全扫描，包括：

- 本机扫描：通过 IDE 或集成测试套件；
- 代码扫描的持续集成：在持续集成系统上对代码推送、合并、发布实施代码扫描；
- 持续交付预部署：部署前扫描；

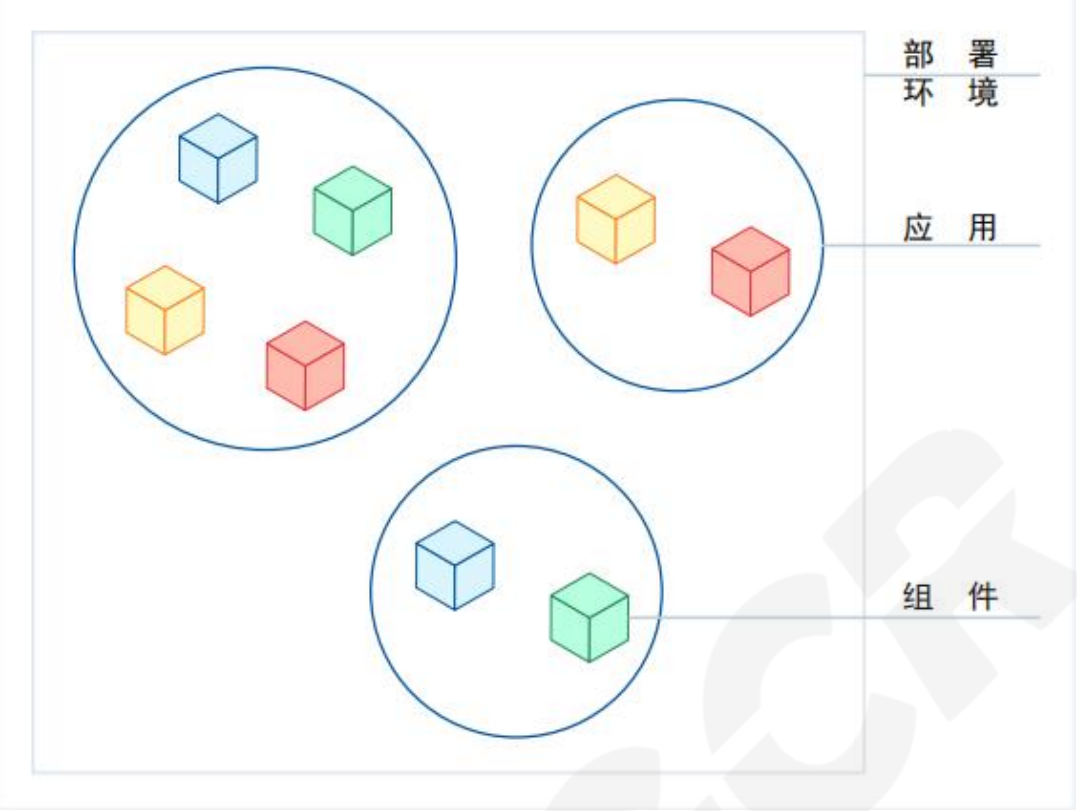


图 3：应用程序组件与环境之间的关系

6.4 安全的软件组件

应用程序的安全性依赖于其组件的安全性。无论是来自内部开发、第三方引入还是开源项目引入的组件漏洞，都会损害更大的应用程序。

以系统的方式管理这些软件组件至关重要，可以采取以下措施：

- 及时发现和报告第三方软件中的漏洞，如开源代码和容器
- 检测未经授权的组件使用情况

组件的漏洞管理应该是连续的，这样不仅在开发过程中进行，而且在其后也同样实施。重要的是确保使用的第三方组件在应用程序生命周期内保持安全。

在公开应用程序组件的漏洞时，应触发一个提示需要将组件库升级到无漏洞版本的流程。

这个流程通常可以通过持续集成过程或运行时扫描，使用源代码扫描工具自动执行。

6.5 安全的应用程序

应用程序必须在现实环境中进行安全性测试，才能被信任。即使使用了可靠的体系结构、安全的代码和无漏洞的组件，它们的集成也不一定会导致安全的应用程序。

动态应用程序测试 DAST 可用于测试分段在测试和生产环境中对应用程序进行安全测试以保证安全性。DAST 工具可以发现各种安全问题，从故障注入、故障检测到资源和秘密泄露。（DAST 在流水线中的适当位置见图 1）。

模糊测试也是识别安全漏洞的一种有效方法。在交付流水线中运行各种类型的自动模糊测试，以确保应用程序对错误和恶意输入具有一定的弹性。

6.6 安全的运行环境

一般情况下

不安全的环境会使得本来安全的应用程序变得不安全。基础架构即代码（IaaC）、应用程序容器化和持续部署的方法为保护基础架构和环境提供了新的可能性，包括：

图4: 采取多管齐下的方法来确保基础设施的安全

- IaaC 代码和工件扫描
- 环境的运行态防护

IaaC 安全

IaaC 代码用于云基础设施资源的声明式管理，IaaC 配置状态可以准确反映这些资源的实际配置。IaaC 代码和工件的安全分析可以通过两种机制执行：

- 对 IaaC 模板和配置状态的静态扫描
- 对声明式部署的资源进行自动扫描

运行时安全

部署的应用程序受其环境的安全约束，因此必须进行监控。通常有两种方法：

- 对已部署的应用程序和环境的运行态监控是确保环境和应用程序都受到保护的重要措施（见图 4）。
- 运行态防御确保在运行时发现的漏洞和攻击得到缓解，并尽快报告。

6.7 密钥管理

应用程序和交付流水线中的机密管理至关重要，会影响到整个交付流水线。使用合适的加密算法不一定能提供足够的安全性对抗攻击者。例如，开发人员选择了最佳的加密算法，但如果把密钥存储在公开的位置，安全性就会受到威胁。

包括密钥管理和密钥轮换在内的过程必须得到妥善处理。云密钥存储库和密钥管理基础架构可以被视为自动化解决方案的一部分。

7. 自动化最佳实践

7.1 概述

在安全的交付流水线中使用的很多安全活动可以在任何 DevSecOps 流程中单独使用。大多数成熟的组织都有带外（也称为异步）测试的规定。本节重点介绍适用于 DevSecOps 以外的最佳实践和注意事项。

7.2 缓解漏洞

漏洞在刚出现时最容易解决。考虑到这一点，我们采取的战略是在部署之前，

将新漏洞的检测转移到开发过程中。例如，可以考虑在 IDE 中或提交代码变更之前部署检查安全问题的工具。

对于在后期阶段检测到的易受攻击的代码，在生产部署之前必须对其进行修补。根据评估的漏洞风险，低风险的漏洞可以在稍后处理，高风险漏洞必须尽快解决。

7.3 异步测试

一般情况下异步测试有两类：

- 某些关键的安全活动无法完全自动化，因为需要人工参与。这些活动，包括威胁建模、渗透测试和代码审查；
- 需要较长时间执行的重量级自动化测试，如 SAST。

这些活动可以“带外”触发和执行，而不是自动部署内联，避免中断软件部署流水线。

带外识别的问题必须由交付团队跟踪，以确保可见性、优先级和根据识别的风险进行潜在的回滚。如果带外测试产生的关键问题没有得到及时解决，则应在适当的检查点将其视为连续交付过程的障碍。

7.4 持续反馈回路

应用程序中存在的安全缺陷时间越长，进一步的变更和部署就越有可能增加修复缺陷的复杂性和成本。

DevSecOps 团队必须尽早警示安全问题，以防止缺陷向下游蔓延。

另外，及时的反馈能够使个人实施快速诊断和漏洞补救，并能在记忆犹新时进行根因分析。

7.5 破坏构建

组织需要认识到，安全性是安全活动的一项业务要求。在中断自动部署流水线的同时，声明未能满足安全要求的“失败构建”。

为安全而中断构建可以加强质量，向 DevSecOps 团队提供快速反馈，并防止数据泄露和组织风险偏好之外的潜在漏洞攻击。例如发布具有高严重性、公开漏洞的代码，以及具有活跃利用的漏洞。

可能中断的构建活动包括：

- 发现的高风险缺陷超出了组织的风险偏好；
- 识别出与组织策略冲突的特定漏洞或漏洞类别；
- 风险级别高或严重的漏洞数量超过一定阈值；
- 必要的安全活动无法有效执行，例如：配置错误、自动扫描失败；

8. 总结

为了从 DevSecOps 和反思性安全方法中获益，必须大量依赖自动化。

自动化支柱可以使用安全的交付流水线并通过在其中应用安全控制来实现。风险优先的方法进一步允许优化持续交付和处理非自动化安全检查机制。

参考文献

1. security risks that architecture analysis can resolve. Synopsys, 2016. Available at: https://www.synopsys.com/blogs/software-security/security-risks-that-architecture-analysis-can-resolve/

2. Tactical Threat Modeling. SAFECode, 2017. Available at: https://safecode.org/wp-content/ uploads/2017/05/SAFECode_TM_Whitepaper.pdf

3. Managing Security Risks Inherent in the Use of Third-party Components. SAFECode, 2017. Available at: https://safecode.org/wp-content/uploads/2017/05/SAFECode_TPC_Whitepaper.pdf

4. Focus on Fuzzing. SAFECode, 2020. Available at: https://safecode.org/fuzzing/

CSA GCR cloud security alliance
GREATER CHINA REGION

Cloud Security Alliance Greater China Region

扫码获取更多报告

云安全联盟大中华区
官网：https://c-csa.cn 电话：0755-86548359
地址：上海市浦东新区东育路255弄5号3FB30（前滩国际经济组织集聚区）/
广东省深圳市南山区粤海街道北京航空航天大厦1号楼905