软件开发过程中的安全前置研究与实践

Research and Practice of Shift Left Security in Software Development Process

王　戈[1,2]，徐　雷[1,2]，郭新海[1,2]，徐　锋[3]，徐积森[4]（1. 中国联通研究院，北京 100048；2. 下一代互联网宽带业务应用国家工程研究中心，北京 100048；3. 杭州孝道科技有限公司，浙江 杭州 310020；4. 中国联合网络通信集团有限公司，北京 100033）

Wang Ge[1,2]，Xu Lei[1,2]，Guo Xinhai[1,2]，Xu Feng[3]，Xu Jisen[4]（1. China Unicom Research Institute，Beijing 100048，China；2. Next Generation Internet Broadband Service Application National Engineering Research Center，Beijing 100048，China；3. Hangzhou Xiaodao Technology Co.，Ltd.，Hangzhou 310020，China；4. China United Network Communications Group Co.，Ltd.，Beijing 100033，China）

摘　要：

软件开发过程中的安全前置是确保软件安全的关键策略，但在实践中仍面临诸多挑战。为此，借鉴SDL和DevSecOps模型中将安全融入软件开发的各个阶段这一理念，建设安全能力与自动化工具链，构建软件开发安全管理体系，并通过建立软件开发过程中各阶段安全能力的关联来提升安全测试效率，在提升软件安全性的同时又降低了软件后期因安全问题而产生的维护成本，真正在软件开发过程中落实了安全前置的理念，并充分发挥了安全前置的作用。

关键词：

软件安全；安全前置；软件生命周期

doi：10.12045/j.issn.1007-3043.2024.08.007

文章编号：1007-3043（2024）08-0034-05

中图分类号：TP311.5

文献标识码：A

开放科学（资源服务）标识码（OSID）：

Abstract：

Shift left security in the software development process is a key strategy to ensure software safety，though it still faces many challenges in practice. Therefore，by adopting the concept from the SDL and DevSecOps models of integrating security into all stages of software development，it builds security capabilities and automated tool-chains，establishs a software development security management system，and enhance the association of security capabilities at various stages of software development to improve security testing efficiency，which improves software security and also reduces maintenance costs incurred due to security issues later in the software lifecycle，thereby truly implementing the shift left security philosophy in software development and fully leveraging its benefits.

Keywords：

Software security；Shift left security；Software lifecycle

引用格式：王戈，徐雷，郭新海，等. 软件开发过程中的安全前置研究与实践[J]. 邮电设计技术，2024（8）：34-38.

1 概述

随着数字化转型的持续推进，软件在日常生活和商业活动中所起的作用越来越重要。然而，如今软件安全事件的频繁发生，使得人们对软件安全性的关注度不断提升，对提升软件安全性的需求不断增强[1]。软件安全也由原先的被动防御型安全逐步发展为当前主流的主动防御型安全，即在软件开发全生命周期的各阶段充分挖掘潜在的安全威胁，通过威胁建模、安全设计、安全测试等多个角度消减威胁和建立防御措施。这种强调在软件开发的早期阶段就考虑和实施安全措施，以确保最终的软件产品具有较好的安全性的行为被称为软件开发安全前置，或者被称为安全左移。通过实施安全前置，可以降低后期修复安全漏洞的成本，并提高系统的整体安全性[2]。美国国家标准与技术研究所（NIST）曾发布了一份报告，报告指

收稿日期：2024-06-19

出，将安全性融入软件开发生命周期的早期阶段可以将漏洞修复成本降低到产品发布后的6~60倍，所以软件开发安全中的安全前置越来越受到重视。

2 软件开发安全前置的现状与痛点

当前软件开发安全大都参考和遵循SDL（Security Development Lifecycle）模型，后来随着云计算的发展与普及，软件开发的迭代速度不断提升，出现了DevSecOps（Development，Security，and Operations）模型，这2种模型已经成为软件开发安全的主流模型并被广泛应用[3]。其实在实际的软件开发安全中，不管是SDL还是DevSecOps，其主要强调的就是安全前置，将安全更早地融入软件开发生命周期，就能更容易地收敛安全问题。

2.1 SDL中的安全前置

SDL是由微软提出并侧重于软件开发的安全保证过程，旨在开发出安全的软件应用[4]。SDL的核心理念就是将安全考虑集成在软件开发的每一个阶段：需求分析、设计、研发、验证、发布和运营。从需求、设计到发布和运营产品的每一个阶段都增加了相应的安全措施，以减少软件中漏洞的数量，并将安全缺陷降低到最小程度，从而提升软件的安全性（见图1）。

2.2 DevSecOps中的安全前置

DevSecOps旨在将安全性纳入到DevOps文化和实践中，从而实现安全、敏捷和持续交付的软件开发流程。DevSecOps不再是一个单纯的安全开发模型，它强调的是人人为安全负责，人人参与安全，安全嵌入到从开发到运维的每个阶段，在不同的阶段需要进行不同的安全动作，将安全的思想贯穿于整个软件的开发和运营过程中，实现安全前置，降低软件的安全风险[5-6]（见图2）。

2.3 当前软件开发过程安全的痛点

在软件安全开发过程中，无论是SDL还是DevSecOps模型中的安全前置，还存在如下2个亟需解决的问

图1　SDL各阶段的安全举措

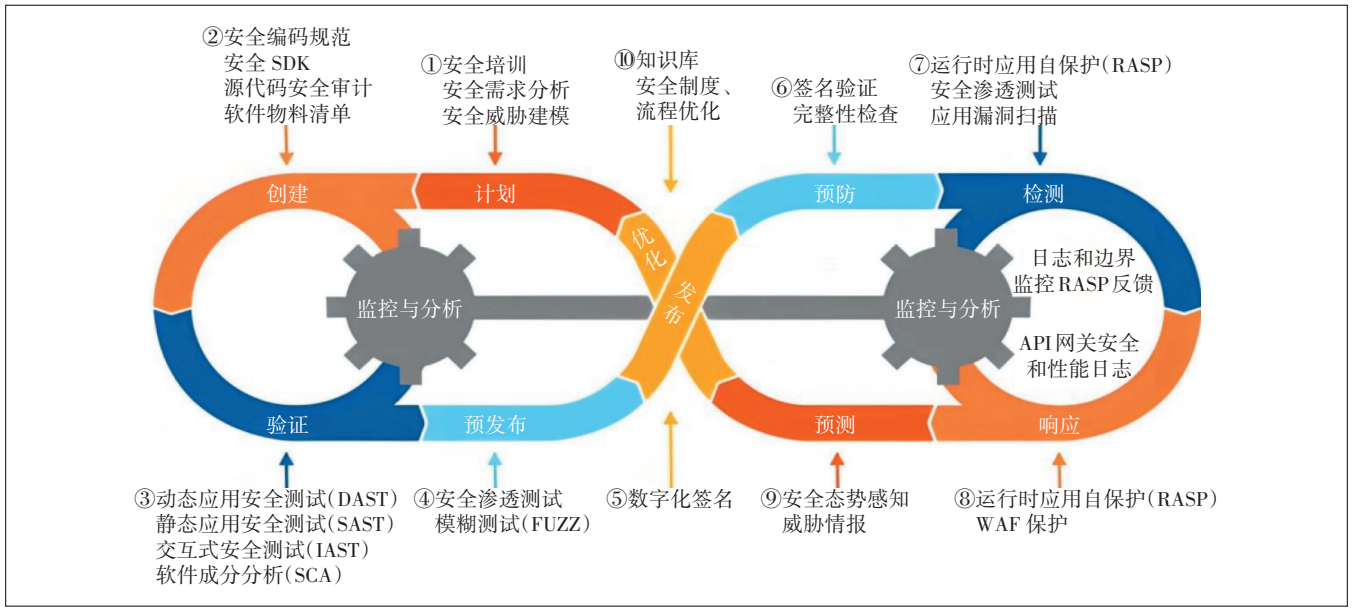

图2　DevSecOps中各阶段的安全举措

题。

a）快速迭代与安全性的平衡。当前的软件开发大都强调快速迭代，又因为在开发过程中引入安全前置会消耗相当的资源和时间，从而会降低迭代速度，这往往导致安全性在开发过程中被忽视或延迟处理。此外，企业需要在保证业务快速交付的同时，确保产品的安全性，这在实际操作中往往难以平衡，产品的安全性会让位于业务的需求，从而降低了软件的安全性。

b）软件开发过程中的安全管理挑战。无论何种安全开发模型，如果没有完善的安全管理体系将安全手段融入软件开发流程中，就无法切实落实软件的开发安全。所以，只有建立一套完善的软件开发安全管理体系，规范化安全开发流程，才能在满足软件开发需求的前提下，保证软件的安全性。

3 安全前置的优化方案

针对前述安全前置现存的问题，如何建设软件开发安全自动化工具链、构建软件开发安全管理体系、提升安全测试效率是落实软件开发安全的重中之重。图3所示为安全前置优化方案的总体架构。

3.1 建设安全能力和工具链

将安全融入软件开发过程，需要在各阶段建设安全能力和对应的安全工具，从而保证开发过程每一个阶段的基础安全。

3.1.1 安全需求分析

安全需求分析旨在识别和定义软件在安全方面的期望行为和要求。这些需求描述了软件应如何处理安全问题，以及必须满足哪些安全标准和规范[7]。

安全需求分析工具可以通过获取用户需求，自动化分析软件的应用场景，得到安全需求，为后续的威胁建模和安全设计、安全控制措施的制定以及安全编码的实现提供了指导和依据，提高软件的安全性和合规性。

3.1.2 安全威胁建模

安全威胁建模是识别并评估如何管理软件中安全弱点的过程，可以快速发现软件应用过程中的安全隐患，清楚地了解安全建设需求，从而有效地建立软件安全防御体系。安全威胁建模其实就是从风险防御的视角来评估软件及其环境的安全性[8]。

自动化威胁建模工具早已被广泛地应用于软件开发过程，从而提升软件的安全性，其主要原因如下：首先能够明确软件的安全防护需求，降低软件的安全风险；其次在开发的早期发现问题，降低后期修复和

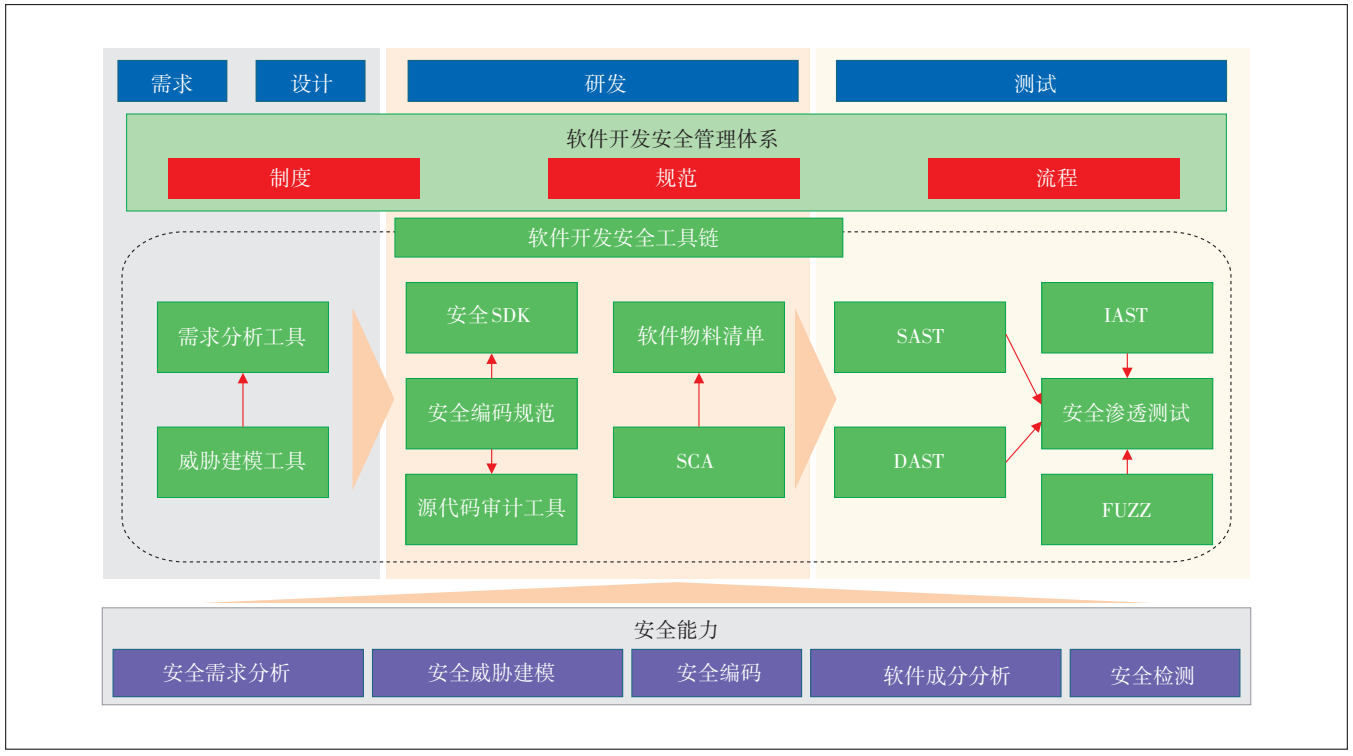

图3　软件开发过程中安全前置体系架构

运维的成本；最后使软件的风险直观可见，量化软件面临的威胁。

3.1.3 安全编码

安全编码能够减少软件开发中漏洞的引入，提高开发人员安全编码水平，从根源上有效避免软件中的原生缺陷[9]。

安全编码规范使开发人员熟悉常见漏洞的安全编码方法和修复方法，避免漏洞的引入，使已存漏洞得到快速修复。安全SDK提供安全函数接口，将安全编码规范内容进行编码落地，实现安全功能通用化，提高安全编码效率。源代码安全审计工具是通过解析源代码、匹配规则集、分析数据流和控制流，自动检测潜在的安全编码问题。

3.1.4 软件成分分析

软件成分分析能够帮助开发人员对软件组成有更深度的了解，更好把握软件结构，增加软件的透明度。软件成分分析技术能力主要包括软件物料清单和软件成分分析等[10]。

软件物料清单是软件成分信息的集合，记录软件产品或服务所使用组件、库、框架的清单，描述软件构建过程中使用的所有组件及其关系。软件成分分析工具（SCA）的主要功能是针对第三方开源软件以及商业软件涉及的各种源码、模块、框架和库进行分析、清点和识别，分析其构成以及依赖关系，识别出已知的安全漏洞等。

3.1.5 软件安全检测

软件安全检测能力需要覆盖软件开发全生命周期，是确保软件安全的关键手段，通过安全检测，能够高效、快速地检测出软件安全问题，降低后期软件运营阶段修复漏洞的成本[11]。

静态应用安全测试（SAST）是对程序源码进行安全检测的技术，与传统的源代码审计工具类似，采用词法、语法、控制、数据流等技术，挖掘源代码中的缺陷。动态应用安全测试（DAST）是一种黑盒测试方法，通过模拟黑客攻击的方式对软件进行测试，评估软件在真实环境下的安全性。交互式应用安全测试（IAST）将特定Agent插桩到软件中，从内部监控软件运行时的行为并分析，发现软件存在的漏洞。模糊测试（FUZZ）将非预期的数据输入目标软件，查看是否发生故障，并提供故障原因及修复建议。安全渗透测试是一种模拟黑客攻击的系统安全评估方法，旨在发现网络和软件中的潜在安全漏洞，当前应用最广的渗透测试还是以人工渗透为主，自动化工具为辅。

3.2 构建软件开发安全管理体系

为了更好地落实软件开发过程中的安全前置，除了需要构建安全能力及自动化工具链以外，还要完善软件开发安全管理体系。首先，根据实际的软件研发环境制定合适的开发安全管理制度，明确软件开发过程中各阶段的安全举措，规范化软件开发流程；其次，在开发过程中各阶段设置安全门禁，只有达到预先设定的安全标准后才能进入下一阶段，从而保证软件在各阶段的安全性；最后，根据安全管理制度和安全标准将安全工具链接入软件开发流程管理工具，形成有规范有落地的软件开发安全管理体系，真正实现安全前置。

3.3 提升安全测试效率

软件安全测试效率的高低制约着安全前置的落实，虽然已有自动化安全测试工具在一定程度上提升了软件安全测试的效率，但是由于安全测试的复杂性和特殊性，会耗用大量的时间，从而拖慢整个软件开发的进度，成为快速迭代的瓶颈。通过将各阶段的安全能力建立关联，对各阶段的安全活动结果做综合的分析和运用，有效提升各阶段的安全活动效率以及安全风险检测的精确度。例如：运用安全需求分析和安全威胁建模的结果生成安全测试用例，更聚焦、更有针对性地减少不必要的安全测试用例，从而提升安全测试的效率；在软件编码阶段严格落实软件编码规范、安全SDK的使用、源代码的安全审计以及软件成分分析，在软件的测试阶段会提升静态应用安全测试的检测效率，降低误报率，从而提升安全测试的效率和准确性，能够更好地适应高速迭代的需求。

4 软件开发安全前置实践

根据前文已给出优化后的安全前置方案构建软件开发安全管理平台，可以更好地保障在软件开发安全实践中落实和推广安全前置，从而提升软件的安全水平（见图4）。

a）接入自动化安全工具，构建安全基础能力。将安全需求分析工具、安全威胁建模工具、安全SDK库、SCA、SAST、DAST、IAST自动化工具接入平台进行统一管理和调度，形成覆盖软件开发全周期的安全基础能力。

b）落实安全管理制度，提升软件安全水平。结合开发流程，将软件开发安全管理制度在管理平台全面

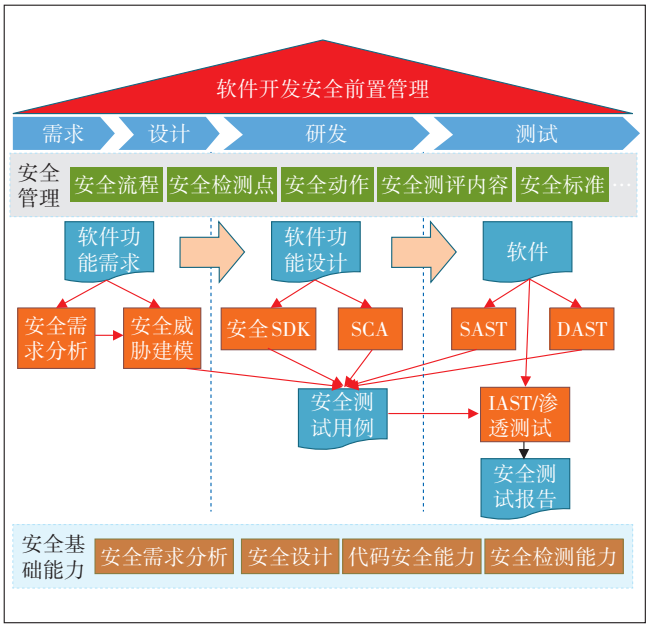

图4　软件开发安全前置平台架构

落实，在软件开发的各阶段调用对应的安全工具，执行相应的安全活动，同时对执行情况进行核实，只有达到相应的安全要求才能通过该阶段进入软件开发的下一阶段，在软件开发过程中严格落实安全前置，全面提升软件的安全水平。

c）优化工具使用，提升安全检测效率。安全融入软件开发流程后，在开发各阶段都有相应的安全工具来执行对应的安全动作，如图4所示，在实践过程中，以安全测试用例为纽带，将安全工具链上的各种工具建立关联，在各阶段借鉴前一步安全活动的结果不断补充和优化安全测试用例，最终形成一份覆盖全面且具有针对性的精简测试用例，在完成对软件安全测试的同时，提升安全检测的效率。

与常规的安全前置方案相比，以自动化的安全工具链为基础，可以实现软件开发过程中安全活动的自动化执行，降低人为因素的干扰，提升结果的精确度；健全软件开发安全管理体系，规范化安全前置流程，确保安全开发流程的全面实施和执行，保证软件的基本安全；建立工具间的关联，优化工具的使用，可以在保证安全测试覆盖全面的前提下，使测试用例更具有针对性，从而减少无效安全测试用例的执行，降低资源的消耗，真正提升测试效率，从而能够更好地适应软件快速迭代开发的需求，保证了安全前置不会因为业务需求而被平衡掉，充分发挥安全前置的作用，切实保障了软件的安全性。

5 结束语

在国家大力推广数字化转型的当下，软件的发展和应用会越来越普及，随之而来的软件安全问题也会更加受到关注，所以如何能在不影响软件开发与应用的同时还能降低软件的安全风险、减少软件的维护成本，在未来很长的一段时间内也会成为研究热点。将安全前置并融入到软件开发的过程中，虽然在降低软件安全风险与减少软件维护成本方面取得了一些成果，但是在实际应用的过程中还存在或多或少的问题，从而限制了安全前置作用的发挥，未来当这些问题被解决之后，相信软件开发过程中的安全前置将会在保证软件安全性方面发挥更大的作用。

参考文献：

[1] 刘剑，苏璞睿，杨珉，等. 软件与网络安全研究综述[J]. 软件学报，2018，29(1)：42-68.
[2] 苏俐竹，徐雷，郭新海，等. 国内外软件供应链安全现状分析与对策建议[J]. 邮电设计技术，2022(9)：24-26.
[3] 罗欣然，营志刚，徐瑞祝，等. 我国软件安全开发体系建设的观察与研究[J]. 中国信息安全，2024(3)：85-87.
[4] 李晓蕴，刘海峰. SDL安全开发流程在软件开发中的应用[J]. 软件，2023，44(5)：104-106.
[5] 张小梅，苏俐竹. 基于DevSecOps的软件供应链安全治理技术简析[J]. 邮电设计技术，2022(9)：13-18.
[6] 刘羿希，何俊，吴波，等. DevSecOps中软件安全性测试技术综述[J/OL]. [2024-05-19]. http://kns.cnki.net/kcms/detail/51.1307.TP.20240124.1520.004.html.
[7] 严超，周悦，张孟. 基于SSDLC的安全需求分析研究[J]. 网信军民融合，2022(2)：26-30.
[8] 刘佳杰，伍宇波，张煜，等. 威胁建模在安全态势感知中的应用研究[J]. 中国金融电脑，2018(11)：81-85.
[9] 贺江敏，相里朋. 代码安全性审查方法研究[J]. 信息安全研究，2018，4(11)：977-986.
[10] 袁豪杰，赵冉，唐刚. 面向开源软件的安全风险分析与防范[J]. 信息安全与通信保密，2023(2)：125-134.
[11] 王环宇. 软件安全性测试方法研究[J]. 电子技术与软件工程，2016(14)：65.

作者简介：
王戈，工程师，硕士，主要从事网络与信息安全研究工作；徐雷，教授级高级工程师，博士，主要从事网络与信息安全研究工作；郭新海，工程师，硕士，主要从事网络与信息安全研究工作；徐锋，高级工程师，主要从事软件供应链安全与安全开发管理研究工作；徐积森，高级工程师，主要从事网络安全运营工作。