



S-SDLC CMM 第一版

二零二二年四月

深圳开源互联网安全技术有限公司

文档修订记录

版本	*变化状态	修订说明	日期	变更人	备注
V1.0	A	建立文档	2022/4/26	seczone	

目 录

第一部分：模型概述	1
1 S-SDLC CMM 介绍	2
1.1 S-SDLC CMM 模型特性	2
1.2 S-SDLC CMM 模型使用	2
1.3 术语	3
2 S-SDLC CMM 结构	3
2.1 过程域	4
2.2 子过程域	4
2.3 S-SDLC CMM 框架	7
3 S-SDLC CMM 能力等级与成熟度级别	14
3.1 S-SDLC CMM 能力等级	14
3.2 S-SDLC CMM 成熟度级别	18
3.3 能力等级，成熟度级别关联关系	20
4 S-SDLC 软件安全成熟度级别评估	22
4.1 评估方法	22
4.2 评估实施和评估	22
4.3 评估制约因素	24

第二部分：S-SDLC CMM 安全实践	25
1 监管	26
1.1 流程与政策.....	26
1.2 合规性	28
1.3 培训	29
2 能力	31
2.1 攻击模型.....	31
2.2 安全特性与安全设计	31
2.3 标准与要求.....	32
3 开发过程接触点	36
3.1 需求分析	36
3.2 安全设计	36
3.3 安全开发	38
3.4 安全测试.....	39
4 部署&运维	41
7.1 渗透测试.....	41
7.2 软件环境.....	42
7.3 漏洞管理.....	43

7.4	运营支持.....	44
7.5	应急响应.....	45

第一部分：模型概述

第一部分主要包括 S-SDLC CMM 模型（软件安全成熟度模型）特点、意义以及常用术语定义。并介绍了 S-SDLC CMM 模型结构和成熟度级别，有助于从整体上帮助理解第二部分 S-SDLC CMM 实践。

1 S-SDLC CMM 介绍

S-SDLC CMM(S-SDLC Capability Maturity Model)是一套软件安全开发体系规范，由最佳实践方法构成，覆盖 S-SDL 流程中的关键要素，强调的是构建并维护整个软件开发周期所需从事的安全开发工作。

S-SDLC CMM 模型针对软件研发过程的每一个阶段，均试图更精准的识别出软件开发组织当前的安全研发的实施现状，从监管、能力、开发过程接触点和部署&运维四个维度来体现每一个过程与细节的实施状态。

1.1 S-SDLC CMM 模型特性

- 更专注于是否得到实施以及实施的方法、手段以及解决方案本身是否符合业界最佳实践，而不只是识别企业安全开发过程与活动是否存在。
- 更专注于每一个具有代表性的细节的实施是否是合理且恰当，而不只是大颗粒度的呈现企业安全开发当中的问题。

1.2 S-SDLC CMM 模型使用

1) 指导软件安全开发过程的改进

S-SDLC CMM 模型描述了对企业达成其安全开发目标富有成效且实用的最佳实践，所以 S-SDLC CMM 可作为企业改进软件安全开发体系的落地参考框架。

2) 指导软件安全开发能力的评估

通过模型评估，可以反映组织软件安全开发体系建设真实水平，从而找到了度量企业软件

开发安全所取得的价值。

1.3 术语

过程域	某一领域内的一组相关安全实践集，当它们共同得到实施时，可帮助企业建立软件安全开发所需的相关能力。
子过程域	某一详细领域内的一组相关安全实践集，当它们共同得到实施时，可帮助企业建立软件安全开发所需的相关能力。
安全实践	每一项安全实践都描述了一组安全活动，这组活动的执行效果以及制度化特征共同决定了安全实践的能力等级。
通用目标	描述了安全实践在制度化方面的特征，S-SDLC CMM 定义了“未执行”、“基本执行”、“计划跟踪”和“充分定义”四个通用目标，分别对应“安全实践是不完整的”、“安全实践是被动的、随机的”、“安全实践是主动的、正式的”以及“安全实践是正式的、规范的”四种制度化特征。
专业目标	特指与安全实践相关的特定目标，这些目标的达成情况体现了安全实践的执行情况：未执行还是已执行。

2 S-SDLC CMM 结构

S-SDLC CMM 包含了 4 个过程域，4 个过程域包含 15 个子过程域，子过程域包含 64 个

实践活动；即 S-SDLC CMM 是由 64 个实践活动共同有序组成的。

其中成熟度级别 1 的安全实践都有对应的成熟度级别 3 的安全实践，因而没有列出。

成熟度级别 5 企业表示 2-4 级的安全实践已经完全满足并在软件安全建设取得显著成效，具有示范性和行业代表性。成熟度级别 5 的企业在软件安全领域已经成为行业标杆。所以成熟度 5 级没有独特的安全实践。

2.1 过程域

S-SDLC CMM 模型按照不同的安全范围分为监管、能力、开发过程接触点和部署&运维四个安全实践集。



2.2 子过程域

四大过程域分为 15 个子过程域，在每个子过程域有序包含同领域的安全实践：



不同的子过程域包含一系列安全实践：





能力

攻击模型

- 从攻击者的视角分析分析和研究可能存在的攻击者、攻击模式和相关技术，以便软件开发组织能够防患未然。

安全特性与安全设计

- 包括开发安全组件、建立威胁库和安全设计库等实践，主要为软件开发组织的安全设计过程提供支撑。

标准与要求

- “标准与要求”包含定义安全需求基线、安全编码规范、代码安全审核、安全测试标准等实践，主要目的是为各阶段的SDL活动设定标准。

开发过程
接触点

需求分析

- 完成系统功能的安全需求分析，识别可能面临的安全风险。

安全设计

- 在架构设计时考虑系统的安全性，对系统进行威胁分析，评估系统面临的风险，对设计方案进行安全检查和评审。

安全开发

- 对代码和第三方组件进行人工和自动化的审核，落实安全编码规范的实施。

安全测试

- 在一系列产品测试中进行安全特性驱动测试，包括功能安全性测试等，并对测试发现的安全漏洞进行跟踪管理，确保问题得到解决。



部署&运维

渗透测试

- 渗透测试活动包括对软件系统进行渗透测试，发现安全漏洞，并对发现的漏洞进行管理和解决。

软件环境

- 软件环境活动主要确保公司网络的安全，对个应用程序的输入进行监控，部署安全基础构件保护内网安全。

漏洞管理

- 漏洞管理阶段包括使用工具对漏洞进行全生命周期的管理，并追溯到研发之前的阶段。

运营支持

- 运营支持阶段包括对敏感行为的监控处理，定期审计各类系统运行日志，管理第三方软件的安全漏洞和补丁修复。

应急响应

- 应急响应阶段包括对外部威胁情报的收集，应对信息安全事件的流程机制，以及相关的应急响应团队建设。

2.3 S-SDLC CMM 框架

S-SDLC CMM 框架是有序简略浏览模型的方法并且有助于快速确定评估软件安全成熟度。

下图表展示了 S-SDLC CMM 框架。在本文的第二部分，更完整的描述了活动、示例。

成熟度级别 安全实践	1 级	2 级	3 级	4 级	5 级
 监管					
流程与政策	/	1、 评估整体业务风险概况 2、 建立并维护安全保障项目的路线图险	1、 公布流程（角色、职责、计划），并随着需要发展更新 2、 建立宣传角色并在组织内部开展安全推广活动 3、 明确关卡位置，为通行决策收集必要信息 4、 基于业务和数据的风险对软件系统进行分类，并建立对应的安全目标	1、 通过统一的 SDL 管理系统量化展现安全活动的执行	/
合规性	/	1、 对重要的系统进行合规性审查 2、 对外部采购系统要求通过渗透测试	1、 将合规性法律文件转化为安全需求进入版本开发计划中 2、 建立外部采购成品软件的安全控制制度	/	/

成熟度级别 安全实践	1 级	2 级	3 级	4 级	5 级
培训	/	1、 提供软件安全意识培训 2、 提供软件安全技术培训	1、 提供基于角色的高级课程 2、 根据公司经验教训建立并使用特定的培训材料 3、 对高管进行安全软件意识培训 4、 聘用安全培训专家强化项目团队的安全开发能力	1、 建立针对研发角色的安全定级以及相关的培训考核机制	/
 能力					
攻击模型	/	1、 识别潜在的攻击者	1. 构建与潜在攻击者及特定技术有关的攻击模式 2. 收集和发布攻击案例 3. 在内部建立对攻击案例进行讨论的渠道和方式	/	/
安全特性与	/	1、 使用第三方的安全组件	1. 构建并发布安全特性 2. 安全团队参与系统架构活动	/	/

成熟度级别 安全实践	1 级	2 级	3 级	4 级	5 级
设计			3. 建立相对完善的威胁库		
标准与要求	/	1. 定义安全编码规范 2. 识别开源软件 3. 定义渗透测试标准	1. 建立一个有关安全信息的内网站点 2. 定义安全需求基线 3. 定义安全编码规范 4. 定义代码安全审核的标准 5. 定义第三方开源组件分析活动的标准 6. 定义安全红线 7. 定义安全测试标准 8. 定义数据库、中间件和操作系统的配置基线	1、建立并使用内部完整定义的第三方组件库	/

成熟度级别 安全实践	1 级	2 级	3 级	4 级	5 级
 开发过程 接触点					
需求分析	/	1、重要软件系统项目有开展过安全需求分析	1、为每一个重要的开发功能匹配安全需求	/	/
安全设计	/	1、重要软件系统项目有执行安全特性审查	1、对重要软件系统进行架构安全设计 2、分析设计是否满足已知的安全需求 3、对软件系统/版本/迭代进行威胁分析	/	/
安全实现	/	1、对重要软件系统执行自动化工具进行安全代码审核	1、对版本/迭代执行人工安全代码审核 2、对版本/迭代执行自动化静态代码安全扫描 3、对版本/迭代执行自动化第三方组件安全扫描	1、安全编码规范通过自动化工具实现审查	/
安全测试		1、保证 QA 支持边缘/边界值条件测试	1、对版本/迭代进行安全需求和安全特性驱动测试	1、利用自研的安全测试工具提升测试效率	

成熟度级别 安全实践	1 级	2 级	3 级	4 级	5 级
 部署&运维					
渗透测试	/	1、 使用外部渗透测试人员发现安全问题	1、 上线前对重要软件系统进行渗透测试 2、 渗透测试问题录入漏洞管理平台跟踪管理	/	/
软件环境	/	1、 定期安装操作系统补丁	1、 对应用程序执行输入监控 2、 确保主机和网络的安全基础构件配备到位 3、 对数据库、中间件和操作系统根据安全配置基线进行加固	1、 统一公司的操作系统、数据库及中间件	/
漏洞管理	/	1、 部署内部漏洞管理系统	1、 使用漏洞管理工具管理安全漏洞的全生命周期管理 2、 对运维阶段发现的安全漏洞，追溯到研发阶段进行根因分析	/	/

成熟度级别 安全实践	1 级	2 级	3 级	4 级	5 级
运营支持	/	1、 为典型的应用程序警报记录过程 2、 维护操作环境规范说明	1、 实时监测敏感行为 2、 对系统日志进行审计 3、 建立第三方软件安全检测、补丁修复以及应急预案等机制与流程	/	/
应急响应	/	1、 为安全事件确定联络点 2、 建立非正式的安全响应团队	1、 建立正式的安全响应团队	1、 利用自研的安全测试工具提升测试效率	/

3 S-SDLC CMM 能力等级与成熟度级别

S-SDLC CMM 模型支持两种改进方法。一种是企业在其选定的单个过程域内提升能力，即“连续式”；另一种是企业以增量方式在同等级所有过程域同步改进相关的过程集，即“阶梯式”。

这两种改进方法与两种类型的级别相关联：能力等级与成熟度级别。使用“连续式”方法能够达成“能力等级”；使用“阶梯式”方法能够达成“成熟度级别”。

不管是能力等级或成熟度级别，企业都必须满足既定过程域（过程域集合）中的所有目标。

3.1 S-SDLC CMM 能力等级

S-SDLC CMM 定义能力等级来表示模型内单个过程域包含的安全实践专业目标以及通用目标达成情况。

1) 专业目标

安全实践的专业目标特指与安全实践相关的特定目标，可能包括一个或多个，不同安全实践的专业目标也各不相同。

以安全需求分析活动为例来说，该活动的专业目标可能包括（但不限于）：软件系统的每个版本/迭代都进行了安全需求分析活动，识别的安全需求都经过了开发团队、测试团队和安全团队的评审，识别的安全需求大都安排在当前版本/迭代中实现，未能安排在当前版本/迭代的安全需求有明确的版本计划。

S-SDLC CMM 定义了“实现”、“部分实现”以及“未实现”三个专用目标实现情况，含

义分别如下：

实现情况	时间相关证据	目标达成情况
未实现	实践相关的证据完全缺失	收集到的证据都表明目标未达成
部分实现	收集到实践相关的证据存在缺失	收集到实践相关的证据完整无缺失，但有部分证据表明目标未达成
实现	收集到实践相关的证据完整无缺失	每一项证据都表明目标达成

2) 通用目标和制度化特征

通用目标是指安全实践在制度化方面的特征，S-SDLC CMM 定义了“未执行”、“基本执行”、“计划跟踪”和“充分定义”四个通用目标，分别对应“安全实践是不完整的”、“安全实践是被动的、随机的”、“安全实践是主动的、正式的”以及“安全实践是正式的、规范的”四种制度化特征。

安全实践的制度化特征主要指有利于实现安全实践的组织环境的管理方式方法，并把这些方式方法作为规范而制度化，并巩固下来，有助于提供达成专业目标所需的稳定组织环境，这使得安全实践在有压力的情况下实施，其专业目标的达成也更具稳定性。安全实践通用目标的达成情况由制度化特征来确定。

制度化特征主要有：

- a) “安全实践是不完整的”、

- b) “安全实践是被动的、随机的”、
- c) “安全实践是主动的、正式的”
- d) “安全实践是正式的、规范的”

S-SDLC CMM 定义了“未执行”、“基本执行”、“计划跟踪”和“充分定义”四个安全实践通用目标，这四个目标存在递进关系并与以上制度化特征一一对应。



3) S-SDLC CMM 能力等级

能力等级适用于企业在单个过程域的改进达成情况。四个能力等级用数字 0 至 3 进行编号：“0-未执行”、“1-基本执行”、“2-计划跟踪”、“3-充分定义”

表 1 S-SDLC CMM 能力等级

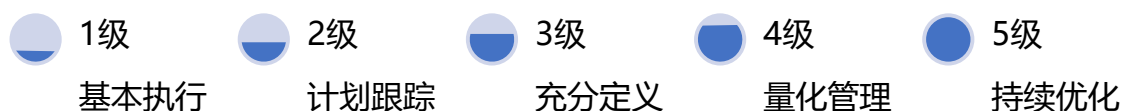
能力等级	专业目标达成情况	通用目标达成情况
0-未执行	➤ 有一个或者多个未达成	➤ 有一个或者多个安全实践未被执行 (制度特征：安全实践的执行是不完整的)
	未实现	未执行
1-基本执行	➤ 已达成	➤ 安全实践没有被计划和跟踪，而完全依赖个人的知识和努力。 ➤ 实践的执行没有制度来保障，执行结果难以维持。 (制度特征：安全实践的执行是被动的、随机的)
	部分实现	基本执行
2-计划跟踪	➤ 已达成	➤ 实践的执行是经计划和被跟踪的，其结果也会被验证。 ➤ 执行结果可以维持。 (制度特征：安全实践的执行是主动的、正式的)
	部分实现	计划跟踪
3-充分定义	➤ 已达成	✓ 实践的执行是经计划和被跟踪的，其结果也会被验证

		证。 ✓ 组织内建立了文档化的标准过程，实践的执行由组织级的标准来规划和跟踪。 (制度特征：安全实践的执行是正式的、规范的)
	实现	充分定义

3.2 S-SDLC CMM 成熟度级别

成熟度级别适用于组织内横跨多个过程域的过程改进达成情况，成熟度级别通过与预先定义好的过程域集合相关联的专业目标与通用目标的达成情况进行度量。五个成熟度级别用数字1至5进行编号。

每一级是一个层次，作为继续进行的过程改进的基础，由于每一成熟度级别都为下一级别打下必要的基础，因此，在成熟度级别上的跳级尝试往往会导致反效果。



具体如下表：

成熟度级别	级别定义	特征
-------	------	----

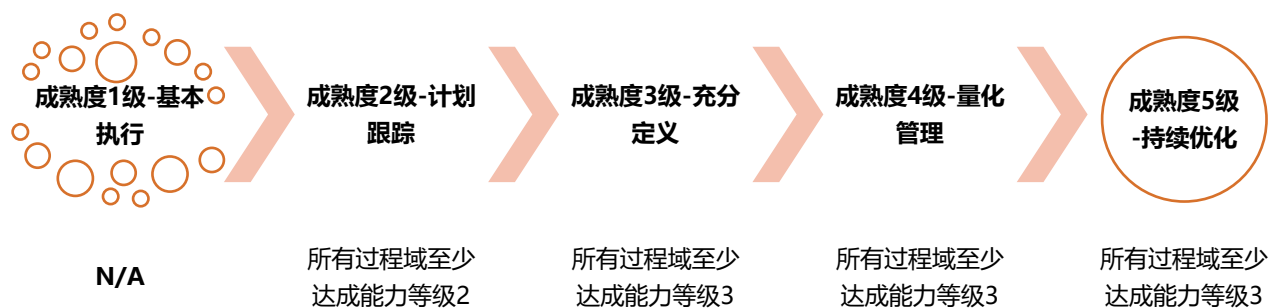
1-基本执行	✓ 安全过程域基本实践通常被执行 ✓ 基本实践的执行可能未经严格的计划和跟踪，而是基于个人的知识和努力。 ✓ 组织内的个人可标识出一个产品安全开发过程应被执行，并同意这个产品安全开发会在需要时执行。	随机、被动的安全开发过程
2-计划跟踪	在这一级别上，过程域基本实践的执行是经计划并被跟踪的，并对实践情况进行验证。产品安全开发管理应符合指定的标准和需求。通过测量来跟踪过程域的执行情况，因此，使组织能够基于实际实践活动进行管理。与非正式实践级别间的主要区别是过程实践被计划和管理。	主动、非正式的安全开发过程
3-充分定义	在这一级别，基本实践按照充分定义的过程执行。充分定义的过程是依据对文档化的标准过程进行裁剪并经批准的过程版本。这一过程与计划跟踪级的主要区别在于利用组织范围内的过程标准来管理和规划。	正式的规范的安全开发过程
4-量化管理	这个级别收集、分析执行的详细测量。这将获得	可量化控制的安全开发过程

	对过程能力和改进能力的量化理解以预测执行情况。这个级别执行的管理是客观的，数据安全管理的数量是量化的。这一级与充分定义级的主要区别在于定义的过程是定量的理解和控制。	
5-持续优化	在这个级别上，基于组织的商务目标并针对过程的有效性和执行效率建立量化执行目标。通过执行已定义过程和有创建的新概念、新技术的量化反馈来保证对这些目标进行持续过程改进。这一级与定量控制级的主要区别在于已定义的过程和标准过程基于对这些过程变化效果的量化理解，进行连续调整和改进。 安全过程可持续优化，实时跟踪行业的最佳实践和业务的最新动向，制度流程和技术工具持续调整以更好适应业务发展，沉淀下来的数据安全最佳实践能推广至行业供其他组织借鉴。	安全过程可持续优化，建立业界标杆

3.3 能力等级，成熟度级别关联关系

如果使用连续式表示法进行评估，那么评定就是一份“能力等级评定”。如果使用“阶梯法”进行评估，那么评定就是一个“成熟度级别评定”（例如成熟度级别2级）。无论是能力等级，成熟度级别都需要满足低级别的所有安全实践和目标才能进入到下一个级别等级。

表 2 S-SDLC CMM 成熟度级别与能力等级的对应关系



4 S-SDLC 软件安全成熟度级别评估

S-SDLC 软件安全成熟度级别评估是对软件服务提供者的综合实力的客观评价和确认, S-SDLC 软件安全成熟度级别反映了软件提供者从事软件开发服务安全保障能力的成熟程度。

4.1 评估方法

S-SDLC 软件安全成熟度级别评估方法 (简称: S-SDLC SSPCA) 是在深入研究了 CMM、CMMI、ISO/IECTR15504、ISO9000 等资料, 结合国内企业的实际情况, 以 CMMI 作为主要参考文件形成的。

评估方式为文件审查和人员访谈, 评估输出物为最终评估报告。评估小组使用 S-SDLC CMM 模型和遵循 S-SDLC 软件安全成熟度级别评估方法, 来指导评估和报告结果。这些评估结果被用于实施组织过程改进, 产生成熟度等级, 降低产品开发风险。

4.2 评估实施和评估

依据 LC 软件安全成熟度级别评估方法评估过程需执行所有的评估步骤, 建立正式评估小组, 并需要有相应经验或是授权的评估师领导评估组进行评估。

S-SDLC SSPCA 评估的目标是评定软件开发组织整体的成熟度级别, 主要流程如下:

- 1) 通过现场调研和资料查看等方式收集和整理有关安全实践的客观证据;
- 2) 基于客观证据评定安全实践的目标 (含: 专业目标及通用目标) 的实现情况;

专业目标的实现情况分为三类: 实现、部分实现以及未实现, 它们的含义分别如下:

- 实现: 收集到实践相关的证据完整无缺失, 且每一项证据都表明目标达成;

- 部分实现：收集到实践相关的证据存在缺失，或者收集到实践相关的证据完整无缺失，但有部分证据表明目标未达成；
- 未实现：实践相关的证据完全缺失，或者收集到的证据都表明目标未达成。

3) 基于专业目标和通用目标的实现情况评定安全实践的能力等级；

前文已经说明，安全实践的能力等级包括：0-未执行，1-基本执行，2-计划跟踪和 3-充分定义，它取决于专业目标和通用目标的实现情况，具体对应关系如下：

- 0-未执行：安全实践的专业目标中，至少一个或者以上“未实现”或者“部分实现”
- 1-基本执行：安全实践的所有专业目标都已“实现”
- 2-计划跟踪：安全实践的所有专业目标都已“实现”，且达成通用目标“计划跟踪”也已“实现”，即安全实践的执行是经计划和被跟踪的，其结果也会被验证，因而实践的执行是主动和正式的，其执行结果也得以维持。
- 3-充分定义：安全实践的所有专业目标都已“实现”，且达成通用目标“充分定义”也已“实现”，即安全实践的执行是经计划和被跟踪的，其结果也会被验证，而且在组织内建立了文档化的标准过程，安全实践的执行过程由组织过程而来，即实践的执行由组织级的标准来规划和跟踪。

4) 基于安全实践的能力等级评定各 S-SDLC 过程域的成熟度级别；

S-SDLC 过程域成熟度与安全实践能力等级的对应关系参考表 2.5

5) 基于各 S-SDLC 域的成熟度级别评定整个组织的成熟度级别。

组织的成熟度级别取各过程域成熟度级别的算术平均值。

4.3 评估制约因素

1. S-SDLC CMM 模型
2. 评估范围，包括待审查的过程域，以及待评估的成熟度级别或能力等级
3. 评估方法
4. 评估小组组长与小组成员
5. 从评估实体中选取的待访谈的评估参与人员
6. 评估输出（如：评定、特定于实例的发现）

第二部分：S-SDLC CMM 安全实践

此部分详细描述了 64 个安全实践的定义

1 监管

1.1 流程与政策

“流程与政策”包含建立安全开发流程、安全活动推广、高管培训、安全决策以及供应商软件安全方面的实践。

【3】公布流程（角色、职责和计划），并随着需要发展更新

战略上重视软件安全的软件开发组织，往往会建立一套指导安全软件开发的流程，以便软件安全活动有序、高效的开展。为此，他们根据组织自身需要定义安全活动，并在已有的软件研发流程（例如瀑布、敏捷或 DevOps）中植入这些安全活动。该流程应该包含清晰的角色职责和计划，还应该在组织内部公布该流程和相关计划，以便该流程得到干系部门的支持。另外，随着组织的发展和流程的运行，流程本身还应该不断更新。

【3】建立宣传角色并在组织内部开展安全推广活动

软件开发组织应该设定类似安全专员/安全倡导者的虚拟角色，安排其在组织内部宣传和推广安全软件开发流程。通过安全软件开发流程的宣传和推广，使研发内各部门（例如：开发部、架构部和运维部等）都能了解软件开发组织的软件安全战略及其相关的安全目标和流程，以便该流程得以顺畅的执行。应该在研发部门内选择项目经理、技术经理或者高级开发工程师等担任此虚拟角色，该虚拟角色可以提供软件安全的咨询服务，也可以在项目组内把控安全软件开发流程的质量。

【3】明确关卡位置，为通行决策收集必要信息

为了确保安全软件开发流程过程和质量，有必要在需求、设计、实现、测试和部署等阶段设定安全关卡。例如：是否有识别安全需求、是否有做安全设计、是否有做威胁分析、已知的高危漏洞是否都已解决等。有条件的软件开发组织可以通过软件研发平台建立软件安全质量门禁，只有通过质量门禁的软件版本才能发布。如果软件开发组织尚未建立工具自动化的质量门禁，应该安排 QA 等角色，对安全关卡进行人工检查。

【3】基于业务和数据的风险对软件系统进行分类，并建立对应的安全目标

一般来说软件系统安全目标定的越高越好，可以投入大量资源和人力做到“零已知安全漏洞”、“覆盖所有业务和产品线”。然而，这么做对软件开发组织来说代价高昂，科学的做法是以经济的方式实现“足够安全”。为此，软件开发组织应该根据业务性质、数据类别以及法律法规等，建立软件系统的安全分类标准，设立每一类安全标准所对应的安全目标。就业务性质来说，如果软件系统涉及到面向互联网的业务、客户量级大的业务、涉及金融和支付等容错性较低的业务等，需要设立更高等级的安全目标；就数据类别来说，如果软件系统的数据涉及到国家机密、商业机密或者个人隐私，也需要设立更高等级的安全目标。

【4】通过统一的 SDL 管理系统量化展现安全活动的执行

当软件开发组织 SDL 流程运转比较顺畅的时候，应该建立统一的 SDL 管理系统以量化展现安全活动的执行，以便在提升 SDL 流程效率的同时不断对其进行改进和优化。为此，首先需要建立可度量的 SDL 过程质量目标以及 SDL 结果质量目标；其次需要建立统一的

SDL 管理平台来统计、计算和展现这些度量目标；再次需要针对 SDL 过程及结果质量不达标的情况进行根因分析并制定相关的改进措施；最后应该定期评估 SDL 的度量目标，有必要则对其进行更新。

1.2 合规性

“合规性”主要包括解读法律合规文件以及参与行业合规安全文件制定等安全实践。

【3】将合规性法律文件转化为安全需求进入版本开发计划中

软件开发组织的软件产品都面临行业法律、法规以及监管政策的约束，为了使得软件开发组织的软件产品合法、合规，软件开发组织应该安排专人或部门，负责解读国家、行业的法律合规文件，并将其转化为安全需求，然后纳入软件产品的开发计划中。

【3】建立外部采购成品软件的安全控制制度

近年来针对软件供应链的安全攻击事件一直呈快速增长态势，造成的危害也越来越严重。因此供应商软件产品的安全性对软件开发组织至关重要。软件开发组织与软件供应商合作时，应该在软件采购合同中明确提出软件安全方面的要求，如：必须提供渗透测试报告或者源代码安全审查报告。对于合作开发的软件产品项目，还应该从需求、设计、实现及测试等过程中提出软件安全要求。在验收供应商的软件产品时，应该让供应商提供代码审核报告、安全测试报告等安全性证明。

1.3 培训

“培训”包含高管软件安全意识的培训以及产品、开发、测试、架构和运维等角色的软件安全能力的培养，其目的是使得高管重视软件安全，相关部门及岗位的人员具备软件安全开发所需的必要知识和技能。

【3】对高管进行安全软件意识培训

高管需要了解软件安全的重要性、缺乏软件安全的危害以及行业最新趋势等，这是高管在制度和资源等方面支持软件安全的必要条件。安全部门需要与人力资源部门合作，一方面建立和不断更新面向高管的软件安全培训课程以及相关的考题，另一方面在高管的入职培训以及入职后的定期培训中，增加软件安全方面的培训和考核。

【3】提供基于角色的高级课程

软件开发组织建设安全软件开发能力，离不开研发、产品、安全和运维等部门的各种角色。应该针对不同角色，建立相关软件安全培训与考核机制，使他们具备足够的软件安全知识与技能，进而支撑软件开发组织的安全软件开发。安全部门应该与研发部门、人力资源部门以及运维部门等部门进行协作，开发软件安全相关的培训课程，在入职培训以及入职后的定期培训中增加软件安全方面的课程及考核。

【3】根据公司经验教训建立并使用特定的培训材料

安全测试、渗透测试或者攻防演练等活动中发现的安全漏洞，或者运营过程中发生的安全事件，都是软件开发组织重要的知识和经验，对它们进行总结和复盘，形成特定的培训材料，发布给相关角色，将有助于提升他们安全意识和技能，而且这种安全意识和技能将更贴近实际。

【3】聘请外部专家强化项目团队的安全开发能力

外部专家能够带来新的安全思想、安全技术以及安全实践，定期聘请外部专家来对组织内进行培训，有助于软件开发组织拓展认知格局，紧跟软件安全发展的最新趋势，提升团队的安全开发能力，进而提升产品的核心竞争力。

【4】建立针对研发角色的安全定级以及相关的培训考核机制

统一安全培训而不区分角色可能造成培训资源浪费或者部分角色的培训深度不够。为了使得软件安全培训更加高效且具有针对性，软件开发组织应该建立针对研发角色的安全定级以及相关的培训考核机制。应建立不同的安全能力等级（例如：初级安全能力、中级安全能力和高级安全能力等），建立各研发角色与安全能力等级的对应关系（例如：初级开发工程师要求具备初级安全能力、高级开发工程师要求中级安全能力、架构师要求高级安全能力等），建立每个安全能力等级的培训和考核机制。

2 能力

2.1 攻击模型

“攻击模型”从攻击者的视角分析分析和研究可能存在的攻击者、攻击模式和相关技术，以便软件开发组织能够防患未然。

【3】构建与特定业务及特定技术有关的攻击模式

软件开发组织应该识别潜在的攻击者及其特定技术有关的攻击模式，以便了解有哪些攻击者，攻击者的动机、能力和攻击方式。软件开发组织可以安排安全、架构、开发和运维等角色组成虚拟团队专门负责研究攻击模式。

【3】收集和发布攻击案例

不论是通过内部渠道还是外部渠道，软件开发组织都应该收集各种攻击案例，并从软件系统、攻击方式和攻击影响等维度对攻击案例进行分类，并发布给相关部门，以便他们可以学习、交流，并为安全设计提供支撑。发布攻击案例应该做好权限控制，确保每个攻击案例只发布给那些真正需要的人员。

2.2 安全特性与安全设计

“安全特性与安全设计”包括开发安全组件、建立威胁库和安全设计库等实践，主要为软件开发组织的安全设计过程提供支撑。

【3】构建并发布安全特性

软件开发组织应该建立专门的团队，构建并发布安全开发组件供产品研发团队使用，以提升安全开发效率。安全开发组件示例有：身份验证、角色管理、密钥管理、日志记录、密码、协议、防 SQL 注入解决方案、防 CSRF 解决方案、会话保持机制等。

【3】建立相对完善安全设计库

软件开发组织应该针对软件系统常见且重要的安全需求，建立安全设计库，以降低开发人员安全设计的难度。例如：给认证、防注入、文件操作和安全传输等重要的安全需求提供安全设计建议，设计不可猜测的用户 ID，调用后台接口需要用户凭据（Token）、避免拼接 SQL 语句、校验文件类型/大小/文件名等。

【3】建立相对完善的威胁库

威胁分析可以从攻击者的视角发现软件系统存在的潜在风险。基于软件系统的类别和系统架构建立威胁库将有助于威胁分析活动。需要收集和整理相关的威胁库，以便研发和安全软件在架构设计或识别安全需求的时候，对软件系统进行快速地威胁分析。

2.3 标准与要求

“标准与要求”包含定义安全需求基线、安全编码规范、代码安全审核、安全测试标准等实践，主要目的是为各阶段的 SDL 活动设定标准。

【3】建立一个有关安全信息的内网站点

企业应该建立安全门户网站以便相关角色获取软件安全的最新信息，内容可以包括：SDL 及相关的流程规范、软件安全培训课程、软件安全最新趋势等。应该安排人员在组织内推广该网站，使相关研发角色了解该网站并从其中获取安全信息。还应定期评估该网站的内容，有必要则对其更新。

【3】定义安全需求基线

安全需求基线指的是软件开发组织内所有软件系统开发项目所要遵循的安全相关的需求和要求。不同的软件系统安全需求基线应该有所差别。软件开发组织制定的安全需求基线应该包含：通用安全需求、软件系统特定安全需求、法律法规以及行业监管等内容。针对软件系统特有的安全需求基线，应该考虑到开发语言和所使用的技术框架。

【3】定义安全编码规范

软件开发组织应该根据自身软件系统所使用的开发语言和开发框架，制定相应的安全编码规范。安全编码规范包括但不限于：防止安全漏洞的安全编码习惯（例如：防 SQL 注入、防 XSS），开发语言相关的安全编码规则，开发框架相关的安全编码规则（例如：Spring、Mybatis）。安全编码规范应该包含软件开发组织使用的开发语言和技术框架。

【3】定义代码安全审核的标准

代码安全审核指的是在软件项目开发过程中，每次代码转测前对当次增量代码进行安全审

核，以确保新增代码遵循安全编码规范，并发现新增代码可能引入的安全漏洞；代码安全审核可采用人工检查或人工与自动化工具检查相结合的方式开展。软件开发组织应该建立代码安全审核的标准，主要包含与安全编码规范的对应的审核内容，以及审核流程、角色及职责等。

【3】定义第三方开源组件分析活动的标准

第三方组件安全是软件开发组织软件安全的重要内容，软件开发组织应该定义第三方开源组件分析活动的标准，内容应该包括定义不同漏洞的风险等级、定义各种开源组件许可协议相关的法律风险等级，分析活动的角色、职责和通过标准等。

【3】定义安全红线

安全红线指的是软件系统必须在线上修复的漏洞类型列表。软件开发组织应该为每一类软件系统定义不同的安全红线，且在研发流程中规定，所有触犯红线的漏洞必须修复完成后软件版本才能发布。

【3】定义安全测试标准

安全测试是保障软件系统安全的重要手段，软件开发组织应该定义安全测试的标准，内容包括但不限于：流程上要求针对安全需求设计安全测试用例，确保安全用例对安全需求的覆盖，要求对测试发现的漏洞进行全生命周期的跟踪管理，确定安全测试活动的角色、职责和通过标准等。

【3】定义数据库、中间件和操作系统的配置基线

基线是一个系统的启动配置，当系统部署在有安全配置基线的状态时将更安全。为确保数据库、中间件和操作系统等软件环境的安全，软件开发组织应该为用到的数据库、中间件和操作系统定义安全配置基线，该基线在生产环境中验证有效后应该固定下来。软件开发组织还需要定期评估安全配置基线，有需要则对其进行更新。

【4】建立并使用内部完整定义的第三方组件库

有条件的情况下，软件开发组织应该严格管控第三方组件库。软件开发组织需要安排专门的团队（例如：第三方组件治理团队）按照正式的流程建立、更新和维护内部的第三方组件库。组件库应该能满足所有产品线的开发需求，如果组件缺失，有正式的申请、批准和新增组件流程。应该管控各个开发团队的组件使用行为，使得开发只能从内部组件库下载使用三方组件，禁止通过互联网下载或者私自拷贝等方式引入第三方组件。应该定期对第三方组件库进行安全扫描，以便发现安全隐患能及时整改。

3 开发过程接触点

3.1 需求分析

需求分析阶段完成系统功能的安全需求分析，识别可能面临的安全风险。

【3】为每一个重要的开发功能匹配安全需求

安全基线和安全红线分别规定了软件系统在安全、合规以及漏洞方面的要求，软件开发组织需要按照既定的流程和方法，以安全基线和安全红线为基础，识别每一项业务需求/用户故事在安全、合规以及漏洞方面所面临的约束，并将这些约束转化为安全需求。安全需求例子可能是：为新增的重要业务功能采用多因素的方式来认证身份，收集敏感数据时弹窗提示用户所收集的数据及其用途，为 WEB 应用订单提交增加预防 WEB 安全漏洞功能等。安全需求应该作为软件质量的一部分来看待，即不能把安全需求与其关联的业务需求/用户故事割裂开来。建立并维护一套安全需求知识库，将有助于团队准确而高效的识别安全需求。

3.2 安全设计

安全设计活动，包括在架构设计时考虑系统的安全性，对系统进行威胁分析，评估系统面临的风险，对设计方案进行安全检查和评审。

【3】对重要软件系统进行架构安全设计

对软件系统进行架构设计，不仅要考虑性能、高可用和扩展性，还要考虑安全性。在基

基础设施安全层面，要考虑是否需要以及如何部署 WAF（应用防火墙）、RASP（运行时应用自我保护）、数据库防火墙和业务风控系统等；在应用安全层面，考虑如何接入 SSO 系统、权限管理系统、密钥管理系统以及统一日志管理平台等。对于云原生应用，架构设计还需要考虑如何利用供应商提供的基础安全能力。

【3】分析设计是否满足已知的安全需求

安全需求由业务需求/用户故事面临的安全、合规和漏洞方面的约束转化而来，它们与常规需求一样，需要对其进行设计以便后续开发实现。安全设计的例子可能是：为重要的操作（转账、支付等）设计不可猜测的用户 ID，对于用户输入设计相应的校验规则或者引入安全开发组件。建立和维护安全设计知识库有助于开发团队高效进行安全设计活动。安全设计完成后，应邀请资深开发工程师、架构师或者安全工程师对设计方案进行评审，以确保安全设计满足已识别的安全需求。

【3】对软件系统/版本/迭代进行威胁分析

威胁分析是指从攻击者的视角来识别软件系统可能面临的威胁，例如：身份认证系统是否有可能被绕过，管理员默认密码未修改等、授权和访问控制系统是否存在缺陷导致越权以及是否存在独立可审计的操作日志等，然后对已识别的威胁设计缓解方案。基于软件系统架构以及系统之间的数据流图进行威胁建模，有助于软件开发组织从系统、模块、业务场景等方面分析可能面临的风险，并对风险提出相应的解决方案或缓解方案。建立和维护一个用于威胁建模的知识库有助于提升威胁分析的质量和效率，威胁知识库可以根据软件系统的架构、与其他系统的交互以及历史安全漏洞总结出常见的威胁，然后对威胁进行分类并设计缓解方

案。

3.3 安全开发

安全开发阶段包括对代码和第三方组件进行人工和自动化的审核, 落实安全编码规范的实施。

【3】对软件系统执行人工安全代码审核

软件开发组织应对代码进行人工检查, 以便发现自动化工具难以识别的漏洞。对于逻辑类漏洞, 例如越权操作、文件上传下载等, 自动化工具难以明确漏洞是否真实存在, 因此需要人工进行确认。除此之外, 对于一些关键功能的代码也可使用人工进行检查, 包括加解密方式、敏感数据操作等, 结合审查人员对系统业务的了解, 可以发现一些比较隐蔽的安全漏洞, 进一步保证编码的安全性。

【3】对软件系统执行自动化静态代码安全扫描

采用自动化的静态代码扫描工具, 能够高效率的对代码进行安全检查。一些常见的安全漏洞, 比如 XSS、SQL、XXE 等, 可以使用自动化工具检测。敏捷开发环境下, 软件开发组织应该将自动化静态代码扫描工具与代码管理平台 (GitLab、SVN 等) 或 DevOps 平台集成, 以实现提交代码合并请求时自动触发安全检查, 发现高危漏洞则拒绝合并请求, 进而帮助开发人员快速修复安全漏洞。

【3】对软件系统执行自动化第三方组件安全扫描

软件开发需要用到大量的开源第三方组件，对这部分组件的安全也应引起重视。对于第三方组件的安全，重点关注组件本身的已发现安全漏洞和使用的许可协议。由于第三方组件的数量较大，为快速定位有问题的组件，软件开发组织应采用自动化的方式对开源组件进行管理，发现的问题可录入漏洞管理平台进行跟踪管理。

【4】安全编码规范通过自动化工具实现审查

软件开发组织应对安全编码规范的落地做实际上的审查。利用商用扫描工具等自动化方式对代码进行扫描可提高审查效率，但对于业务复杂、安全编码规范细致的软件系统，商用工具的默认扫描规则并不能很好的匹配编码规范，需要软件开发组织进行自定义扫描规则。作为更进一步的考虑，可自研扫描工具更好地符合软件开发组织自身的安全需求。

3.4 安全测试

安全测试阶段包括在一系列产品测试中进行安全特性驱动测试，包括功能安全性测试等，并对测试发现的安全漏洞进行跟踪管理，确保问题得到解决。

【3】对软件系统进行安全需求和安全特性驱动测试

安全需求和安全特性驱动测试主要是为了验证需求分析阶段识别的安全需求是否得到正确实现。与常规的功能测试一样，软件开发组织应该安排测试人员设计测试用例、执行测试

以及管理和跟踪 bug/漏洞。建立和维护一套安全测试用例库，将有助于提升测试效率。对于云原生、中台等新型技术架构的软件系统，则需要针对其安全特性设计合适的安全测试方案。

【4】利用自研的安全测试工具提升测试效率

软件开发组织可考虑研发安全测试工具，以便更有效、全面地对软件系统进行测试。对于安全能力成熟度较高的软件开发组织，通用的解决方案已无法满足复杂的安全需求，须针对软件开发组织自身技术架构，结合实际业务研发合适的安全测试工具。此项活动对企业的安全能力要求较高，需要依赖企业整体的安全能力建设。工具能提高安全测试的效率，尤其自动化方式，在采用敏捷方式的企业中更有意义。

4 部署&运维

7.1 渗透测试

渗透测试活动包括对软件系统进行渗透测试，发现安全漏洞，并对发现的漏洞进行管理和解决。

【3】上线前对重要软件系统进行渗透测试

不同于开发过程中的安全测试，渗透测试是指从攻击者的视角，采用一系列工具，尽可能多的发现软件系统的漏洞，尝试利用这些漏洞，评估他们的影响，最终形成渗透报告，以供软件开发组织完善软件系统。渗透测试通过模拟黑客对系统的攻击，还原真实环境可能面临的安全风险，不仅能发现软件本身的安全漏洞，还能对软件运行时环境进行安全检查，包括操作系统、中间件、网络环境等，以期发现最终配置中的漏洞。渗透测试作为上线前最后一道安全测试，是安全软件开发工作一次重要的验证和把关。

【3】渗透测试问题录入漏洞管理平台跟踪管理

软件开发组织应对渗透测试问题进行全生命周期管理来确保问题得到有效修复。渗透测试作为上线前最后一道安全测试，其发现的问题应引起足够的重视。一个比较好的方法是通过安全漏洞管理系统反馈给开发人员，由开发人员进行修复，测试或安全人员进行有效性的验证。大多数情况下，存在渗透测试问题的系统应谨慎上线或推迟上线，直至问题修复。

7.2 软件环境

软件环境活动主要确保公司网络的安全，对个应用程序的输入进行监控，部署安全基础构件保护内网安全。

【3】确保主机和网络的安全基础构件配备到位

企业的网络大致可分为外部网络、办公网络、生产网络和生产网络（敏感区）。软件开发组织应该分别针对办公网络、生产网络和生产网络（敏感区）设置响应的基础网络安全构件，例如统一接入网关、WAF（应用防火墙）、IDS（入侵检测系统）、IPS（入侵防护系统）、统一日志平台以及抗 DDos 系统等，以确保身份认证、授权与访问控制、行为监控、行为审计以及资产保护等功能正常运行。

【3】对数据库、中间件和操作系统根据安全配置基线进行加固

为保证数据库、中间件和操作系统得到安全的安装和配置，软件开发组织应为其制定安全配置基线。安全配置基线即该软件系统必须满足的最基本的安全要求。安全基线的制定可围绕系统漏洞、配置的脆弱性等方面开展（例如信息泄露、蠕虫漏洞，端口的开启与关闭、口令配置等）。定期对数据库、中间件和操作系统的基线进行检查，评估检查结果，以便安全基线随着技术升级进行更新。

【4】统一软件开发组织的操作系统、数据库及中间件

面对不断发展更新的操作系统、数据库和中间件，软件开发组织应对此建立清单进行统一管

理以降低安全风险。各产品线一般基于自身业务情况选择技术组件，例如 Windows 或 Linux，MySQL 或 Oracle。随着产品线的扩增，技术组件的选择容易变得多样化和无序，对此有必要规范技术组件的使用，例如定期统计各产品线对操作系统、数据库和中间件的需求，评估归并版本，要求各产品线采用归并后的版本，并定期更新技术组件信息库。

7.3 漏洞管理

漏洞管理阶段包括使用工具对漏洞进行全生命周期的管理，并追溯到研发之前的阶段。

【3】使用漏洞管理工具，确保安全漏洞有完整生命周期管理

为实现漏洞的完整生命周期管理，软件开发组织应建立漏洞管理平台。通过将各个阶段发现的安全漏洞录入漏洞管理平台，各相关人员能共同参与漏洞的修复，例如研发人员对漏洞相关功能进行重新编码，测试人员验证漏洞修复方案的有效性等，促进跨部门的协作，且项目管理人员能够清晰地了解漏洞的修复进展情况，完成漏洞的闭环管理。

【3】确认在运维阶段发现的安全漏洞，并追溯到研发阶段

对于运维阶段的安全漏洞，软件开发组织应探究其在整个研发流程的成因，避免或减少出现类似的情况。一个比较好的方法是将漏洞录入漏洞管理平台，由各阶段的相关人员对漏洞进行管理解决，而不仅仅是运维人员参与修复。通过对漏洞的闭环管理，将有助于完善整个 S-SDLC 流程，提升各个阶段参与人员的安全能力。

7.4 运营支持

运营支持阶段包括对敏感行为的监控处理，定期审计各类系统运行日志，管理第三方软件的安全漏洞和补丁修复。

【3】实时监测敏感行为

软件开发组织应实时监测程序的运行情况，以便及时发现安全风险。为此可以建立风控系统或采取风险管理措施对应用程序的运行进行风险识别，基于业务场景建立敏感行为清单，对敏感行为进行实时监控。应用程序层次上的入侵检测和异常检测系统可能会聚焦于应用程序与操作系统（通过系统调用）之间的交互，或者应用程序与其所使用的、操作的数据类型之间的交互。因此应基于程序的业务场景进行风险识别规则的制定。

【3】对系统日志进行审计

软件开发组织应规范系统日志的管理，定期对日志进行审计。日志一般会记录系统状态、用户 IP、操作业务、访问日期等信息，通过对日志进行审计，有助于发现程序自身的安全缺陷，乃至为安全事件发生后的成因分析提供线索。对于某些关键系统或业务，还可以基于审计日志构建日常例行的大数据分析与事件挖掘活动，主动发现未告警的安全事件或隐患。综合各种监管要求，一般至少需要保留六个月的操作日志，用于追溯。

【3】建立第三方软件安全检测、补丁修复以及应急预案等机制与流程

要想更全面地保证软件运行时的安全，除了关注应用程序本身和操作系统外，软件开发组织应持续关注第三方软件的安全态势，主动地修复相关缺陷。为此可以定义一套明确的流程机制，来管理第三方软件的安全缺陷和补丁更新，例如制定补丁的登记分类、缺陷的影响评估、测试等相关的管理规范。事前预防胜过事后补救，软件开发组织应从各种渠道监视外部威胁情报，评估并处理第三方软件可能产生的安全风险。特殊情况下，对于已经发生的影响较大的安全漏洞，启动应急预案，将其作为信息安全事件处理解决。

7.5 应急响应

应急响应阶段包括对外部威胁情报的收集，应对信息安全事件的流程机制，以及相关的应急响应团队建设。

【3】持续收集威胁情报数据

软件开发组织应从多种渠道收集威胁情报数据，以便及时对信息安全事件做出响应。采集情报数据的渠道可选择专业安全公司或开源威胁数据等，形成威胁指标或数据检测模型。为提高采集的规模和效率，可部署态势感知系统，实时监测内网环境，对异常行为及时告警。建立网上安全应急响应中心，接收外部白帽子提交的安全漏洞信息，能更精确的发现系统面临的威胁。

【3】建立正式的安全响应团队

信息安全事件发生时, 软件开发组织应有安全响应团队依照明确的应急响应流程对事件进行处理和解决。该团队负责定义、收集、评估事件响应过程中的指标, 并根据指标对安全事件做出响应处理。团队可定期组织安全事件的应急演练, 演练结果可作为流程不断改进的依据。团队宜与其他部门建立联络通道, 以便安全事件发生时进行跨门联合工作, 快速定位解决问题。

【3】制定明确的安全响应流程

软件开发组织应依照明确的流程机制应对信息安全事件, 避免采取临时应急措施导致事件影响进一步扩大。例如对事件进行分类分级, 制定相应的管理计划; 与内部或外部软件开发组织保持适当的关系和联络, 建立沟通渠道和信息共享策略。安全响应团队应对流程的各个环节应做好文档记录, 包括处理者和负责的事项, 事件的成因分析等, 并纳入公司信息