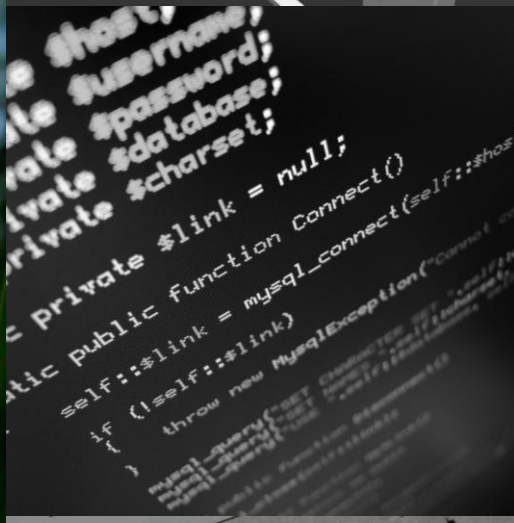
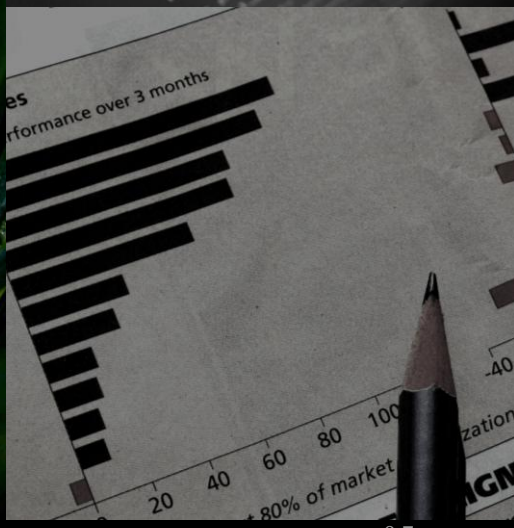




v0.7

企业软件保证成熟度模型

目录

简介 **2**
- 动机和基本原理 2
- 角色定义 2
- 目标受众 2
- 使用成熟度模型 3

成熟度模型 **4**

调整与管理
- 培训与指导 6
- 标准与遵从 10
- 战略规划 14

需求与设计
- 威胁建模 18
- 安全要求 22
- 防御设计 26

验证与评估
- 体系结构审查 30
- 代码审查 34
- 安全测试 38

部署与操作
- 漏洞管理 42
- 基础架构强化 46
- 操作实现 50

路线图 **54**
- 独立软件供应商 56
- 在线服务提供商 58
- 金融服务机构 即将推出
- 政府机构 即将推出

案例研究 60
- VIRTUALWARE - 独立软件供应商 **62**
- MOOGLE - 在线服务提供商 即将推出
- UNIGROUP - 金融服务机构 即将推出
- SOCIALLINK - 政府机构 即将推出

简介 *2*

简介

成熟度模型的背景信息

动机和基本原理

该软件保证计划的成熟度模型是根据一些尚在理论阶段的指导原则和目标构建的。

首先，要正确构建保证计划，组织就需要执行各种不同的措施，而改变组织的一贯行为很难。要在不增加间接成本的前提下高效地构建保证计划，必须在每个改进阶段明确定义短期目标，并逐步完善长期目标。

第二，必须有相应的标准来衡量保证计划的改进效果，且该衡量标准应简单易懂。若措施的定义含糊不清，会导致难以在正常情况下使用成熟度模型，并且使用时很容易出错。

第三，一个保证计划无法用于所有类型的组织。应该针对每个组织所面临的特定风险，根据软件的构建方式和业务用途来构建相应的保证计划。因此，成熟度模型应该根据业务类型来定义改进方案，并使用建议的路线图及案例分析来阐明常见用例。

角色定义

开发人员

负责软件的低级设计和实施的人员

架构师

负责高级设计和大型系统工程技术的人员

管理人员

负责日常管理开发人员的人员

QA 测试人员

负责软件的常规测试和发布前验证的人员

安全审核人员

具备软件技术安全知识的人员

企业所有者

负责对软件及其业务要求做出关键决策的人员

运营支持人员

负责提供客户支持或直接技术支持的人员

目标受众

本文档介绍了与软件生产中众多角色相关的信息，可供各类人员使用。采用以下建议可以简化您开始使用 SAMM 的过程

管理人员、企业所有者、安全审核人员、架构师

对于希望了解如何领导软件保证计划和管理计划进展的人员，应该先阅读“成熟度领域”一节中每页的概述，以了解高级安全功能。掌握概要信息对于之后的阅读非常有用，您随后可以仔细阅读与您的组织相关的路线图和案例研究。如果需要，您可以返回到“成熟度领域”一节以了解特定功能和目标的详细信息。

开发人员、架构师、QA 测试人员、安全审核人员、运营支持人员

若要了解如何在项目级别执行特定措施以改进安全功能的置备过程，可以选择阅读“成熟度领域”一节中介绍“目标”和“措施”的部分。如果需要，可以参阅相关的目标和其他资源，以了解背景和上下文信息。要了解功能将来的进一步改进，还应该查看后续目标。

使用成熟度模型

该成熟度模型介绍了组织用以降低安全风险并提高保障水平的各种措施。在最高级别，包含代表一般措施的以下四种**规范**：

在每个规范下，按照三种**功能**进行分组，其中每种功能分别代表组织能够逐步改进的安全相关措施的范围：

每种功能都有三个连续的**目标**，这三个目标为组织定义了针对指定功能的逐渐复杂的目标。依照本文档中的指导，组织可以实现一个目标。实现第一个目标之后，可以继续实施指定功能的下一个目标。一般而言，每个功能下的连续目标代表：

大多数组织已执行了其中的某些措施，因此，必须执行初步评估，以便根据已完成的目标为每个功能进行评分(0-3)，将组织的绩效与十二种功能关联起来。之后，组织应该根据业务驱动因素和特定于组织的信息重复执行这一过程，方法是选择要改进的功能，然后完成每个功能的下一个连续目标。

成熟领域

本小节将定义组成成熟模型的构造块。本小节包含四大规范，每条规范下具有三项安全功能，用来定义可执行安全相关活动的范围。每项功能又有三个级别的设置，以及相应的活动、成功标准、人员开销和相关成本。

调整与管理

培训与指导 6
1: 为开发人员提供有关安全编程和安全部署的资源访问方式。 7
2: 对软件生命周期中的所有人员进行培训，并就安全开发提供针对特定角色的指导。 8
3: 实施全面的安全培训，并检验员工对基本知识的掌握情况。 9

标准与遵从 10
1: 了解针对组织的相关管理规定和行为要求 11
2: 制定安全基准和行为基准，并了解每个项目的风险。 12
3: 要求遵从标准，衡量项目是否符合组织的标准和策略。 13

战略规划 14
1: 针对组织内的软件安全性，制定统一的战略性方案。 15
2: 了解数据和软件资产的相对价值，并选择风险承受水平。 16
3: 使安全开支与相关业务指标和资产价值相符。 17

需求与设计

威胁模型 18
1: 确定和了解对组织和个别项目的高级威胁。 19
2: 提高威胁评估的准确性，更深入地了解每个项目的细节。 20
3: 将补偿控制与对内部和第三方软件的每种威胁具体联系起来。 21

安全要求 22
1: 在制定软件要求时，明确地将安全性考虑在内。 23
2: 根据应用程序特定的风险以及滥用情况，制定安全要求。 24
3: 强制所有软件项目和第三方相依性都符合安全要求。 25

防御性设计 26
1: 向项目团队提供超前预防的设计指导，并加强项目周边安全。 27
2: 在设计过程中，开发可全面增强安全性的软件。 28
3: 评估项目团队的超前预防措施，使防御性设计更高效。 29

验证与评估

体系结构审查 30
1: 支持对软件体系结构进行专门审查，以确保已基本排除了已知风险。 31
2: 提供评估服务，根据全面的安全最佳实践审查软件设计。 32
3: 要求实施评估和验证设计，以详细了解保护机制。 33

代码审查 34
1: 随机查找基本的代码级漏洞和其他高风险的安全问题。 35
2: 通过自动审查，在开发过程中更准确、更高效地执行代码审查。 36
3: 强制执行全面的代码审查流程，查找语言级的风险和特定于应用程序的风险。 37

安全测试 38
1: 启用测试流程，以根据软件要求进行基本安全测试。 39
2: 通过自动测试，在开发过程中更全面、更高效地执行安全测试。 40
3: 要求执行特定于应用程序的安全测试，以确保部署前的基准安全。 41

部署与操作

漏洞管理 42
1: 了解关于响应漏洞报告或事件的高级规划。 43
2: 阐述对响应流程的期望，以改善一致性和通信有效性。 44
3: 改进响应流程中的分析和数据收集过程，为预先规划提供反馈信息。 45

增强基础架构 46
1: 了解应用程序和软件组件的基准操作环境。 47
2: 通过增强操作环境，提高应用程序操作的保密性。 48
3: 依照已知的最佳实践，验证应用程序运行状况和操作环境状态。 49

操作实现 50
1: 实现开发团队与关键安全数据操作员之间的交流。 51
2: 通过制定详细的操作步骤，提高对持续安全操作的期望。 52
3: 强制进行安全信息的交流并验证设计的完整性。 53

培训与指导

培训与指导功能专注于让软件生命周期中涉及的所有人员掌握应用程序安全性的相关知识。通过更快捷地掌握更多信息，项目团队就能更好地预先确定并抑制组织所面临的特定安全风险。

为顺利实现所有目标，一项重要的改进方案是通过讲师授课或电脑授课对员工进行培训。随着组织在这些阶段中不断进步，确立了广泛的培训，先培训开发人员，继而推广至组织内更多的角色，最后对担当不同角色的员工进行认证，以确保他们完全掌握培训内容。

除了培训以外，该功能还要求将安全相关信息归纳为操作准则，作为员工的参考信息。这就为建立组织的安全实践基准目标奠定了基础，然后实施这些准则，就可以不断地提高安全性。

	EG 1	EG 2	EG 3
目标	**为开发人员提供有关安全编程和安全部署的资源访问方式。**	**对软件生命周期中的所有人员进行培训，并就安全开发提供针对特定角色的指导。**	**实施全面的安全培训，并检验员工对基本知识的掌握情况。**
措施	• 对员工进行技术安全意识的培训 • 制定和维护技术指导制度	• 对员工进行针对特定角色的应用程序安全培训 • 聘用安全指导员，增强项目团队	• 创建应用程序安全支持门户网站 • 建立基于角色的考试/认证制度
结果	📖提高了开发人员对最常见的代码级问题的安全意识 📖使用基本的安全最佳实践来维护软件 📖为技术人员设定了安全操作的基准 📖根据基准安全知识，启用定性安全检查	📖全程关注导致产品、设计和代码级安全漏洞的问题。 📖制定现行项目中漏洞和设计缺陷的补救计划 📖在需求、设计和开发阶段，启用定性安全检查点 📖加深了对安全问题的了解，促进实施更主动的预先安全规划	📖有效地补救现行代码库和旧版代码库的漏洞 📖快速了解和抑制新的攻击和威胁 📖判断员工对安全问题的认知度，并根据通用标准进行评估 📖为提高安全意识，制定公平的激励制度

培训与指导

EG 1

为开发人员提供有关安全编程和安全部署的资源访问方式。

措施

对员工进行技术安全意识的培训

对技术人员进行应用程序安全性基本原则的安全培训，培训讲师可以是组织内部人员，也可以是外部人员。通常，培训可通过两种方式进行：一种是讲师讲授 1-2 天的培训课程；另一种是电脑模块化培训，每位开发员工以等量的学时学习各个模块内容。

课程内容应既包含概念信息，也包含技术信息。适当的主题包括有关输入验证、输出编码、错误处理、日志记录、身份验证、授权的高级最佳实践。培训内容还应包含一般软件漏洞，如 OWASP 排名前 10 位的 Web 应用程序，或为其他范例（嵌入式设备、客户端与服务器应用程序、后端事务处理系统等）开发的软件的更合适的修改列表。在可能的情况下，使用以适用的特定编程语言编写的代码示例和库练习。

开始进行这种培训时，建议强制进行年度安全培训，然后根据开发人员的数目进行授课（讲师教授或电脑模块化教学）。

制定和维护技术指导制度

对于开发人员，建立一个可提供特定于技术的安全建议的有效文件、网页和技术注释的列表。这些参考资料可从互联网上许多可用的公开资源中收集。如果开发环境中涉及到许多非常专业的技术或专利技术，可由深谙安全问题的资深员工随时编写安全记录，采用即时方式来创建此知识库。

确保管理层了解相关资源，并为员工介绍其预期用途。尽量简化和更新参考列表，以避免混乱、不切题。建立一种舒适度后，技术参考资料可用作定性检查表，以确保在开发阶段已阅读、理解和遵循该指导。

结果

- 提高了开发人员对最常见代码级问题的安全意识
- 使用基本的安全最佳实践来维护软件
- 为技术人员设定了安全操作的基准
- 根据基准安全知识，启用定性安全检查

成功标准

- 过去 1 年中，50%以上的开发人员简要介绍了安全问题
- 过去 1 年中，75%以上的高级开发人员/架构师简要介绍了安全问题
- 首次培训 3 个月内，启动技术指导

成本

- 培训课程的扩充或许可
- 技术指导的持续维护

员工

- 开发人员（1-2 天/年）
- 架构师（1-2 天/年）

相关目标

- 标准与遵从-2
- 安全要求-1
- 防御设计-1

相关标准遵从

- PCIv1.1，第 6.5 节
- PCIv1.1，第 12.5.1 节
- PCIv1.1，第 12.6 节

EG 2

培训与指导

对软件生命周期中的所有人员进行培训，并就安全开发提供针对特定角色的指导。

结果

- 全程关注导致产品、设计和代码级安全漏洞的问题。
- 制定现行项目中漏洞和设计缺陷的补救计划
- 在需求、设计和开发阶段，启用定性安全检查点
- 加深了对安全问题的了解，促进实施更主动的预先安全规划

其他成功标准

- 过去 1 年中，60%以上的开发人员接受了培训
- 过去 1 年中，50%以上的管理人员/分析师接受了培训
- 过去 1 年中，80%以上的高级开发人员/架构师接受了培训
- 对于培训课程的有用性，大于 3.0 Likert

其他成本

- 培训库的扩充或许可
- 深谙安全知识的员工的实际训练

其他员工

- 开发人员（2 天/年）
- 架构师（2 天/年）
- 管理人员（1-2 天/年）
- 企业所有者（1-2 天/年）
- QA 测试人员（1-2 天/年）
- 安全审核员（1-2 天/年）

相关目标

- 漏洞管理-1
- 体系结构审查-2
- 防御设计-2

相关标准遵从

- PCIv1.1，第 6.5 节
- PCIv1.1，第 12.5.1 节
- PCIv1.1，第 12.6 节
- PCIv1.1，第 12.9.4 节

措施

对员工进行针对特定角色的应用程序安全培训

对那些工作职能强调应用程序安全性的员工进行安全培训。通常，培训可通过两种方式进行：一种是讲师讲授 1-2 天的培训课程；另一种是电脑模块化培训，每人利用等量的学时学习各个模块内容。

对于管理人员和需求规划人员而言，课程内容应着重讲述安全需求规划、漏洞和事件管理、威胁建模，以及误用/滥用案例设计。

针对测试人员和审核人员的培训，应着重培训员工识别和更高效地分析软件有无安全问题。同样，这类课程应着重讲述代码审查、体系结构和设计分析、运行时分析和高效安全测试规划。

扩展针对开发人员和架构师的技术培训，将其他相关主题纳入其中，例如：安全设计模式、特定于工具的培训、威胁模型和软件评估技术。

首次进行这种培训时，建议每年进行一次安全意识培训，并定期进行专门主题的培训。应根据每种角色的员工人数，按需要提供培训（讲师授课或电脑模块化培训）。

聘用安全指导员，增强项目团队

聘用组织内部或外部的安全专家，加入项目团队提供咨询和支持。另外，应在组织内部公告安全专家的入职，确保所有员工知晓。

可在组织内部招募资深员工作为安全指导员，让他们投入一些时间（最多 10%）进行安全指导。这些指导员应通过交流来确保他们了解彼此的专业领域，并将问题传达给相应的人员以提高效率。

在软件生命周期中，任何时候都需要安全指导员。下列情况下，需要安全指导员：初步产品概念研究、完成功能或详细设计规范之前、开发和测试规划期间出现问题时，以及发生操作安全事故时。

随着时间的推移，指导资源的内部网络既可用作整个组织内交流安全信息的联络点，也可作为比完全集中式安全团队更熟悉当前项目团队的本地资源。

培训与指导

EG 3

实施全面的安全培训，并检验员工对基本知识的掌握情况。

措施

创建应用程序安全支持门户网站

在与应用程序安全性相关主题的书面资源的基础上，创建并公布集中式存储库（通常是内部网站）。可以用对组织有意义的任何方式创建操作准则，但必须要建立批准委员会和简明的变更控制流程。

除了最佳实践列表、特定于工具的指南、常见问题解答和其他文章等形式的静态内容，支持门户网站还应具有交互组件，如邮件列表、基于 Web 的论坛或 wiki，以便内部资源交流安全相关的主题且对信息进行分类，以备日后参考。

门户网站的内容应进行归类，以便根据几大常用的要素（如平台、编程语言、与特定第三方库或框架的相关性、生命周期阶段等）轻松搜索。制造软件的项目团队应在产品开发早期就遵守他们应当遵循的特定准则。在产品评估阶段，适用的准则列表和与产品相关的讨论应当用作审核标准。

建立基于角色的考试/认证制度

按角色或按培训班级/模块来创建并管理能力考试，测试员工对安全知识的理解和利用情况。一般说来，应在基于角色的课程的基础上设立考试，目标是正确率至少达到 75%左右。虽然要求员工必须每年参加适当的培训或进修课程，但认证考试至少应每半年进行一次。

根据通过/未通过标准或例外绩效，应将员工分成不同等级，这样其他相关安全活动可要求某个认证级别的个人在活动完成之前结束该活动，例如，未通过认证的开发人员在未得到认证架构师的明确批准前，无法将设计付诸实施。这就为每个项目提供了更为细致的可视性，通过个人责任来跟踪安全决策。整体上看，这为做出有关应用程序安全性的业务决策而奖惩员工奠定了基础。

结果

- 有效地补救现行代码库和旧版代码库的漏洞
- 快速了解和抑制新的攻击和威胁
- 判断员工对安全问题的认知度，并根据通用标准进行评估
- 为提高安全意识，制定公平的激励制度

其他成功标准

- 过去 1 年中，80%以上的员工获得了认证

其他成本

- 认证考试扩充或许可
- 对应用程序安全支持门户网站的现行维护和变更控制
- 执行员工认证的人力资源和间接成本

其他人员

- 开发人员（1 天/年）
- 架构师（1 天/年）
- 管理人员（1 天/年）
- 企业所有者（1 天/年）
- QA 测试人员（1 天/年）
- 安全审核员（1 天/年）

相关目标

- 标准与遵从-2 和 3

相关标准遵从

- PCIv1.1，第 2.2 节
- PCIv1.1，第 6.5 节
- PCIv1.1，第 12.5.1 节
- PCIv1.1，第 12.6 节
- PCIv1.1，第 12.9.4 节

标准和遵从

标准与遵从功能着重于理解和满足外部法规要求，同时还要推动内部安全标准，使其符合外部法规要求，方法是使组织的软件项目业务目标与这些标准保持一致。

改善这些功能的一个具有推动性质的主题是专注于项目级审核，这种审核能搜集组织行为信息，以检查是否达到了预期。例行审核行使简便，随着时间发展其内容会越来越多。通过引入例行审核，组织会不断得到改观。

这一功能的设置形势复杂，还需要使整个组织都能理解内部标准和外部标准遵从的驱动因素，同时还能保持项目团队的低延迟检查点，以确保所有项目都在预期内运行。

	SC 1	SC 2	SC 3
目标	**了解针对组织的相关管理规定和行为要求**	**制定安全基准和行为基准，并了解每个项目的风啊险。**	**要求遵从标准，衡量项目是否符合组织的标准和策略。**
措施	确定并监视外部标准遵从的驱动因素 建立和维护标准遵从检查表	建立安全和标准遵从的内部标准 建立项目审核实践	为项目制定标准遵从关卡 采用审核数据集合的解决方案
结果	提高了第三方审核得到积极结果的信心 根据标准遵从要求的优先顺序，调整内部资源 及时发现影响您的组织的现行管制要求	项目团队具有了对安全和标准遵从预期的意识 组织所有人更清楚地了解其生产线中特定的标准遵从风险 通过随机改进安全性，优化了有效满足标准的方法	整个组织都能了解到因不遵从标准而遇到的风险 在项目级别具体确保标准遵从 准确地跟踪过去的标准遵从历史 利用工具的高效审核流程削减了人工工作量

标准和遵从

SC 1

了解针对组织的相关管理规定和行为要求

措施

确定并监视外部标准遵从的驱动因素

虽然组织有许多标准遵从要求，但该措施专门面向那些直接或间接影响组织构建或使用软件和/或数据方式的法规。利用内部员工专门处理标准遵从事务（如果可行）。

根据组织的核心业务，研究和确定需要遵从或被认定为是业界规范的第三方监管标准。可能需要遵循的标准包括萨班斯-奥克斯利法案 (SOX)、支付卡行业数据安全标准 (PCI-DSS)、健康保险流通与责任法案 (HIPAA) 等。阅读并理解每条第三方标准后，收集与软件和数据相关的具体要求，建立一个汇总表，将每个驱动因素（第三方标准）与其特定的安全要求一一对应。在这一阶段，尝试通过除去可选项或仅推荐的条目来限制要求的数量。

至少每半年进行一次研究，以确保组织针对第三方标准的更改保持最新。根据业界情况和标准遵从的重要性，这一措施可能在成果和涉及人员方面有所变更，但必须明确地执行此措施。

建立和维护标准遵从检查表

根据标准遵从驱动因素的软件和数据相关要求的汇总表，通过为每个需求创建一个相应的响应声明来详细描述这个列表。这个列表有时称为控制声明，每个响应都应当抓住组织经营活动的理念来确保已达到了要求（或指出没能符合要求的原因）。

由于一般的审核实践常常涉及到检查控制声明是否充分，然后参照控制声明本身来衡量组织，因而它们必须要准确地代表实际的组织实践。同样，通过创立简单、轻便的流程元素以做到基本的标准遵从也能满足许多要求，继而使组织能够更好地满足需求。

研究具体的列表，确定主要的差距，以在建立保证计划的未来规划工作中进行参考。与利益相关方就有关标准遵从差距的信息进行交流，确保他们意识到不遵从标准会带来哪些风险。

至少每半年要与利益相关方更新和审查一次控制声明。由于标准遵从的驱动因素很多，因而经常进行更新是很有意义的。

结果

- 提高了第三方审核得到积极结果的信心
- 根据标准遵从要求的优先顺序，调整内部资源
- 及时发现影响您的组织的现行管制要求

成功标准

- 过去 6 个月中，召开了一次以上标准遵从发现会议
- 过去 6 个月中，完成并更新了标准遵从检查表
- 过去 6 个月中，与利益相关方进行了一次以上的标准遵从审查会议

成本

- 初始创建和持续维护标准遵从检查表

员工

- 架构师（1 天/年）
- 管理人员（2 天/年）
- 企业所有者（1-2 天/年）

相关目标

- 战略规划-1

相关标准遵从

-

SC 2

标准和遵从

制定安全基准和行为基准，并了解每个项目的风险。

结果

- 项目团队具有了对安全和标准遵从预期的意识
- 组织所有人更清楚地了解其生产线中特定的标准遵从风险
- 通过随机改进安全性，优化了有效满足标准的方法

其他成功标准

- 过去 6 个月中，75%以上的员工简要介绍了内部标准
- 80%以上的利益相关方关注内部标准的遵从状态

其他成本

- 扩充或许可内部标准
- 每个项目遵从内部标准和审核的开销

其他人员

- 架构师（1 天/年）
- 管理人员（1 天/年）
- 安全审核员（2 天/项目/年）

相关目标

- 培训与指导-1 和 3
- 战略规划-2
- 安全要求-1 和 3
- 防御设计-3
- 代码审查-3
- 体系结构审查-3
- 增强基础架构-3

相关标准遵从

措施

建立安全和标准遵从的内部标准

从当前标准遵从检查表开始，审查监管标准并注意任何可选或建议的安全需求。同样，组织应当进行一些研究，以揭示相关标准遵从要求的未来更改。

根据已知业务安全驱动因素添加一些要求来扩充该列表。通常，只需参考已为开发人员提供的现有指南并搜集一系列最佳实践。

将普通/相似的要求进行分组，将每个组重写为更一般化/简单的、可满足所有标准遵从驱动因素并能提供一些额外安全值的声明。每一次分组都执行该流程，目的是建立一组可直接映射回标准遵从驱动因素和最佳实践的内部标准。

统一标准不能包含项目团队难以遵循或遵循成本太高的要求，这一点很重要。大约 80%的项目应能够在中断最少的情况下遵从标准。这需要制定一个良好的交流计划，以宣传新的内部标准，并在需要时帮助项目团队遵从标准。

建立项目审核实践

为项目团队创建一个简单的审核流程，根据内部标准来请求和接受审核。审核通常由安全审核员执行，不过，只要熟知安全知识的员工熟悉内部标准，也可由他们执行这一任务。

根据所有已知的业务风险指标，对可按优先级排列为审核队列的项目进行分类，这样，高风险软件的评估将更早地执行或更频繁地执行。此外，低风险项目可以使内部审核要求放宽，以使审核实践更为经济高效。

从整体上看，每个活动的项目都至少要每半年进行一次审核。一般说来，如果保留了足够的有关应用程序的审核信息，初次审核之后的审核会更简单。

向组织所有人和其他利益相关方介绍该服务，以便他们能够为他们的项目提出审核申请。根据内部标准，每个需求通过/未通过的详细结果都应交给项目利益相关方进行评估。在可行的情况下，审核结果也应包含对影响的解释和修复建议。

标准和遵从

SC 3

要求遵从标准，衡量项目是否符合组织的标准和策略。

措施

为项目制定标准遵从关卡

组织制定内部安全标准后，要实施的下一步措施就是在项目生命周期上设置一些特殊的点，在项目通过内部标准审核、被认定遵从标准之前，项目不能通过。

通常，标准遵从关卡位于软件发布的点上，这样在通过标准遵从检查之前，不允许发布发行版。留出足够的审核和修复时间很重要。因此，通常应较早开始进行审核，例如在将发行版交给 QA 的时候。

尽管标准遵从关卡是固定的标准，但旧项目或其他特定的项目可能无法遵从标准，因而必须建立例外批准流程。获得例外批准的项目不应超过所有项目的 20%。

采用审核数据集合的解决方案

定期对项目团队进行审核的组织会逐渐积累大量审核数据。应使用自动化工具帮助自动收集信息、管理存储和检索的核对，并限制个人访问敏感审核数据。

对于内部标准的许多具体要求，现有的工具（如代码分析器、应用程序渗透测试工具、监控软件等）可以经过定制，用来自动检查内部标准的遵从情况。自动检查遵从情况的目的在于提高审核的效率，同时在正式审核前使更多的员工能够自我检查遵从情况。此外，自动检查不容易发生错误，发现问题的延迟时间短。

信息存储功能应当支持按项目集中访问当前和历史审核数据。自动解决方案还能提供详细的访问控制功能，仅限获得批准的、具有合法业务目的之个人访问审核数据。

整个团队的成员都应了解访问标准遵从数据和请求访问权限的所有说明和步骤。安全审核员初次引导项目团队时需要的时间会更长一些。

结果

- 整个组织都能了解到因不遵从标准而遇到的风险
- 在项目级别具体确保标准遵从
- 准确地跟踪过去的标准遵从历史
- 利用工具的高效审核流程削减了人工工作量

其他成功标准

- 审核发现 80%以上的项目遵从内部标准
- 每次自动审核使用的时间不到人工审核的 50%

其他成本

- 扩充或许可内部标准审核自动化工具
- 审核关卡和例外流程的当前维护费用

其他人员

- 开发人员（1 天/年）
- 架构师（1 天/年）
- 管理人员（1 天/年）

相关目标

- 培训与指导-3
- 代码审查-2
- 安全测试-2

相关标准遵从

战略规划

战略规划功能着重于在组织内部为软件安全保证计划建立框架。这是定义与组织真实商业风验相切合的安全目标的最基本步骤。

组织从简便的风险配置文件入手，逐渐形成更高级的应用程序和数据资产风险分类方案。通过更深入地了解相关风险度量，组织便可以调整项目级安全目标，并制定出更为细化的方案，使安全计划更为高效。

在该功能更为高级的层面上，组织可利用许多内部和外部数据源来收集关于安全计划的标准和定性反馈。这使得组织能够在整个计划层面精确地调整成本支出与实现的收益的关系。

	SP 1	SP 2	SP 3
目标	**针对组织内的软件安全性，制定统一的战略性方案。**	**了解数据和软件资产的相对价值，并选择风险承受水平。**	**使安全开支与相关业务指标和资产价值相符。**
措施	• 估计总体业务风险概况 • 建立和维护保证计划	• 根据业务风险对数据和应用程序进行分类 • 建立每个分类的安全目标	• 定期进行业界范围的成本比较 • 收集历史安全费用的标准
结果	📖具体列出了软件导致的最关键的业务级风险 📖定制了能够满足组织安全需求、同时开销最小的方案 📖整个组织都了解保证方案的逐步进展方式	📖根据对业务的核心价值，为每个项目定制保证方案 📖整个组织都了解数据和应用程序资产的安全相关性 📖使利益相关方更好地了解并接受风险	📖有助于制定每个案例安全支出决策的信息 📖估算过去由于安全问题而造成的损失 📖考虑每个项目的安全支出与潜在损失 📖整个行业对于安全性的审慎调查

战略规划

SP 1

针对组织内的软件安全性，制定统一的战略性方案。

措施

估计总体商业风险概况

采访组织所有人和利益相关方，列出组织各种应用程序和数据资产的最坏情况方案。根据组织构建、使用或出售软件的方式，最坏情况方案的列表也会有很大的不同，但一般的问题包括数据盗窃或损坏、服务中断、金钱损失、逆向工程和帐户泄漏等。

广泛地获得最坏情况方案观点后，根据所收集到的信息和相关核心业务的知识，核对并选择最重要的最坏情况方案观点。虽然可以选择任何数目的观点，但选择目标的范围应在 3 到 7 个，以便高效利用时间并使工作有重点。

向组织阐释所选择每一个观点，并记录起作用的最坏情况方案、潜在成因及潜在的牵制要素的详情。

最终的组织风险概况应经过组织所有人和其他利益相关方的审查，确保他们了解这些内容。

建立和维护保证方案

了解组织的主要业务风险，评估组织针对十二项功能中每项功能目前的绩效。如果组织通过了所有累积的成功标准，可以根据相应的目标为每项功能打 1、2 或 3 分。如果没有达到成功标准，则对该功能打 0 分。

完全了解了目前状态后，下一个目标就是确定下一次需要改善的功能了。根据业务风险概况、其他业务驱动因素、标准遵从需求、预算容限等选择功能。之后，重复操作的目的就是达到每个功能的下一目标。

在保证计划上再次取得进展约需要 3-6 个月，但应当至少每 3 个月召开一次保证战略会议，参照成功标准和要求更改计划的其他业务驱动因素，对工作进度和绩效进行审核。

结果

- 具体列出了软件导致的最关键的业务级风险
- 定制了能够满足组织安全需求、同时开销最小的方案
- 整个组织都了解保证计划的进展方式

成功标准

- 过去 6 个月中，80%以上的利益相关方都简要介绍了业务风险概况
- 过去 3 个月中，80%以上的员工都简要介绍了解了保证方案
- 过去 3 个月中，召开了一次以上的保证计划战略会议

成本

- 扩充和维护组织风险概况
- 每季度评估一次保证计划

人员

- 开发人员（1 天/年）
- 架构师（4 天/年）
- 管理人员（4 天/年）
- 企业所有者（4 天/年）
- QA 测试人员（1 天/年）
- 安全审核员（4 天/年）

相关目标

- 标准与遵从-1
- 威胁建模-1
- 安全要求-2

相关标准遵从

SP 2

战略规划

理解数据和软件资产的相对价值并选择风险容限。

结果

- 根据对业务的核心价值为每个项目定制保证计划
- 整个组织都了解数据和应用程序资产的安全相关性
- 使利益相关方能够更好地了解并接受风险

其他成功标准

- 过去 6 个月中，90%以上的应用程序和数据资产进行了风险分类评估
- 过去 6 个月中，80%以上的员工简要介绍了相关应用程序和数据的风险评级
- 过去 3 个月中，80%以上的员工简要介绍了相关保证计划方案

其他成本

- 扩充或许可应用程序和数据风险分类方案
- 更为精细的规划的编程开销

其他人员

- 架构师（2 天/年）
- 管理人员（2 天/年）
- 企业所有者（2 天/年）
- 安全审核员（2 天/年）

相关目标

- 标准与遵从-2
- 威胁建模-2
- 体系结构审查-2

相关标准遵从

-

措施

根据业务风险对数据和应用程序进行分类

建立一个简单的分类系统来表示应用程序的风险等级。最简单的形式是分为高/中/低类别。也可以使用更为复杂的分类方法，但所分的类别不应超过七个，它们从高到低表示对业务风险的影响。

研究组织的业务风险概况，制定能将每个项目映射到一个风险分类的项目评估标准。应当为数据资产制定一个相似但又独立的分类模式，必须根据对业务风险的潜在影响对每一项进行权衡和分类。

对收集的每个应用程序的信息进行评估，并根据总评估标准和目前使用的数据资产风险分类，为每个应用程序的分配风险类别。这项工作可由安全小组集中进行，也可由单独的项目组通过定制的调查问卷来收集必要的信息。

应当制定一个针对应用程序和数据资产风险分类的流程，以便为新资产分配风险类别，并至少每半年更新一次现有数据。

建立每个分类的安全目标

实施了组织应用程序系列的分类方案后，可以更细致地选择直接安全目标和保证方案。

应通过指定每个类别特定功能的重点将保证计划的路线图修改为能够说明每个应用程序风险类别。每次执行保证计划时，这通常会按照优先级将更高级别的目标排在最高的风险应用程序等级，并依次将不太紧要目标的排在较低/其他类别。

这一过程可建立组织的风险容限，因为必须要为每个风险类别中的应用程序期待哪些具体的目标做出主动决策。通过选择将较低风险的应用程序保持在较低的安全功能性能级别，资源会被保存，以换取接受权重风险。但是没有必要为每个风险类别构建单独的路线图，因为这样会使管理保证计划本身效率低下。

战略规划

SP 3

使安全开支符合相关业务指标和资产价值

措施

定期进行业界范围的成本比较

研究并从从行业间交流论坛、业务分析和咨询公司或其他来源收集相关安全成本信息。特别需要确定一些关键因素。

首先，使用收集的信息确定行业中同类组织在安全方面投入的平均工作量。这可以通过自上往下估算预算、收入等的总百分比完成，或通过确定被认为是组织正常的相关安全活动自下往上完成。从整体上说，某些行业的情况很难估计，因此需要从尽可能多的可访问相关信息源收集信息。

研究安全成本的下一个目标是确定您的组织正使用的第三方安全产品和服务是否可以节约成本。考虑决定更换供应商时，要将隐藏成本（如重新培训员工费用或其他计划开销）考虑在内。

总的说来，这些成本比较工作应当在后续保证计划战略会议之前至少每年进行一次。比较信息应呈送给利益相关方，以更好地使保证计划与业务保持一致。

收集历史安全费用的标准

收集特定于项目的、关于以往安全事件的成本的信息。例如，为清除违反案例而耗费的时间和金钱、因系统停机而导致的金钱损失、管理组织的罚款和收费、特定于项目的工具或服务的一次性安全支出等。

使用应用程序风险分类和为各类别分别规定的保证计划，每个应用程序的基本安全成本最初可以从与相应的风险类别相关的成本计算。

根据风险类别将特定于应用程序的成本信息与一般成本模型相结合，然后评估项目是否有离群值，即总结风险评级不相称的方面。这些不相称的方面表示风险评估/分类有错误，或需要调整组织的保证计划，以更有效地解决安全成本的根源问题。

应每季在保证计划战略会议上对每个项目的安全开支进行一次跟踪，利益相关方应至少每年对这些信息进行一次评估和审查。应当讨论离群值和其他无法预见的成本对保证计划路线图的潜在影响。

结果

- 有助于制定每个案例安全支出决策的信息
- 估算过去由于安全问题而造成的损失
- 考虑每个项目安全支出与潜在损失
- 整个行业对于安全性的审慎调查

其他成功标准

- 过去 3 个月中，80%以上的项目报告了安全成本
- 过去 1 年中进行了一次以上业界范围的成本比较
- 过去一年中进行了一次以上历史安全支出评估

其他成本

- 扩充或许可安全计划行业情报
- 因成本核算、跟踪和评估而导致的编程开销

其他人员

- 架构师（1 天/年）
- 管理人员（1 天/年）
- 企业所有者（1 天/年）
- 安全审核员（1 天/年）

相关目标

- 漏洞管理-1

相关标准遵从

-

威胁建模

威胁建模功能的中心问题是根据正在开发的软件的业务功能确定和了解项目级风险。详细了解了对每个项目的威胁和可能的攻击后，通过更明智地决定安全计划的优先级，整个组织可以更高效的运营。此外，对风险接受的决策会更广为人知，因而会更好地与业务相协调。

通过先从简单的攻击树开始，并构建更为详细的威胁检测和权重方法，组织会逐渐得到改善。最终，组织繁杂的组织会维护这些信息，使其与补偿因素紧密结合，从而通过来自外部实体风险的考验。这扩大了对潜在下游安全问题影响的理解，同时能紧密观察组织目前在应对已知威胁方面的绩效。

	TM 1	TM 2	TM 3
目标	**确定并了解对组织和单个项目的高级别威胁。**	**提高威胁评估的准确性，更深入了解每个项目的细节。**	**将补偿控制与对内部和第三方软件的每个威胁具体联系起来。**
措施	• 建立并维护每个应用程序的攻击树 • 根据软件体系结构开发攻击者概况	• 用特定于应用程序的数据详细描述攻击树 • 采纳一种威胁度量的权重系统	• 明确估计第三方组件的风险 • 用补偿控制详细描述攻击树
结果	📖深入理解可能导致消极结果的因素 📖越来越意识到项目团队中的威胁 📖详细列出了您的组织的威胁	📖更详细地了解单个项目可能面临的威胁 📖建立了在项目组内更好地进行权衡决策的框架 📖能够根据风险权重对项目组内的开发工作进行优先级排序	📖更深层次地考虑每个软件项目的完整威胁概况 📖更详细地将保证功能与每个软件项目已确定的威胁对应 📖创建了根据每个软件项目的业务功能记录审慎调查的对象

威胁模型

TM 1

确定并了解对组织和单个项目的高级别威胁。

措施

建立并维护每个应用程序的攻击树

仅根据每个软件项目的业务目标和组织风险概况（如果有），确定每个项目团队正在开发的软件的可能的最坏情况方案。用一句话来说明每个最坏情况方案，并将其标记为攻击者的高级别目标。

针对已确定的每个攻击者目标，确定实现每个目标必须拥有的前提。这一信息应位于每个目标的分支下，其中每个分支都是下面所含的语句的逻辑 AND 或逻辑 OR。AND 分支表示每个直接连接的子节点必须是真才能实现该父节点。OR 分支表示任何一个直接连接的子节点必须是真就能实现其父节点。

检查每个当前功能需求和历史功能需求（如果可用），并扩增攻击树以显示与每个需求相关联的安全故障。集思广益，反复将每个节点分为分支，表示攻击者为达到某个目标使用的所有可能的方法。初步创建完成后，应用程序的攻击树应在软件做出了重大修改时进行更新，可删除任何节点，也可添加新节点。这一评估应由高级开发人员和架构师以及一名或多名安全审核员来完成。

根据软件体系结构开发攻击者概况

首先进行一次评估，基于软件项目确定组织可能面临的所有攻击。对于这次评估，可以将威胁限制在恶意内容代理，而忽略其他风险，如已知漏洞、可能的缺陷等。

从普遍认定的外部代理及它们相应的攻击动机入手。向该列表添加可能导致损害的内部角色及它们进行内部攻击的动机。根据考虑中的软件项目的体系结构，对每种体系结构进行一次这种分析会比对每个项目单独进行这种分析要高效得多，因为相同体系结构的和业务目的的应用程序通常很容易受到相同的攻击。

这一评估应由组织所有人和其他利益相关方以及一名或多名审核员进行，以获得对威胁的更多了解。最后，目标是制定一个包括威胁代理及其相应的攻击动机的简明列表。

结果

- 深入理解可能导致消极结果的因素
- 越来越意识到项目团队中的威胁
- 详细列出了您的组织的威胁

成功标准

- 过去 12 个月中，50%以上的项目利益相关方简要介绍了相关项目的攻击树
- 75%以上的项目利益相关方简要介绍了相关体系结构的攻击者概况

成本

- 扩充和维护攻击树的项目对象

人员

- 企业所有者（1 天/年）
- 开发人员（1 天/年）
- 架构师（1 天/年）
- 安全审核员（2 天/年）
- 管理人员（1 天/年）

相关目标

- 战略规划-1
- 安全要求-2

相关标准遵从

TM 2

威胁建模

提高威胁评估的准确性，更深入了解每个项目的细节。

结果

- 更详细地了解单个项目可能面临的威胁
- 建立了在项目组内更好地进行权衡决策的框架
- 能够根据风险权重对项目组内的开发工作进行优先级排序

其他成功标准

- 75%以上的项目团队确定了威胁并对威胁进行了评级
- 过去 6 个月中，75%以上的项目利益相关方简要介绍了相关项目的攻击树
- 过去 12 个月中，项目攻击树确定了 50%以上的安全事件

其他成本

- 维护攻击树和攻击者概况的项目开销

其他人员

- 安全审核员（1 天/年）
- 企业所有者（1 天/年）
- 管理人员（1 天/年）

相关目标

- 战略规划-2
- 防御设计-2

相关标准遵从

措施

用特定于应用程序的数据详细描述攻击树

从为每个软件项目建立的攻击树，详细描述分析情况，将软件设计和实现的技术详情包含在内。这可以通过考虑关于软件的几大因素来做到。由于大型应用程序的攻击树会很冗长，可在需要的时候将它们拆分，但需要确保每一段攻击树的路径都要包含总攻击树的根目标节点和连接节点的副本。

先从安全相关的假定（一般在设计阶段考虑）入手，然后通过说明每个假定失败的节点来扩展攻击树。根据项目的安全需求，详细描述攻击树上说明每个安全需求失败的必要前提的节点。此外，说明与软件项目相关的所有已知安全事件，以确保攻击树上它们已存在。

最后，确定用户角色及其对应进一步详细描述攻击树中捕获的内部攻击方案的能力。如果可以使用，则使用特定于应用程序的权限标准来促进他们考虑故障案例，如责任分离问题。

为威胁度量采纳权重系统

根据已建立的攻击者概况，确定一个分级系统，使得能够对威胁进行比较。首先，这可以是一个简单的按业务风险划分的“高-中-低”分级系统，但是任何度量方法都可以使用，只要所分的级别不超过 5。

确定了分级系统后，可建立能为每个威胁指定一个级别的评估标准。为了正确地做到这一点，除了攻击动机外，还需要考虑每个威胁的其他因素。重要的因素包括：资金和人力资源、固有的访问权限、技术能力、攻击树的相关目标及攻击成功的可能性等。

为每个威胁指定了级别后，可使用该分级信息将开发生命周期内的风险排除工作进行优先级排序。建立该系统后，应在设计新功能或进行重构工作期间对其进行更新。

威胁模型

TM 3

将补偿控制与对内部和第三方软件的每个威胁具体联系起来。

措施

明确估计第三方组件的风险

对软件代码库进行一次评估，确定来自外部的组件。一般说来，外部组件包括开源项目、购买的 COTS 软件及软件使用的联机服务。

对于每一个已确定的组件，根据第三方组件可能造成的危害详细描述软件项目的攻击者概况。在新确定的攻击者概况基础上更新软件攻击树，根据新攻击者目标或能力将所有可能的风险纳入其中。

除了威胁方案外，还应考虑第三方软件的漏洞或设计缺陷影响您的代码和设计的方式。使用漏洞的潜在风险并通过对更新的攻击者概况的了解来对攻击树做相应的阐述。

最初为项目详细描述了攻击树后，设计阶段或每个开发周期都要对其进行更新和审查。这一工作应由安全审核员和相关的技术和组织利益相关方来完成。

用补偿控制详细描述攻击树

进行一次评估，正式确定能够直接防止攻击树上的节点代表的危及安全的前提的因素。这些牵制因素是补偿控制，能够有效地抵御软件的直接风险。这些因素可以是软件本身的技术功能，但也可以是开发生命周期的过程元素以及基础设施功能等。

由于攻击树上表示了逻辑关系，每个分支都代表 AND 或 OR。因此，只要牵制 AND 分支的一个前题，父节点和所有连接的叶节点都可标记为被牵制。但是，要使父节点标记为被牵制，必须阻止 OR 节点上的所有子节点。

通过审查攻击树，确定可将标记为被牵制节点的数目最大化的补偿控制。如果从叶节点到顶级目标有多个可行路径，则需要确定反馈到组织战略中的潜在补偿控制。

最初为项目详细描述了攻击树后，设计阶段或每个开发周期都要对其进行更新和审查。这一工作应由安全审核员和相关的技术和组织利益相关方来完成。

结果

- 更深层次地考虑每个软件项目的完整威胁概况
- 更详细地将保证功能与每个软件项目已确定的威胁对应
- 创建了根据每个软件项目的业务功能记录审慎调查的对象

其他成功标准

- 在每个实现周期之前，80%以上的项目团队都更新了攻击树
- 在发布软件前，80%以上的项目团队更新了第三方组件的清单

其他成本

- 维护详细的攻击树和扩充的攻击者概况所产生的项目开销
- 发现所有第三方 dependency

其他人员

- 企业所有者（1 天/年）
- 开发人员（1 天/年）
- 架构师（1 天/年）
- 安全审核员（2 天/年）
- 管理人员（1 天/年）

相关目标

- 安全要求-2 和 3

相关标准遵从

安全要求

系统需求功能着重于主动指定软件的期望安全行为。通过在项目级别添加分析工作，安全需求最初是根据软件的高级业务目标来收集的。随着组织的发展，将使用更为复杂的技术（如滥用案例建模）来发现最初对开发不明显的新安全需求。

在复杂的形式中，本功能还包括将组织的安全需求纳入组织与供应商的关系、然后对项目进行审核，以确保各项工作都能达到安全需求规范的预期。

	SR 1	SR 2	SR 3
目标	**在软件需求阶段明确地将安全考虑在内。**	**制定特定于应用程序的风险和滥用方案的安全需求。**	**对所有软件项目和第三方 dependency 强制要求全面的安全需求过程。**
措施	从业务功能推出安全需求 使用指导原则、标准和标准遵从资源	为每个项目建立和维护滥用方案模型 根据已知风险指定安全需求	将安全要求写入供应商协议 扩展审核计划以满足安全需求
结果	开发工作与业务风险保持高度一致 将专门捕获业界安全最佳实践作为明确要求 股东都意识到要采取措施降低软件风险	详细了解了业务逻辑的攻击案例 根据可能的攻击对安全功能的开发工作进行优先级排序 功能和安全工作间的权衡决策更有根据 各利益相关方能更好地避免那些本身有安全缺陷的功能要求	正式对外部代码设定了基准安全预期 每个项目团队都能使用集中的安全措施信息 能够根据应用程序风险和期望的安全需求安排项目的资源

安全要求

SR 1

在软件需求阶段明确地将安全考虑在内。

措施

从业务功能推出安全需求

对为每个软件项目指定业务逻辑和整体行为的功能需求进行一次审查。为项目收集需求后，进行一次评估，以导出相关安全需求。即使软件由第三方构建，这些需求一旦确定，就应当纳入交付给供应商的功能需求中。

对于每个功能需求，安全审核员都应当引导利益相关方明确指出关于安全的任何预期。一般说来，对每个需求需澄清的问题包括：对数据安全的期望、访问控制、交易完整性、业务功能的关键程度、职责划分、正常运行时间等。

确保所有的安全需求总体上都遵循与编写良好需求相同的原则很重要。具体说来就是，安全需求应具体、可度量且合理。

对活动项目的所有新需求都执行该步骤。对于现有的功能，建议将这一步骤作为差距分析，以促进未来的安全重构。

使用指导原则、标准和标准遵从资源

确定项目团队应将其视为需求的业界最佳实践。这些最佳实践可从公开可用的指导原则、内部/外部标准或已订立的标准遵从要求选择。

不要尝试将太多的最佳实践要求引入每个开发过程，这一点很重要，因为需要权衡设计和实施的时间。建议在连续的开发周期中慢慢地添加最佳实践，随时间推动软件的总体保证概况。

对于现有的系统，重构安全最佳实践是一项复杂的工作。在可能的情况下，添加新功能时适当增加安全需求。至少要进行分析，确定适用的最佳实践已执行，以帮助推动未来的规划工作。

该审查应由安全审核员和组织利益相关方进行。高级开发人员、架构师和其他技术利益相关方也应参与进来，为决策过程带来特定于设计和实现的知识。

结果

- 开发工作与业务风险保持高度一致
- 将专门捕获业界安全最佳实践作为明确要求
- 利益相关方都意识到要采取措施降低软件风险

成功标准

- 50%以上的项目团队明确确定了安全需求

成本

- 向每个开发周期添加安全需求需要支付的项目开销

人员

- 安全审核员（2 天/年）
- 企业所有者（1 天/年）
- 管理人员（1 天/年）
- 架构师（1 天/年）

相关目标

- 培训与指导-1
- 标准与遵从-2
- 体系结构审查-1
- 代码审查-1
- 安全测试-1

相关标准遵从

-

SR 2

安全要求

制定特定于应用程序的风险和滥用方案的安全需求。

结果

- 详细了解了业务逻辑的攻击案例
- 根据可能的攻击对安全功能的开发工作进行优先级排序
- 功能和安全工作之间的权衡决策更有根据
- 各利益相关方能更好地避免那些本身有安全缺陷的功能要求

其他成功标准

- 过去 6 个月中，75%以上的项目更新了滥用案例模型

其他成本

- 扩充和维护滥用案例模型的项目开销

其他人员

- 安全审核员（2 天/年）
- 管理人员（1 天/年）
- 架构师（2 天/年）
- 企业所有者（1 天/年）

相关目标

- 威胁建模-1 和 3
- 战略规划-1

相关标准遵从

-

措施

为每个项目建立和维护滥用案例模型

进一步考虑软件项目的功能需求，进行一次更为正式的分析，以确定潜在的功能误用或滥用情况。一般说来，这一过程首先要确定一般使用方案，如用例图表（如可用）。

对于每个使用方案，可以先声明一般使用情况，然后再集体讨论该声明可能被整体或部分否定的方式，从而制定出一系列滥用案例。最简单的开始方式是将“否”或“不”这样的词语尽可能多地插入到使用声明中，特别是在名词和动词周围。每个使用方案都应当生成几个可能的滥用案例声明。

进一步阐述滥用案例声明，根据软件的业务功能将所有特定于应用程序的担忧包含在内。最终目标是为完整的滥用声明系列构成一个软件不接受的使用模式模型。

初步创建完成之后，在设计阶段应为活动项目更新滥用案例模型。对于现有项目，应对新需求进行分析以防止可能的滥用。现有项目应视情况为已确定的功能构建滥用案例。

根据已知风险指定安全需求

明确审查表明特定于组织或项目的安全风险的现有对象，以便更好地了解软件的整体风险概况。如果可用，可利用像高级别业务风险概况、个人应用程序攻击树、体系结构中的查找结果、代码审查和安全测试等这样的资源。

除了审查现有对象之外，可使用应用程序的滥用案例模型，来促进确定具体直接或间接牵制滥用案例的安全需求。

这一过程应按照需要，由组织所有人和安全审核员来进行。最终，引起新安全需求的风险概念应成为规划阶段的一个必要的步骤。项目团队可通过该步骤明确评估新发现的风险。

安全要求

SR 3

要求对所有软件项目和 dependency 执行全面的安全需求过程。

措施

将安全要求写入供应商协议

除了前面的分析已确定的安全需求种类之外，还能从第三方协议获取安全优势。一般说来，需求和可能的高级设计会在组织内部开发，而低级设计和实施工作则会留给供应商完成。

由于每个外部开发组件的具体劳动分工各有不同，需要确定具体的安全工作和技术评估标准，并将它们写入到供应商合同中。一般情况下，这是“验证和评估规范”的一组工作，涵盖了体系结构审查、代码审查和安全测试功能。

外部开发组件的使用方式

修改协议的措辞应与供应商针对不同案例进行讨论，因为增加额外的需求通常也意味着增加成本。应按照正在考虑的组件或系统的用途来权衡每个潜在安全工作的成本和工作的收益。

扩展审核计划以满足安全需求

将检查安全需求的完整性纳入例行项目审核中。由于在没有项目专业知识的情况下很难做到这一点，因此审核应着重于检查项目对象（如需求或设计文档），以查看执行适当类型分析的证据。

特别是应当在组织推动因素和期望的滥用案例基础上给每个功能要求加上安全要求注解。项目整体需求应包含从指导原则和标准中的最佳实践生成的需求列表。此外，应当清楚地列出未实现的安全需求，以及未来版本中实现这些要求的预估时间线。

这一审核应在每次重复开发的情况下执行，理想情况是在需求流程结束时进行，但在发行软件版本之前必须进行审核。

结果

- 正式对外部代码设定了基准安全预期
- 每个项目团队都能使用集中的安全措施信息
- 能够根据应用程序风险和期望的安全需求安排项目的资源

其他成功标准

- 过去 6 个月中，80%以上的项目通过了安全需求审核
- 过去 12 个月中，对 80%以上的供应商协议的合同安全需求进行了分析

其他成本

- 因安全需求增加而增加外包开发的成本
- 因安全需求的发行关卡造成的当前项目开销

其他人员

- 安全审核员（2 天/年）
- 管理人员（2 天/年）
- 企业所有者（1 天/年）

相关目标

- 威胁建模-3
- 标准与遵从-2

相关标准遵从

防御设计

防御设计功能注重组织在默认情况下能够采取积极措施来构建安全软件。通过构建一种出现下游漏洞的机率较小的软件，可大大降低软件的整体安全风险。

组织最开始会采用基于最佳实践的高级设计指南，然后发展到采用一贯使用的安全功能设计模式。以此为基础，各组织还鼓励项目团队更好地理解业务逻辑，并创建能够从一开始就确保软件设计安全的正规方法。

随着组织的发展，此功能提供的复杂内容将包括促使组织建立参照平台，以涵盖其构建的普通类型软件。开发人员将这些平台作为框架来构建漏洞风险较低的自定义软件。

	DD 1	DD 2	DD 3
目标	**向项目团队提供超前预防的设计指导，并加强项目周边安全。**	**在设计过程中，开发可全面增强安全性的软件。**	**评估项目团队的超前预防措施，使防御性设计更高效。**
措施	维护推荐软件框架的列表 将安全原则显式应用于外围界面	构建资源访问权限矩阵 根据体系结构确定安全设计模式	建立正式的参照平台 验证框架、模式和平台的使用
结果	专门防止意外依赖性和选择一次性实现 利益相关方意识到因选择库和框架而使项目风险加大 在开发过程中建立了协议，旨在积极地将安全机制应用于设计	详细地将资产映射到用户角色，以鼓励在设计中更好地进行划分 提供了具有安全保护和功能的可重用的设计组件 通过使用成熟的安全设计技术，软件项目的可信度得到提高	可提供内置安全保护的自定义应用程序开发平台 整个组织都期望在开发过程中针对安全性做出更积极的努力 利益相关方更有能力根据组织对安全设计的需求做出权衡决策

防御设计

DD 1

向项目团队提供积极的设计指导，并加强项目周边安全。

措施

维护推荐软件框架的列表

确定组织中软件项目所常用的第三方软件库以及正在使用的框架。通常情况下，这无需艰难地搜索依赖性，而是要着重于捕获使用频率最高的高级组件。

根据第三方组件所提供的核心功能，在组件列表中将其划分成不同的功能类别。另外，注意各项目组每个组件的使用情况，以衡量对第三方代码的依赖程度。将这个经权衡后的列表作为指南，创建一个推荐组件列表并通告给开发机构。

对于推荐列表所包含的内容，有几个决定因素。虽然可以不进行专门调查便可确定列表，但是我们建议检查每个组件的事件历史、响应漏洞的跟踪记录、对于组织而言其功能的适合性、使用第三方组件带来的额外复杂性等。

此列表应由高级开发人员和架构师创建，但也需要管理人员和安全审核员的介入。创建推荐组件列表后，应将与功能类别相匹配的列表通告给开发机构。最后，目标是为项目团队提供为人熟知的默认资源。

将安全原则显式应用于外围界面

在设计期间，项目团队中的技术人员应使用简短的指导性安全原则列表，作为检查系统中边缘界面的检查表。通常情况下，安全原则包含深度防御、保护最脆弱的链接、使用安全默认选项、安全功能设计的简明性、保护失败操作、平衡安全性和可用性、用最低权限运行、避免通过隐匿来确保安全性等。

对于任何一个已知的周边界面，设计团队都应在整体系统环境中考虑每一条原则，并确定可添加到该界面以增强安全性的特性。通常情况下，应该对其进行限制，使其只需在功能要求常规实现成本之外做少量额外工作，应指出量更大的工作，并计划用于未来版本。

在接受安全意识培训后，每个项目团队都应执行此流程，这有助于使更多通晓安全问题的员工帮助制定设计决策。

结果

- 专门防止意外依赖性和选择一次性实现
- 利益相关方意识到因选择库和框架而使项目风险加大
- 在开发过程中建立了协议，旨在积极地将安全机制应用于设计

成功标准

- 过去 1 年中，80%以上的开发人员了解了软件框架建议
- 50%以上的安全原则的项目自报告应用程序将要进行设计

成本

- 扩充、维护和了解软件框架建议
- 正在进行项目的分析开销和安全原则的应用开销

人员

- 架构师（2-4 天/年）
- 开发人员（2-4 天/年）
- 安全审核员（2-4 天/年）
- 管理人员（2 天/年）

相关目标

- 培训与指导-1

相关标准遵从

-

DD 2

防御设计

在设计过程中，开发可全面增强安全性的软件。

结果

- 详细地将资产映射到用户角色，以鼓励在设计中更好地进行划分
- 提供了具有安全保护和功能的可重用的设计构件块
- 通过使用成熟的安全设计技术，软件项目的可信度得到提高

其他成功标准

- 过去 6 个月中，80%以上的项目更新了权限矩阵
- 过去六个月中，80%以上的项目组了解了适用的安全模式

其他成本

- 扩充或许可适用的安全模式
- 正在进行的项目维护权限矩阵的开销

其他人员

- 架构师（2-4 天/年）
- 开发人员（1-2 天/年）
- 管理人员（1-2 天/年）
- 企业所有者（1 天/年）
- 安全审核员（1-2 天/年）

相关目标

- 培训与指导-2

相关标准遵从

-

措施

构建资源访问权限矩阵

根据该应用程序的业务目标，确定用户和操作员角色。另外，通过收集所有相关的数据资产和以任何形式的访问控制限定的应用程序特定的特性，建立资源列表。

在一个轴表示角色、另一个轴表示资源的简单矩阵中，考虑每个角色和资源之间的关系，并记录根据利益相关方在访问控制方面每个交叉点中系统的正确行为。

对于数据资源，注意创建、读访问、更新和删除的访问权限很重要。对于特性资源，访问权限的分级可能是应用程序特定的，但至少要注意某角色是否应被允许访问特性。

此权限矩阵可作为一个工件，用来记录整个系统业务逻辑的正确访问控制权限。同样，它应由项目团队创建，并需要组织利益相关方参与。最初创建矩阵后，应在每次发布前应由组织利益相关方对其进行更新，但是通常是在设计阶段初期进行。

根据体系结构确定安全设计模式

组织中的软件项目都应该按照常用架构类型进行分类。常见类别包括客户端-服务器应用程序、嵌入式系统、桌面应用程序、面向 Web 的应用程序、Web 服务平台、事务中间件系统、大型机应用程序等。根据组织的专注领域，可能需要根据语言、处理器体系结构甚或是部署时期来划分出更详细的类别。

对于普通的软件体系结构类型，可推出一系列代表执行安全功能可靠方法的常用设计模式，并应用于组织软件项目的单独设计。这些安全设计模式代表可研究或购买的普通设计元素的一般性定义。如果对这些模式进行自定义以更特定于组织，则它们会更高效。示例模式包括单点登录子系统、跨层委派模型、增强的界面设计、职能分离授权模型、集中式日志记录模式等。

确定适用和合适模式的流程应由架构师、高级开发人员以及其他技术利益相关方在设计阶段执行。

防御设计

DD 3

评估项目团队的超前预防措施，使防御性设计更高效。

措施

建立正式的参照平台

使用特定于每种类型体系结构的安全模式之后，项目团队应选择执行这些功能片段的代码集，并用作共享基本代码的基础。此共享基本代码最初可作为每个项目都要使用的一组常用推荐库，时间一长，可发展成为一个或更多软件框架，代表项目团队在其上构建软件的参照平台。参考平台的示例包括：模型-视图-控制器 Web 应用程序的框架、支持事务后端系统的库、Web 服务平台的框架、客户端-服务器应用程序的基本框架、带可插入业务逻辑的中间件框架等。

构建初始参照平台的另一方法是在生命周期早期选择特定的项目，并让通晓安全问题的员工从事这些项目，以常用方式构建安全功能，从而能够从项目中提取此功能并用于组织中的其他项目。

无论采用何种创建方法，参照平台都具有加快审核和安全性相关审查的速度、提高开发效率以及降低维护开销的优点。

架构师、高级开发人员和其他利益相关方应参与设计和创建参照平台。创建平台后，必须有一个团队负责继续提供支持和更新。

验证框架、模式和平台的使用

在项目日常审核期间对项目工件执行额外的分析，以衡量推荐框架、设计模式和参照平台的使用情况。虽然是在日常审核期间进行该工作，但其目标是尽可能多地从项目团队收集反馈，以衡量它们各自积极的安全工作。

总体而言，对项目团队验证几个因素非常重要。确定使用不推荐的框架，以确定推荐框架与组织对功能的需求之间是否存在差距。检查未使用或错误使用的设计模式和参照平台模块以确定是否需要更新。此外，随着公司的发展，项目组希望在参照平台上执行更多或不同的功能。

任何通晓安全问题的技术人员都可以执行此分析操作。管理人员和利益相关方应对从每个项目收集的衡量指标进行整理，以用于分析。

结果

- 可提供内置安全保护的自定义应用程序开发平台
- 整个组织都期望在开发过程中针对安全性做出更积极的努力
- 利益相关方更有能力根据组织对安全设计的需求做出权衡决策

其他成功标准

- 50%以上的活动项目使用参照平台
- 过去 6 个月中，80%以上的项目报告了框架、模式和平台使用反馈
- 对于项目组报告指南/平台的有用性，大于 3.0 Likert

其他成本

- 扩充或许可参照平台
- 持续维护和支持参照平台
- 正在进行的项目审核期间的使用验证开销

其他人员

- 管理人员（1 天/年）
- 企业所有者（1 天/年）
- 架构师（3-4 天/年）
- 开发人员（2-3 天/年）
- 安全审核员（2 天/年）

相关目标

- 标准与遵从-2
- 体系结构审查-3
- 代码审查-3
- 安全测试-3

相关标准遵从

体系结构审查

体系结构审查功能注重检查软件设计和体系结构模型的安全问题。这使组织可在软件开发的早期检测到体系结构级别的问题，从而避免进行重构可能带来的巨大成本。

组织最开始会采取简便的工作来了解有关体系结构安全相关的详情，然后发展到采取更正式的检查方法来验证是否提供了完整的安全机制。在组织级别，建立了体系结构审查服务并提供给利益相关方。

在复杂的形式中，体系结构审查功能涉及到了对设计进行详细的、数据级检查，以及在版本被接受前对评估结果执行标准的期望。

	AR 1	AR 2	AR 3
目标	**支持对软件体系结构进行专门的审查，确保已基本排除了已知风险。**	**提供评估服务以根据广泛的安全最佳实践审查软件设计。**	**需要评估和验证工件以详细了解保护机制。**
措施	确定软件攻击面 根据已知的安全要求分析设计	检查是否完整提供了安全机制 为项目团队部署设计审查服务	为敏感资源开发数据流图表 为体系结构审查建立发布关卡
结果	非常了解周边体系结构隐含的安全问题 支持开发团队自行检查设计是否符合安全性最佳实践 执行项目级体系结构审查的简便流程	正式提供评估服务以持续审查体系结构的安全性 查出维护模式和旧系统中的安全缺陷 加深了项目利益相关方对软件提供保证保护方式的理解	系统设计薄弱点的细化视图可鼓励进行更好的划分 整个组织都注意到项目的体系结构应符合基本安全期望 在排除已知缺陷方面，对各项目的效率和进展进行了比较

体系结构审查

AR 1

支持对软件体系结构进行专门的审查，确保已基本排除了已知风险。

措施

确定软件攻击面

对于每一个软件项目，创建整体体系结构的简化视图。通常情况下，应基于项目对象创建此视图，例如高级需求和设计文档、采访技术人员或基本代码的模块级审查。捕获系统中的高级模块非常重要，但是良好的细化缩略图规则是确保要审查的整个系统的图表可在一页中显示。

在单一页面的体系结构视图中，分析每个组件的界面对于授权用户、匿名用户、操作员和应用程序特定的角色等的可访问性。在单页视图的上下文中应考虑提供界面的组件，以在图表上查找功能委派点或传递到其他组件的数据。对具有相似可访问性配阿置文件的界面和组件进行分组，并捕获它作为软件攻击面。

对于每个界面，进一步阐述单页图表，记录任何与安全性相关的功能。基于组成攻击面的已识别的界面组，检查模型的设计级一致性，以及如何保护具有相似访问权限的界面。任何违反一致性的信息都会记录在评估结果中。

应由项目组中或项目组外通晓安全问题的技术人员来执行此分析。通常情况下，在初始创建后，当对边缘系统界面进行添加或更改时，只需在设计阶段对图表和攻击面分析进行更新。

根据已知的安全要求分析设计

应确定和收集安全要求，无论是已正式确认的还是非正式已知的。另外，确定并包含安全操作系统所依靠的任何安全假设。

根据系统体系结构的单页图表，审查已知安全要求列表上的每一项。阐述该图表，显示针对每条安全要求的设计级功能。如果系统较大和/或比较复杂，则可创建独立的细分图表，以简化捕获此信息的操作。总体目标是验证系统设计已满足的每一条已知的安全要求。任何未在设计级清楚提供的安全要求都应记录为评估结果。

应由通晓安全问题的技术人员执行此分析，并按照需要要求架构师、开发人员、管理人员和组织所有人的介入。当安全要求或高级系统设计有所更改时，应在设计阶段对其进行更新。

结果

- 非常了解周边体系结构隐含的安全问题
- 支持开发团队自行检查设计是否符合安全性最佳实践
- 执行项目级体系结构审查的简便流程

成功标准

- 过去十二个月中，50%以上的项目更新了攻击面分析
- 过去十二个月中，50%以上的项目更新了安全要求设计级分析

成本

- 扩充和维护每个项目的体系结构图表
- 正在进行的项目检查攻击面和安全要求设计的开销

人员

- 架构师（2-3 天/年）
- 开发人员（1-2 天/年）
- 管理人员（1 天/年）
- 安全审核员（1 天/年）

相关目标

- 安全要求-1

相关标准遵从

-

AR 2

体系结构审查

提供评估服务以根据广泛的安全最佳实践审查软件设计。

结果

- 正式提供评估服务以持续审查体系结构的安全性
- 查出维护模式和旧系统中的安全缺陷
- 加深了项目利益相关方对软件提供保证保护方式的理解

其他成功标准

- 过去六个月中，80%以上的利益相关方了解了审查请求的状态
- 过去十二个月中，75%以上的项目经过了体系结构审查

其他成本

- 扩充、培训和维护设计审查团队
- 正在进行项目审查活动的开支

其他人员

- 架构师（1-2 天/年）
- 开发人员（1 天/年）
- 管理人员（1 天/年）
- 安全审核员（2-3 天/年）

相关目标

- 培训与指导-2
- 战略规划-2

相关标准遵从

措施

检查是否完整提供了安全机制

对于高级体系结构图表中模块上的每个界面，正式迭代安全机制列表，并分析系统安全机制的提供情况。这种类型的分析应对内部界面（如层之间）和外部界面（如组成攻击面的界面）进行。

需要考虑的六个主要安全机制为认证、授权、输入验证、输出编码、错误处理和日志记录。在相关的地方还要考虑加密机制和会话管理。对于每个界面，都需要确定提供每个机制的系统设计位置，并将缺失或不清楚的功能记录为结果。

应由通晓安全问题的员工执行此分析，项目团队应帮助提供应用程序特定的知识。此分析应每次发行软件时执行一次，通常是在设计阶段的结束时进行。初始分析后，后续版本需要根据开发周期中所作的更改对结果进行更新。

为项目团队部署设计审查服务

设计可使利益相关方请求体系结构审查的流程。此服务可在组织中集中提供或通过现有员工分布式提供，但是必须就执行完整持续的审查对所有的审查人员进行培训。

此审查服务应被集中管理，由熟悉组织整体业务风险概况的高级管理人员、架构师和利益相关方对它们的优先级进行分类。这使得项目审查的优先级与整体业务风险能够相对应。

在设计审查期间，审查团队应与项目团队合作，收集足够的信息以阐明对攻击面的理解、将项目特定的安全要求与设计元素相匹配，并验证模块界面上的安全机制。

体系结构审查

AR 3

需要评估和验证工件以详细了解保护机制。

措施

为敏感资源开发数据流图表

根据软件项目的业务功能，执行分析以围绕高风险功能确定系统行为的详细信息。通常情况下，高风险的功能会与执行创建、访问、更新和删除敏感数据的特性相对应。除数据之外，高风险功能还包括项目特定的、拒绝服务或危害方面的关键业务逻辑。

对于每个已确定的数据源或业务功能，选择和使用标准符号来捕获相关的软件模块、数据源、参与者以及这些因素之间传递的消息。以高级设计图表开始，不断充实相关详细信息，同时删除与敏感资源不对应的元素，这种方式通常很有帮助。

为项目创建数据流图表后，对其进行分析以确定设计中的内在瓶颈。一般说来，这些瓶颈是处理不同敏感度级别数据的单个软件模块，或是不同业务关键性级别的几个业务功能的访问关卡。

为体系结构审查建立发布关卡

建立一致的体系结构审查计划后，下一步就是设定软件开发生命周期中的一个特定的点。在执行体系结构审查和审查结果被接受之前，项目无法通过这个点。为完成这一步，应设置期望的基准水平，例如不允许具有高严重性级别结果的项目通过，其他任何结果都必须为组织所有人接受。

总体而言，体系结构审查应在设计阶段的结尾执行，以帮助尽早检测出安全问题。但是审查必须在设计团队可生成版本之前进行。

对于旧系统或不活动的项目，应当创建例外流程以允许这些项目继续操作，但是要对每个要审查的内容明确分配时间框架，以阐明现有系统中的任何隐藏漏洞。例外应限制在不大于所有项目的 20%。

结果

- 系统设计薄弱点的细化视图可鼓励进行更好的划分
- 整个组织都注意到项目的体系结构应符合基本安全期望
- 在排除已知缺陷方面，对各项目的效率和进展进行了比较

其他成功标准

- 过去六个月中，80%以上的项目更新了数据流图表
- 过去六个月中，75%以上的项目通过了体系结构审查审核

其他成本

- 持续维护数据流图图表的项目开销
- 由于体系结构审查审核失败造成项目延迟带来的组织开销

其他人员

- 开发人员（2 天/年）
- 架构师（1 天/年）
- 管理人员（1-2 天/年）
- 企业所有者（1-2 天/年）
- 安全审核员（2-3 天/年）

相关目标

- 防御设计-3
- 代码审查-3

相关标准遵从

代码检查

代码审查功能注重于在源代码级别检查软件，目的是查找安全漏洞。代码级漏洞的概念通常简单易懂，但是即便是有经验的开发人员也容易犯错误，使软件可能出现漏洞。

在开始，组织采用简便的检查表，并且为提高效率，只检查最为关键的软件模块。然而，随着组织的发展，它会使用自动技术来极大地改善代码审查活动的覆盖率和功效。

该功能提供的复杂内容涉及将代码审查更深地集成到开发流程，以使项目团队更易于发现问题。它还使组织能够在发行版本之前，针对代码审查结果更好地审核及设定预期。

	CR 1	CR 2	CR 3
目标	**随机查找基本代码级漏洞和其他高风险安全问题。**	**通过自动检查使开发过程中的代码审查更为准确、高效。**	**必须进行全面的代码审查过程，以发现语言级别的风险和特定于应用程序的风险。**
措施	根据已知的安全要求创建审查检查表 对高风险代码执行定点审查	利用自动代码分析工具 将代码分析集成到开发流程	针对应用程序特定的问题自定义代码分析 为代码审查建立发布关卡
结果	检查可能会导致被发现或被攻击的普通代码漏洞 对可能导致严重安全影响的代码错误进行简便审查 针对安全保证的基本代码级审慎调查	开发过程实现了对代码级别安全漏洞的持续自检 对结果进行日常分析，以对每个团队的安全代码习惯的历史数据进行编译 利益相关方注意到未牵制的漏洞，从而做出更好的权衡分析	对代码分析结果的准确性和适用性的信心增强 整个组织对安全编码期望的基准 项目团队对判断代码级安全性设立了目标

代码检查

CR 1

随机查找基本代码级漏洞和其他高风险安全问题。

措施

根据已知的安全要求创建审查检查表

根据已知的项目安全要求，导出简便的代码安全性审查检查表。这些可能是特定于围绕功能要求的安全问题的检测，或针对基于实施语言、平台、典型技术堆栈等的安全代码最佳实践的检测。由于这些变化因素，通常需要一系列检查表来涵盖组织中不同类型的软件开发。

无论是从可用的公共资源进行创建还是购买，技术利益相关方（如开发管理人员、架构师、开发人员和安全审核员）都应审查检查表，以确保功效和可行性。保持列表简短非常重要，这样做的目的是手动或使用简单的搜索工具捕获可直接在代码中找到的高优先级问题。代码分析自动化工具也可用来实现这一目的，但是还应对其进行自定义以将安全检测的整体集合缩小为有价值的小集合，以使扫描和审查流程高效运行。

应对开发人员阐明适合其作业功能的检查表的目标。

对高风险代码执行定点审查

由于如果代码级漏洞发生在软件的关键安全性部分，这些漏洞会极大地增大其影响，项目团队应审查高风险模块，查找普通漏洞。高风险功能的常见示例包括认证模块、访问控制强制点、会话管理方案、外部界面、输入验证器和数据解析器等等。

利用代码审查检查表，该分析可作为开发流程的一个普通部分执行，当进行更改时，项目团队的成员会被分配给要审查的模块。还可利用安全审核员和自动化审查工具进行审查。

在更改和审查高风险代码的开发周期期间，开发管理人员应在其他项目利益相关方的介入下，对结果进行分类并适当地将修复工作进行优先级排序。

结果

- 检查可能会导致被发现或被攻击的普通代码漏洞
- 对可能导致严重安全影响的代码错误进行简便审查
- 针对安全保证的基本代码级审慎调查

成功标准

- 过去六个月中，80%以上的项目团队简要介绍了相关代码审查检查表
- 过去六个月中，50%以上的项目团队对高风险代码执行了代码审查
- 对于项目团队报告代码审查检查表的有用性，大于 3.0 Likert

成本

- 扩充或许可代码审查检查表
- 对高风险代码持续执行代码审查工作的项目开销

人员

- 开发人员（2-4 天/年）
- 架构师（1-2 天/年）
- 管理人员（1-2 天/年）
- 企业所有者（1 天/年）

相关目标

- 安全要求-1

相关标准遵从

-

CR 2

代码检查

通过自动检查使开发过程中的代码审查更为准确、高效。

结果

- 开发过程实现了对代码级别安全漏洞的持续自检
- 对结果进行日常分析，以对每个团队的安全代码习惯的历史数据进行编译
- 利益相关方注意到未牵制的漏洞，从而做出更好的权衡分析

其他成功标准

- 过去六个月中，50%以上的项目经过了项目审查和利益相关方认可
- 过去一个月中，80%以上的项目可访问自动代码审查结果

其他成本

- 代码分析解决方案的研究和选择
- 自动集成的初始成本和维护
- 持续进行自动代码审查和抑制的项目开销

其他人员

- 开发人员（1-2 天/年）
- 架构师（1 天/年）
- 管理人员（1-2 天/年）
- 安全审核员（3-4 天/年）

相关目标

相关标准遵从

措施

利用自动代码分析工具

许多代码级别的安全漏洞复杂而难以理解，要发现它们必须进行仔细的检查。然而，可使用许多有用的自动化解决方案来自动分析代码的 bug 和漏洞。

可使用商用产品和开源产品，以涵盖普遍的编程语言和框架。基于以下几个因素选择适当的代码分析解决方案，包括检测的深度和准确度、产品可用性和用途模型、可扩展性和自定义功能、对组织体系结构和技术堆栈的适用性等。

在选择流程中采用通晓安全问题的技术人员以及开发人员和开发管理人员的介入，由利益相关方来审查整体结果。

将代码分析集成到开发流程

一旦选中代码分析解决方案后，必须将其集成到开发流程，鼓励项目团队利用其功能。要做到这一点的一个有效方法是设置基础架构，使扫描在创建时自动运行，或从项目系统代码库的代码自动运行。这样，可更早得到结果，使开发团队能在发布前进行自检。

旧系统或正在进行的大型项目的一个潜在问题就是代码扫描器通常会报告模块中的结果，而在此版本中这些模块不会更新。如果将自动扫描设置为定期运行，避免审查开销的有效策略为将考虑的结果限制在自上次扫描后添加、删除或更改的结果。然而，如果不忽略其他结果非常关键，则开发管理人员应与安全审核员、利益相关方和项目团队合作，制定一个具体计划以解决其他结果的问题。

如果在发布时仍然有代码审查未能解决的结果，则利益相关方应审查并接受这些结果。

代码检查

CR 3

必须进行全面的代码审查过程，以发现语言级别的风险和特定于应用程序的风险。

措施

针对应用程序特定的问题自定义代码分析

代码扫描工具由内置的规则知识库支持和推动，以基于语言 API 和常用库来检查代码，但是这些工具理解自定义 API 和应用于模拟检测的设计的能力有限。然而，通过自定义，代码扫描器可成为查找组织和项目特定的安全问题的强大的通用分析引擎。

各种工具在自定义分析的易用性和能力方面的详细信息各不相同，通常情况下代码扫描器自定义都涉及指定在特定的 API 和功能调用站点执行各种检测。这些检测包括分析内部编码标准的遵循情况、传递到自定义界面的未检测的受感染数据、对敏感数据处理的跟踪和验证、内部 API 的正确用法等。

从共享代码库用法的检查器开始自定义扫描器非常有效，因为已创建的检查器可在多个项目中使用。要为代码库自定义工具，安全审核员应检测代码和高级设计以确定候选的检查器，并与开发人员和利益相关方讨论实施问题。

为代码审查建立发布关卡

要设置所有软件项目的代码级安全基准，应在软件开发生命周期中确定一个特定的点作为检查点，其中应满足代码审查结果的最低标准，以创建一个版本。

首先，此标准应比较直观，例如，通过选择一两个漏洞类型来设定标准：具有相应结果的项目不能通过。随着时间过去，应通过添加额外的通过测试点的标准来改进此基准。

通常情况下，代码审查检查点应发生在实施阶段的末尾、发布之前。

对于旧系统或不活动的项目，应创建一个例外流程以允许这些项目继续操作，但是要明确指定抑制结果的时间框架。例外应限制在不大于所有项目的 20%。

结果

- 对代码分析结果的准确性和适用性的信心增强
- 整个组织对安全编码期望的基准
- 项目团队对判断代码级安全性设立了目标

其他成功标准

- 50%以上的项目正在使用代码分析自定义
- 过去六个月中，75%以上的项目通过了代码审查审核

其他成本

- 扩充和维护自定义代码审查检测
- 持续进行代码审查审核的项目开销
- 由于代码审查审核失败造成项目延迟带来的组织开销

其他人员

- 架构师（1 天/年）
- 开发人员（1 天/年）
- 安全审核员（1-2 天/年）
- 企业所有者（1 天/年）
- 管理人员（1 天/年）

相关目标

- 标准与遵从-2
- 防御设计-3

相关标准遵从

安全性测试

安全性测试功能专注于在运行时环境中操作的同时进行软件检查，目标是发现安全问题。通过在与预期运行软件的上下文相同的上下文中检测软件，这些测试活动将支持软件的保证案例，从而可使在其他情况下难以发现的错误操作配置和业务逻辑中的错误变得直观。

组织开始会采用基于软件基本功能的高级测试案例规范，然后会发展到采用安全性测试自动操作，以涵盖可证明系统中漏洞的广泛且多种多样的测试案例。

在高级形式中，此设置提供的内容涉及自定义测试自动化，以构建一组涵盖应用程序特定问题详细信息的安全性测试。安全性测试提供了组织级的额外可视性，支持组织在项目发布被接受前设置对安全性测试结果的最低预期。

	ST 1	ST 2	ST 3
目标	**启动测试过程，以根据软件需求进行基本的安全测试。**	**通过自动检查使开发过程中的安全测试更为完整、高效。**	**部署前需要进行特定于应用程序的测试以确保基本安全。**
措施	• 从已知安全要求推出测试案例 • 明确评估已知的安全性测试案例	• 利用自动安全性测试工具 • 将安全性测试集成到开发流程	• 采用应用程序特定的安全测试自动化 • 建立安全测试发布关卡
结果	📖独立验证围绕关键业务功能的预期安全机制 📖针对安全性测试的高级审慎调查 📖每一个软件项目安全性测试套件的特定增长	📖更深入和一致的软件功能安全性验证 📖允许开发团队在发布前进行自检和纠正问题 📖制定风险接受决策时，利益相关方最好要了解开放式漏洞	📖面对攻击的预期应用程序性能的组织整体基准 📖旨在改善自动分析准确性的自定义安全性测试套件 📖项目团队了解抵抗攻击的目标

安全性测试

ST 1

启动测试过程，以根据软件需求进行基本的安全测试。

措施

从已知安全要求推出测试案例

从项目的已知安全要求确定一系列测试案例，以检查软件功能的准确性。通常情况下，这些测试案例是从围绕功能要求和系统业务逻辑的安全问题中推出的，但是还应该包含针对基于实施语言或技术堆栈的常见漏洞的普通测试。

通常，使用项目团队的时间来构建应用程序特定的测试案例和利用可用的公共资源或购买的知识库，以选择适用的常见安全性测试案例最为高效。虽然未要求，也可利用自动安全性测试工具以涵盖常见的安全性测试案例。

此测试案例规划应发生在需求和/或设计阶段，且必须在发布前的最终测试之前。应由相关开发、安全性和质量保证人员来审查候选测试案例的可应用性、功效和可行性。

明确评估已知的安全性测试案例

通过使用为每个项目确定的安全性测试案例集，应进行测试，以对每个案例的系统性能进行评估。通常，此评估应发生在发布前的测试阶段。

指定安全性测试案例后，可由通晓安全问题的质量保证人员或开发人员执行，但是项目团队首次执行安全性测试案例必须在安全审核员的监控下进行，以便为团队成员提供帮助和指导。

在发布或部署前，利益相关方必须审查安全性测试的结果，并接受在发布时安全性测试失败带来的风险。如果在部署之前，则必须设定具体的时间线，以解决时间差距问题。

结果

- 独立验证围绕关键业务功能的预期安全机制
- 针对安全性测试的高级审慎调查
- 每一个软件项目安全性测试套件的特定增长

成功标准

- 过去十二个月中，50%以上的项目指定了安全性测试案例
- 过去六个月中，50%以上的利益相关方了解了项目的安全性测试状态。

成本

- 扩充或许可安全测试案例
- 持续维护和评估安全性测试案例的项目开销

人员

- QA 测试人员（1-2 天/年）
- 安全审核员（1-2 天/年）
- 开发人员（1 天/年）
- 架构师（1 天/年）
- 企业所有者（1 天/年）

相关目标

- 安全要求-1

相关标准遵从

-

ST 2

安全性测试

通过自动检查使开发过程中的安全测试更为完整、高效。

结果

- 更深入和一致的软件功能安全性验证
- 允许开发团队在发布前进行自检和纠正问题
- 制定风险接受决策时，利益相关方最好要了解开放式漏洞

其他成功标准

- 过去六个月中，50%以上的项目经过了安全性测试和利益相关方认可
- 过去一个月中，80%以上的项目可访问自动安全性测试结果

其他成本

- 搜索和选择自动安全测试解决方案
- 自动集成的初始成本和维护
- 持续进行自动安全测试和抑制的开销

其他人员

- 开发人员（1 天/年）
- 架构师（1 天/年）
- 管理人员（1-2 天/年）
- 安全审核员（2 天/年）
- QA 测试人员（3-4 天/年）

相关目标

相关标准遵从

措施

利用自动安全性测试工具

为了测试安全问题，必须对每个软件界面检测可能数量巨大的输入案例，这样会使使用手动测试案例实施和执行的有效安全测试难以控制。应使用自动安全测试工具来自动测试软件，使安全测试更为高效，结果质量更高。

可使用商业产品和开源产品，应对对其在组织中的适合性进行审查。选择合适的工具取决于多个因素，包含内置安全测试案例的健壮性和准确度，对于组织非常重要的测试体系结构类型的功效、对于更改或添加测试案例的自定义、对于开发机构而言结果的质量和可用性等等。

在选择流程中，利用通晓安全问题的技术人员以及开发人员和质量保证人员的参与，由利益相关方来审查整体结果。

将安全性测试集成到开发流程

采用工具来运行自动安全测试，在开发期间，组织中的项目应例行运行安全测试和审查结果。为了以低开销使其具有扩展性，应将安全测试工具配置为日常自动运行（例如每晚或每周），且在发生结果时对其进行检查。

只要对需求和设计阶段有益，就要及早执行安全测试。这种测试驱动的开发方法传统上用于功能测试案例，但它涉及到了在开发周期早期，特别是在设计期间确定和运行相关安全测试案例。自动执行安全测试案例后，项目即进入实施阶段，对不存在的功能的一些测试失败。所有的测试都通过时，实施完成。这就在开发周期早期为开发人员树立了一个明确直观的目标，从而可以降低由于安全问题或强制接受风险以满足项目最终期限而造成的推迟发布的风险。

对于每一次项目发布，都应将自动或手动安全测试的结果提供给管理层和组织利益相关方进行审查。如果在发布内容中仍然有未解决的结果作为接受的风险，则利益相关方和开发管理人员应进行合作，为解决这些问题建立一个具体的时间框架。

安全性测试

ST 3

部署前需要进行特定于应用程序的测试以确保基本安全。

措施

采用应用程序特定的安全测试自动化

通过自定义安全测试工具、增强普通测试案例执行工具或扩充自定义测试工具，项目团队应正式通过安全要求进行迭代并构建一系列自动检验器，来测试已实施的业务逻辑的安全性。

另外，如果对许多自动安全测试工具进行自定义以使它们可理解关于正在进行测试的项目中的特定软件界面的更多详细信息，则可极大地改善其精度和覆盖深度。此外，可将特定于组织的标准遵从或技术问题编码为一组可重用的中心测试，使得可以更容易地查看审核数据集合和每个项目的管理情况。

项目团队应注重根据其软件的业务功能扩充安全性测试案例的详细内容，而安全审核员领导的组织级团队应注重为标准遵从和内部标准指定自动测试。

建立安全测试发布关卡

为避免即将发行的软件含有容易发现的安全 bug，应在软件开发生命周期中确定一个特定点作为检查点，已建立的安全测试案例必须通过该点才能从项目中发布。这就为各种类型的安全测试树立了一个基准，预期所有的项目都应该通过该基准。

由于最初添加过多测试案例可导致开销成本增加，可从选择一两个安全问题入手，纳入各种类型的测试案例。按照预期，如果有任何测试没通过，则项目无法通过。时间一长，可通过选择额外的安全问题并添加各种相应的测试案例来改进此基准。

通常情况下，安全测试检查点应发生在实现或测试阶段的末尾、发布之前。

对于旧系统或不活动的项目，应创建一个例外流程以允许这些项目继续操作，但是要明确指定抑制结果的时间框架。例外应限制在不大于所有项目的 20%。

结果

- 面对攻击的预期应用程序性能的组织整体基准
- 旨在改善自动分析准确性的自定义安全性测试套件
- 项目团队了解抵抗攻击的目标

其他成功标准

- 50%以上的项目正在使用安全测试自定义
- 过去六个月中，75%以上的项目通过了所有的安全测试

其他成本

- 扩充和维护对安全测试自动化的自定义
- 持续进行安全测试审核流程的项目开销
- 由于安全测试审核失败造成项目延迟带来的组织开销

其他人员

- 架构师（1 天/年）
- 开发人员（1 天/年）
- 安全审核员（1-2 天/年）
- QA 测试人员（1-2 天/年）
- 企业所有者（1 天/年）
- 管理人员（1 天/年）

相关目标

- 标准与遵从-2
- 防御设计-3

相关标准遵从

-

漏洞管理

漏洞管理功能注重关于处理漏洞报告和操作事件的组织内部流程。流程就位后，通常会导致响应杂乱无序且没有通知的事件会支持组织中的项目具有一致的预期和更高的响应效率。

从发生事件时简单地分配角色入手，组织会逐步形成更为正式的事件响应流程，以确保可以看到并跟踪发生的问题。另外还要促进交流，使更多的人理解该流程。

在高级形式中，漏洞管理涉及对事件和漏洞报告详尽彻底的分析，以收集详细的衡量指标来反馈给组织的下游行为。

	VM 1	VM 2	VM 3
目标	**理解响应漏洞报告或事件的高级别计划。**	**阐述对响应过程的期望，以改善一致性和通信。**	**改善响应过程的分析和数据收集，为积极规划提供反馈。**
措施	确定安全问题的联络点 创建非正式安全响应团队	建立一致的事件响应流程 采用安全问题披露流程	分析事件的根源 收集每一事件的衡量指标
结果	处理高优先级漏洞或事件的简便流程就位 供利益相关方通知和报告影响安全的事件的框架 处理安全问题的高级审慎调查	处理第三方漏洞报告的沟通计划 对软件操作员发布安全补丁的清晰流程 事件跟踪、处理和内部沟通的正式流程	每个事件后针对组织改进的详细反馈 对漏洞和损坏带来成本的粗略估计 利益相关方能够更好地根据历史事件趋势做出权衡决策

漏洞管理

VM 1

理解响应漏洞报告或事件的高级别计划。

措施

确定安全问题的联络点

对于组织中的每一个部门或每个项目团队，都需要建立一个联络点作为安全信息的交流中心。虽然通常此职责不会占用个人很多时间，但预先设定联络点的目的是为漏洞管理增加结构和治理。

可造成利用的事件示例包括：接收外部实体的漏洞报告，现场发生损坏或其他软件安全故障、内部发现高风险漏洞等。如果发生事件，则最近的联络点将作为受影响项目团队的额外资源和顾问，提供技术指导，并为其他利益相关方介绍抑制措施的进展。

联络点应该从通晓安全问题的技术人员或管理人员中选取，他们必须对组织中的软件项目有较广的知识面。这些分配的安全联络点的列表应集中维护，并至少每六个月更新一次。此外，发布和公布此列表使组织中的成员之间能够就安全问题更直接地请求帮助和进行合作。

创建非正式安全响应团队

从分配有安全联络点职责的个人列表中或从专门的安全工作人员中，选出一组人，作为集中式技术安全响应小组。该小组的职责包括直接负责安全事件或漏洞报告，并负责分类、抑制和向利益相关方报告。

除具有上述这些职责外，安全响应小组的成员还负责发生事件期间进行行政报告和与上级沟通。大多数情况下，安全响应团队可能都在执行这些职责，但是他们还必须具有足够的灵活性，能够快速响应，或必须存在一个流畅的延迟流程和其他组员的事件。

响应团队应至少每年举行一次会议，以简要介绍响应流程的安全联络点和项目团队对安全相关报告的高级预期。

结果

- 处理高优先级漏洞或事件的简便流程就位
- 供利益相关方通知和报告影响安全的事件的框架
- 处理安全问题的高级审慎调查

成功标准

- 过去六个月，组织的 50%以上部门了解了最近的安全联络点
- 过去 12 个月中，安全响应小组和联络点举行了一次以上的会议

成本

- 正在进行的项目中人员担任安全联络点角色造成的开销
- 确定适当的安全响应团队

人员

- 安全审核员（1 天/年）
- 架构师（1 天/年）
- 管理人员（1 天/年）
- 企业所有者（1 天/年）

相关目标

- 培训与指导-2
- 战略规划-3

相关标准遵从

-

VM 2

漏洞管理

阐述对响应过程的期望，以改善一致性和通信。

结果

- 处理第三方漏洞报告的沟通计划
- 对软件操作员发布安全补丁的清晰流程
- 事件跟踪、处理和内部沟通的正式流程

其他成功标准

- 过去六个月中，80%以上的项目组了解了事件响应流程
- 过去 6 个月中，80%以上的利益相关方了解了安全问题披露情况

其他成本

- 组织的当前事件响应流程开销

其他人员

- 安全审核员（3-5 天/年）
- 管理人员（1-2 天/年）
- 企业所有者（1-2 天/年）
- 支持/操作人员（1-2 天/年）

相关目标

相关标准遵从

措施

建立一致的事件响应流程

从非正式安全响应小组扩展开来，清楚地记录组织的事件响应流程以及团队成员可望遵循的步骤。此外，必须对安全响应小组的成员每年至少进行一次针对针对些材料的培训。

合理的事件响应流程具有几条原则，它们包括：初始分类以避免额外损坏、更改管理和补丁应用程序、管理项目人员和事件相关的其他人员、法庭证据收集和保存、限制将该事件透露给利益相关方、明确的向利益相关方报告机制和/或通信树等等。

安全响应人员应与开发团队合作，执行技术分析以验证关于每个事件或漏洞报告的事实和假设。同样，当项目团队检测到事件或高风险漏洞时，他们应遵循一个内部流程，与安全响应小组的成员取得联系。

采用安全问题披露流程

对于大多数组织而言，它们不愿意公开安全问题的消息，对安全问题的内部到外部通信应遵循几个重要方法。

首先且最常见的方法是为组织开发的软件创建和部署安全补丁。通常情况下，如果所有软件项目只是内部使用，则这种方法不怎么重要；但是对于在组织外操作该软件的所有环境而言，必须存在补丁发布流程。它应该规定多个因素，这些因素包括：发布补丁前的变更管理和回归测试、通知操作员/用户为补丁分配的关键性类别、使技术详细信息分散从而使攻击者无法直接加以利用，等等。

外部通信的另一个方法是利用报告组织软件中安全漏洞的第三方。通过采用和对外发布具有相应期限的预期流程，鼓励漏洞报告人员遵循负责的披露实践步骤。

最后，许多州和国家/地区的法律规定，要求组织公布涉及个人身份信息和其他类型敏感数据的数据盗窃事件。如果发生了这种事件，安全响应团队应与管理人员和组织利益相关方合作，确定适当的下一步操作。

漏洞管理

VM 3

改善响应过程的分析和数据收集，为积极规划提供反馈。

措施

分析事件的根源

虽然可能会花费大量时间，但应当将事件响应流程放大进行额外的分析，以确定关键的底层安全故障。这些根源可能是技术问题，例如代码级漏洞、配置错误等，也可能是人员/流程问题，例如社会工程、未能遵循程序等。

确定事件根源后，应将其作为工具使用，来在可能发生类似事件的组织中查找其他潜在缺陷。对于每一个已确认的缺陷，还应该给出预先抑制的建议，这是结束原始事件响应工作的一部分。

所有根据根源分析得出的建议应由管理层和相关利益相关方审查，以安排牵制工作活动或记录所接受的风险。

收集每一事件的衡量指标

通过采用集中流程处理所有损坏和高优先级漏洞报告，组织能够逐渐采纳趋势措施，以确定针对安全保证倡议的影响和效率。

应至少每 6 个月应存储和审查过去事件的记录。对相似的事件进行分组，并简单记录每种类型问题的总数。对事件采取的额外措施包括：软件项目受事件影响的频率、因无法使用而导致的系统停机和成本、处理和清除事件占用的人力资源、估算的长期成本（如管理费用或品牌损失）等。对于是技术问题的根源，确定哪种类型的主动、审查或操作实践最初检测到它或减轻了其损坏也很有帮助。

此信息是对程序规划流程的具体反馈，因为它代表组织长期以来所经受的真实安全影响。

结果

- 每个事件后针对组织改进的详细反馈
- 对漏洞和损坏带来成本的粗略估计
- 利益相关方能够更好地根据历史事件趋势做出权衡决策

其他成功标准

- 过去 6 个月中，记录了 80%以上事件根源和进一步建议
- 过去 6 个月中，对 80%以上事件的衡量指标进行了整理

其他成本

- 组织对事件持续进行更深入研究和分析的开销
- 组织持续整理和审查事件衡量指标的开销

其他人员

- 安全审核员（3 天/年）
- 管理人员（2 天/年）
- 企业所有者（2 天/年）

相关目标

相关标准遵从

基础架构强化

基础架构强化功能注重支持托管组织软件的运行时环境。因为软件的安全操作可能因外部组件问题而损坏，强化底层基础架构可直接改进组织软件的整体安全状态。

从简单跟踪和分发有关操作环境的信息以更好地通知开发团队入手，组织会逐步采纳可扩展的方法，管理安全补丁部署和用早期警告检测器装置操作环境，以在造成损坏前检测出潜在的安全问题。

随着组织的发展，可以部署保护工具，以添加防护层和安全网，从而限制漏洞被利用时造成的损失，通过这些操作环境将得到进一步审查和强化。

	IH 1	IH 2	IH 3
目标	**了解应用程序和软件组件的基本操作环境。**	**通过强化操作环境提高对应用程序操作的信心。**	**参照已知最佳实践验证应用程序运行状况和操作环境的状态。**
措施	维护操作环境规范 确定和安装关键的安全补丁	建立基础架构补丁管理流程 监控基准基础架构配置状态	确定和部署相关操作保护工具 扩展基础架构配置的审核程序
结果	开发团队清楚理解操作预期 按照被充分理解的时间表抑制了底层基础架构的高优先级风险 周密策划对基础架构进行安全维护的软件操作人员	对操作中的系统安全特性进行了粒度验证 正式预测基础架构风险抑制时间表 利益相关方持续关注软件项目的当前操作状态	通过分层检测安全性强化了操作环境 已确立并测量了操作维护和性能目标 通过外部依赖性的缺陷降低了成功攻击的可能性

基础架构强化

IH 1

了解应用程序和软件组件的基本操作环境。

措施

维护操作环境规范

对于每一个项目，应创建和维护预期操作平台的具体定义。根据组织的组成，此规范应由开发人员、利益相关方、支持和操作组等共同创建。

开始创建该规范应首先根据软件的商业功能捕获有关此操作环境的所有真实详细信息。这些信息可能包括以下因素，例如处理器体系结构、操作系统版本、必备软件、冲突软件等。此外，记录影响软件行为方式的任何已知用户或操作环境有关的操作人员配置选项。

另外，确定在项目设计和实施阶段提出的有关操作环境的假设，并在规范中捕获这些假设。

对于活动的项目，应至少每 6 个月对此规范进行审查和更新，而如果对此软件设计或预期的操作环境进行了更改，这个时间应该更短。

确定和安装关键的安全补丁

大多数应用程序是运行在另一大堆软件上的软件，这堆软件由内置编程语言库、第三方组件和开发框架、基本操作系统等组成。因为这大堆软件中任何模块的安全缺陷都会影响该组织软件的整体安全性，所以必须安装技术元素的重要安全更新。

同样，应对高风险依赖性执行常规搜索和持续监视，针对安全缺陷安装最新补丁。确定可影响软件项目安全状态的重要补丁后，应制定相应计划，使受影响的用户和操作人员更新其安装。根据软件项目的类型，此操作的详情可能不同。

结果

- 开发团队清楚理解操作预期
- 按照被充分理解的时间表抑制了底层基础架构的高优先级风险
- 周密策划对基础架构进行安全维护的软件操作人员

成功标准

- 过去 6 个月中，50%以上的项目更新了操作环境规范
- 过去 6 个月中，50%以上的项目更新了相关重要安全补丁的列表

成本

- 持续扩充和维护操作环境规范的项目开销
- 持续监视和安装重要安全更新的项目开销

人员

- 开发人员（1-2 天/年）
- 架构师（1-2 天/年）
- 管理人员（2-4 天/年）
- 支持/操作人员（3-4 天/年）

相关目标

- 操作实现-2

相关标准遵从

IH 2

基础架构强化

通过强化操作环境提高对应用程序操作的信心。

结果

- 对操作中的系统安全特性进行了粒度验证
- 正式预测基础架构风险抑制时间表
- 利益相关方持续关注软件项目的当前操作状态

其他成功标准

- 过去 12 个月中，80%以上的项目团队了解了补丁管理流程
- 过去 6 个月中，80%以上的利益相关方关注了当前补丁状态

其他成本

- 组织持续进行补丁管理和监视的开销
- 扩充和许可基础架构监视工具

其他人员

- 架构师（1-2 天/年）
- 开发人员（1-2 天/年）
- 企业所有者（1-2 天/年）
- 管理人员（1-2 天/年）
- 支持/操作人员（3-4 天/年）

相关目标

相关标准遵从

措施

建立基础架构补丁管理流程

转到比重要补丁特定的应用程序更正式的流程，应在组织中创建一个持续的流程，以在操作环境中持续将安全补丁应用于基础架构。

在最基本的形式中，此流程的目标是针对发布和应用安全补丁的时间间隔做出保证。为使此流程高效，通常组织对低优先级补丁接受高延迟，例如，重要补丁的延迟最多为 2 天，而低优先级补丁的期限最多为 30 天。

这项工作主要由支持和操作人员进行，但是也应当与开发团队的召开例会，以使项目与过去的更改和预定的升级保持一致。

另外，开发人员应共享软件项目内部依赖的第三方组件列表，从而使支持和操作人员可监视这些组件，并提示开发团队需要进行升级的时间。

监控基准基础架构配置状态

鉴于监视和管理组成软件项目基础架构的各种组件的补丁很复杂，应利用自动化工具来自动监视系统，以保证配置的有效性。

可使用商业工具和开源工具来提供这种类型的功能，因此项目团队应基于组织需求的适合性选择一个解决方案。典型的选择标准包括易于部署和自定义、组织平台和技术堆栈的适用性、变更管理和警报的内置功能、衡量指标收集和趋势追踪等。

除主机和平台检测外，应对监视自动化进行自定义，以执行应用程序特定的运行状况检测和配置验证。支持和操作人员应与架构师和开发人员合作，共同确定给定软件项目的最佳监视数量。

最后，部署解决方案以监视基础架构的配置状态后，应由项目利益相关方每周收集和审查意外警报或配置更改，最少不能低于每季度一次。

基础架构强化

IH 3

参照已知最佳实践验证应用程序运行状况和操作环境的状态。

措施

确定和部署相关操作保护工具

为了在其操作环境中为软件构建一个更好的保证案例，可使用额外的工具来增强系统整体的安全状态。各种操作环境可能会有极大的不同，因此应在项目环境中考虑给定保护技术的适当性。

常用的保护工具包括：Web 应用程序防火墙、面向 web 服务的 XML 安全网关、客户端/嵌入式系统的防篡改和模糊化包、针对旧基础架构的网络入侵检测/防御系统、法庭日志聚合工具、基于主机的完整性验证工具等。

根据组织和项目特定的知识，技术利益相关方应与支持和操作人员合作，以确定和为业务利益相关方推荐选定的操作保护工具。如果认为一项投资在降低风险对比实施成本方面有价值，则利益相关方应同意进行试验、广泛展示和持续维护的计划。

扩展基础架构配置的审核程序

执行例行项目级审核时，扩展审查以纳入对与强化操作环境相关的工件的检查。除操作环境的最新规范外，审核还应检查当前补丁状态和自上次审核以来的历史数据。通过接入监控工具，审核还能验证有关应用程序配置管理和历史更改的关键因素。审核还应针对该软件体系结构类型可用的操作保护工具检查这些工具的使用情况。

可在项目初始发布和部署后的任何时间进行基础架构审核，但应至少每 6 个月进行一次。对于旧系统或无活动开发工作的项目，仍应由组织利益相关方执行和审查基础架构审核。应创建一个例外流程，以允许特殊案例的项目继续操作，但必须明确分配抑制结果的时间框架。例外应限制在不大于所有项目的 20%。

结果

- 通过分层检测安全性强化了操作环境
- 已确立并测量了操作维护和性能目标
- 通过外部依赖性的缺陷降低了成功攻击的可能性

其他成功标准

- 过去 6 个月中，80%以上的利益相关方了解了相关操作保护工具
- 过去 6 个月中，75%以上的项目通过了基础架构审核

其他成本

- 操作保护解决方案的研究和选择
- 扩充和许可操作保护工具
- 持续维护保护工具的操作开销
- 持续进行基础架构相关审核的项目开销

其他人员

- 企业所有者（1 天/年）
- 管理人员（1-2 天/年）
- 支持/操作人员（3-4 天）

相关目标

- 标准与遵从-2

相关标准遵从

操作实现

操作实现功能注重从构建软件的项目团队收集重要的安全信息，并将其传达给该软件的用户和操作人员。如果没有此信息，则即便是设计最安全的软件也包含不合理的风险，因为在部署站点上无法了解重要的安全特性和选择。

从为用户和操作人员捕获最具影响的详细信息的简便文档记录入手，组织会逐渐构建随每个版本提供的完整操作安全指南。

在高级形式中，操作实现也需要对单个项目团队进行组织级检查，以确保根据预期捕获和共享此信息。

	OE 1	OE 2	OE 3
目标	**使开发团队和操作人员能够就关键安全相关数据进行交流。**	**通过制定详细的步骤来提高对持续安全操作的期望。**	**强制宣传安全信息，验证工件的完整性。**
措施	捕获部署的重要安全信息 记录典型应用程序警报的步骤	创建每次发布的变更管理步骤 维护正式操作安全指南	扩展操作信息的审核程序 对应用程序组件执行代码签名
结果	通过更好地理解正确操作实现了软件安全状态的特定改进 操作员和用户意识到在确保安全部署方面的职责 改善了软件开发人员和用户之间就重要安全信息的交流	针对随软件版本提供的安全相关更改的详细指南 更新了每个应用程序安全操作步骤的信息库 对开发人员、操作人员和用户的操作预期保持一致	整个组织都理解了应需要安全相关文档 利益相关方能更明智地根据部署和操作的反馈做出权衡决策 操作人员和/或用户能够独立验证软件版本的完整性

操作实现

OE 1

使开发团队和操作人员能够就关键安全相关数据进行交流。

措施

捕获部署的重要安全信息

利用特定的软件知识，项目团队应确认和安全相关的配置和操作信息，并将其传达给用户和操作人员。这样就使部署站点上软件的实际安全状态能够按照项目团队设计人员的意图运行。

此分析应从架构师和开发人员构建软件内置安全特性的列表入手。在该列表的基础上，还应捕获有关配置选项及其安全影响的信息。对于提供了多个不同部署模块的项目,应记录每个模块相关安全问题的信息，以更好地通知用户和操作人员其选择的影响。

整体而言，此列表应该非常简便，旨在捕获最重要的信息。最初创建了列表之后，应由项目团队和组织利益相关方进行审查同意。另外，与选定的操作人员和用户共同审查此列表会非常高效，目的是确保可理解和操作该信息。项目团队应对每个版本审查和更新此信息，但必须至少每 6 个月进行一次。

记录典型应用程序警报的步骤

借助软件行为方式的特定知识，项目团队应确定需要用户/操作人员注意的、最重要的错误和警报消息。对于每个已确定的事件，应捕获用户/操作人员为响应此事件应采取的适当操作。

对于软件会生成的可能很大的一组事件，根据软件业务目标的相关性选择优先级最高的集。它应该包含安全相关的事件，但也可能包含涉及软件运行状况和配置状态的重要错误和警报。

对于每一个事件，应捕获可操作的建议，以通知用户和操作人员所需的下一步操作的和事件的潜在根源。这些步骤必须由项目团队进行审查并每 6 个月更新一次，但也可以进行更频繁的审查和更新，例如每次发布时。

结果

- 通过更好地理解正确操作实现了软件安全状态的特定改进
- 操作员和用户意识到在确保安全部署方面的职责
- 改善了软件开发人员和用户之间就重要安全信息的交流

成功标准

- 过去 6 个月中，50%以上的项目更新了部署安全信息
- 过去 6 个月中，50%以上的项目更新了事件的操作程序

成本

- 持续进行项目维护部署安全信息的项目开销
- 持续进行维护重要操作程序的项目开销

人员

- 开发人员（1-2 天/年）
- 架构师（1-2 天/年）
- 管理人员（1 天/年）
- 支持/操作人员（1 天/年）

相关目标

相关标准遵从

OE 2

操作实现

通过制定详细的步骤来提高对持续安全操作的期望。

结果

- 针对随软件版本提供的安全相关更改的详细指南
- 更新了每个应用程序安全操作步骤的信息库
- 对开发人员、操作人员和用户的操作预期保持一致

其他成功标准

- 过去 6 个月中，50%以上的项目更新了更改管理程序
- 过去 6 个月中，80%以上的利益相关方简要介绍了操作安全指南的状态

其他成本

- 持续维护更改管理程序的项目开销
- 持续维护操作安全指南的项目开销

其他人员

- 开发人员（1-2 天/年）
- 架构师（1-2 天/年）
- 管理人员（1 天/年）
- 支持/操作人员（1 天/年）

相关目标

- 增强基础架构-1

相关标准遵从

-

措施

创建每次发布的变更管理步骤

要更正式地为用户和操作人员更新软件相关更改的信息，每个版本都必须包含与升级和首次安装相关联的变更管理步骤。总而言之，目标是捕获预期的随附步骤，这些步骤可确保部署成功，且不会导致过多的停机时间或安全状态的降级。

要在开发期间构建这些步骤，项目团队应设置简便的内部流程来捕获会影响部署的相关项。在开发周期早期使用此流程会非常高效，这样可在需求、设计和实施阶段确定此信息后立即保留。

每次发布前，项目团队应整体审查此列表以检查完整性和可行性。对于一些项目，给定版本所附带的大量变更步骤可授权特殊处理，例如构建自动升级的脚本以避免部署中发生错误。

维护正式操作安全指南

从捕获的重要软件事件的信息以及处理每个事件的步骤入手，项目团队应构建和维护正式指南，以捕获用户和操作人员需要了解到的所有安全相关的信息。

最初，应根据有关系统的已知信息（例如安全相关的配置选项、事件处理步骤、安装和升级指南、操作环境规范、有关部署环境的安全相关假设等）建立该指南。以此扩展开来，正式的操作安全指南应详细阐述上述每一项以涵盖更多信息，这样多数用户和操作人员都会了解到可能提出的所有问题。对于大型或复杂的系统，这可能很具挑战性，因此项目团队应与组织利益相关方合作，共同确定文档的适当级别。另外，项目团队应记录任何会增强安全性的部署建议。

操作安全性指南初始创建后，应由项目团队进行审查，并随每次发布进行更新。

操作实现

OE 3

强制宣传安全信息，验证工件的完整性。

措施

扩展操作信息的审核程序

执行例行项目级审核时，扩展审查以包含对涉及安全性操作实现工件的检查。应对项目进行检查，以确保其具有涉及软件细节的已更新的完整操作安全指南。

这些审核应在接近发布的开发周期结束处开始，但是必须在生成发布版本前完成并通过。对于旧系统或不活动的项目，应执行这种类型的审核并一次性解决结果，并验证审核标准遵从，随后则不再需要针对操作实现进行另外的审核。

发布前，必须由利益相关方对审核结果进行审查。应创建一个例外流程，以允许未通过一项审核的项目继续发布，但是这些项目应具有抑制结果的具体期限。例外应限制在不大于所有活动项目的 20%。

对应用程序组件执行代码签名

虽然经常与特殊目的的软件共同使用，代码签名支持用户和操作人员执行软件的完整性检查，这样它们就可以加密验证模块或版本的真实性。通过对软件模块进行签名，项目团队能使部署的操作得到更高的保证，避免已部署的软件在其操作环境中被损坏或修改。

对代码进行签名会造成组织管理签名凭证的开销。组织必须遵循安全密钥管理流程，以确保签名密钥的保密性。处理任何加密的密钥时，项目利益相关方还必须要考虑处理有关加密的常见操作问题（例如密钥循环、密钥泄露或密钥丢失）的计划。

由于代码签名并非对所有部件都适用，架构师和开发人员应与安全审核员和组织利益相关方合作，确定软件应进行签名的部件。随着项目的发展，每次发布都应对此列表进行审查，特别是当添加新模块或对以前签名的组件进行更改时。

结果

- 整个组织都理解了应需要安全相关文档
- 利益相关方能更明智地根据部署和操作的反馈做出权衡决策
- 操作人员和/或用户能够独立验证软件版本的完整性

其他成功标准

- 过去 6 个月中，80%以上的项目更新了操作安全指南
- 过去 6 个月中，80%以上的利益相关方了解了代码签名选项和状态

其他成本

- 持续审核操作指南的项目开销
- 组织持续管理代码签名凭证的开销
- 持续对代码模块进行确定和签名的项目开销

其他人员

- 开发人员（1 天/年）
- 架构师（1 天/年）
- 管理人员（1 天/年）
- 安全审核员（1-2 天/年）

相关目标

相关标准遵从

路线图

本节包含的路线图示例，详细描述了根据高级组织类型实现 SAMM 中目标的建议的顺序。建议的每个路线图都重点说明了交叉事项，以显示与建议不同的常见案例。

独立软件供应商 56

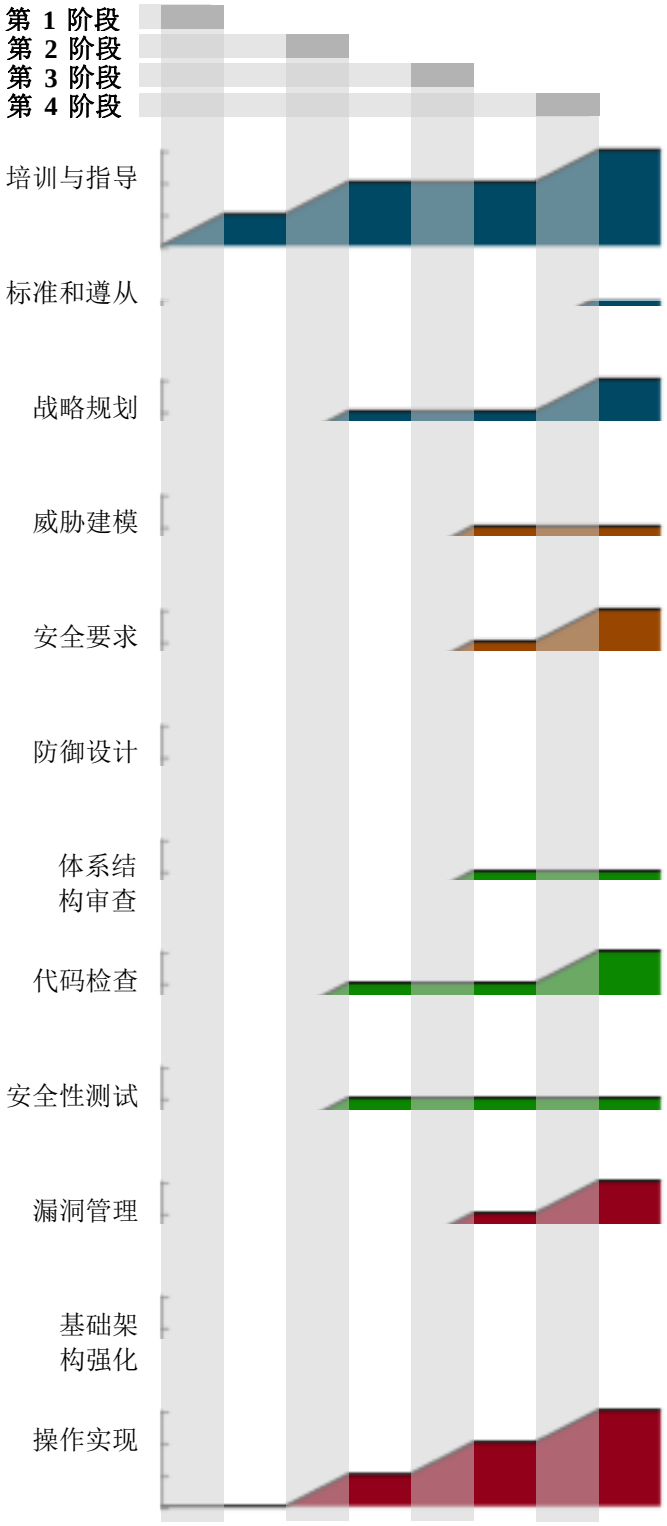
在线服务提供商 58

金融服务机构 ..... 即将推出

政府机构 ..... 即将推出

独立软件供应商

路线图示例



独立软件供应商 (Independent Software Vendor, ISV) 从事构建及销售软件组件和应用程序的核心业务。

由于最初目的是限制可影响客户和用户业务的常见漏洞，因而早期行动专注于代码审查和安全性测试。

在转向更主动地避免产品规格出现安全错误时，组织会逐渐增加针对安全要求的措施。

此外，为了尽量降低已发现的安全问题所造成的影响，组织会逐渐增加漏洞管理措施。

随着组织逐渐成熟，会增加“操作实现”功能的知识转移措施，以更好地为客户和用户提供软件安全操作信息。

第 1 阶段

对员工进行技术安全意识的培训
制定和维护技术指导制度
估计总体业务风险概况
EG 1 建立和维护保证方案
SP 1 根据业务功能，制定安全要求
使用指导原则、标准和标准遵从资源
SR 1 根据已知的安全要求，创建审查检查表
对高风险代码执行定点审查
CR 1 根据已知的安全要求，创建测试案例
明确评估已知的安全测试案例
ST 1 确定安全问题的联络点
创建非正式的安全响应团队
VM 1

第 2 阶段

对员工进行针对特定角色的应用程序安全培训
EG 2 聘用安全指导员，增强项目团队
根据业务风险对数据和应用程序进行分类
SP 2 建立每个分类的安全目标
建立并维护每个应用程序的攻击树
TM 1 根据软件体系结构，制定攻击者配置文件
确定软件攻击面
AR 1 根据已知的安全要求分析设计
利用自动代码分析工具
CR 2 将代码分析集成到开发流程
利用自动安全性测试工具
ST 2 将安全性测试集成到开发流程
捕获关于部署的重要安全信息
OE 1 记录典型应用程序警报的步骤

第 3 阶段

确定并监视外部标准遵从的驱动因素
SC 1 建立和维护标准遵从检查表
用应用程序特定的数据详细描述攻击树
TM 2 采用一种衡量威胁的权重系统
为每个项目建立和维护滥用案例模型
SR 2 根据已知风险，指定安全要求
维护推荐软件框架的列表
DD 1 将安全原则显式应用于外围界面
检查是否完整提供了安全机制
AR 1 为项目团队部署设计审查服务
建立一致的事件响应流程
VM 2 采用安全问题披露流程
创建每次发布的变更管理步骤
OE 2 维护正式的操作安全指南

第 4 阶段

创建应用程序安全支持门户网站
EG 3 建立基于角色的考试/认证制度
建立安全和标准遵从的内部标准
SC 2 建立项目审核实践
定期进行业界范围的成本比较
SP 3 收集历史安全费用的标准
将安全要求写入供应商协议
SR 3 扩展审核计划以满足安全需求
针对应用程序特定的问题自定义代码分析
CR 3 为代码审查建立发布关卡
分析事件的根源
VM 3 收集每一事件的衡量指标
扩展操作信息的审核程序
OE 3 对应用程序组件执行代码签名

对组织的路线图有影响的 ISV 具有一些共同的特征。

外包式开发

对于采用外部开发资源的组织，通常需要限制代码的访问权限，这会导致组织优先执行安全要求任务，而不会先执行代码审查任务。此外，如果在较早的阶段执行威胁建模，组织便可以向外包开发人员提出更明确的安全要求。由于有关软件配置的专业技术通常是外包组中最重要的，因此必须制定合同以说明与操作实现相关的措施。

连接 INTERNET 的应用程序

构建使用在线资源的应用程序的组织存在一些额外的风险，这些风险源自运行面向 Internet 系统的面向 Internet 的核心基础架构。考虑到这方面的风险，组织应该将基础架构强化措施添加到路线图中。

驱动程序与嵌入式开发

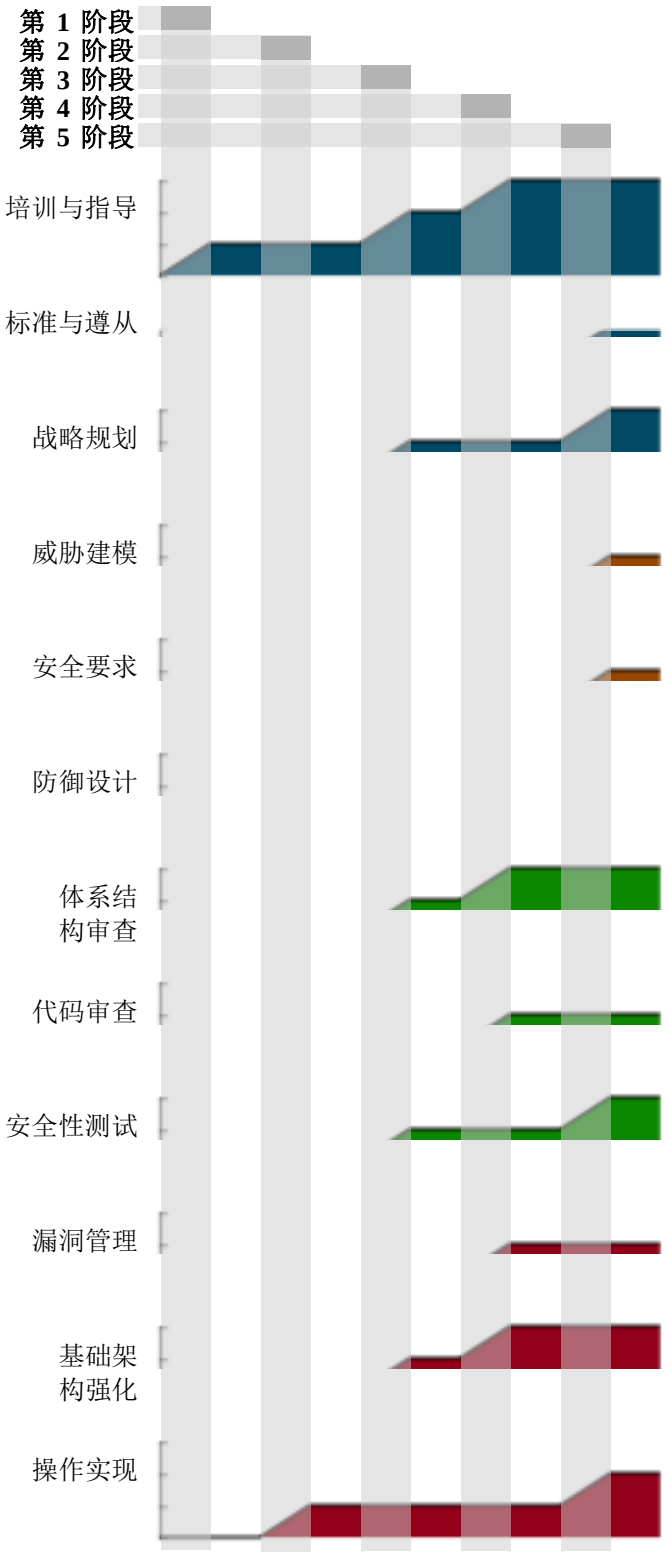
对于为嵌入式系统构建低级驱动程序或软件的组织而言，软件设计方面的安全漏洞更具破坏性，且修复成本更高。因此，必须将路线图修改为强调在早期阶段执行防御设计和体系结构审查措施。

通过收购而发展壮大的组织

在通过收购而发展壮大的组织中，通常存在遵循不同开发模式的若干个项目团队，其安全相关任务的等级也不同。对于这类组织，如果正在开发不同类型的软件，那么需要为各个部门或项目团队分别制定相应的路线图，以解决各个项目的起点和特定于项目的关注点不同这一问题。

在线服务提供商

路线图示例



在线服务提供商 (Online Services Provider, OSP) 从事构建 WEB 应用程序及其他网络可访问界面的核心业务。

由于最初目的是在不妨碍创新的前提下验证设计的整体稳固性，因而早期行动专注于体系结构审查和安全性测试。

由于重要的系统将面向网络，还应尽早执行基础架构强化措施，并逐渐解决托管环境所存在的风险。

尽管该措施基于组织的核心业务，但还是应该尽早启动标准与遵从措施，然后根据外部遵从驱动因素的关键性进行推进。

随着组织的逐渐成熟，可逐渐增加威胁建模、安全要求以及防御设计等措施，以便在确定基准安全预期后支持预先安全防御。

第 1 阶段

对员工进行技术安全意识的培训
EG 1 制定和维护技术指导制度
估计总体商业风险概况
SP 1 建立和维护保证方案
确定软件攻击面
AR 1 根据已知的安全要求分析设计
根据已知的安全要求，创建测试案例
ST 1 明确评估已知的安全测试案例
维护操作环境规范
OE 1 确定和安装关键的安全补丁

第 2 阶段

确定并监视外部标准遵从的驱动因素
SC 1 建立和维护标准遵从检查表
建立并维护每个应用程序的攻击树
TM 1 根据软件体系结构，制定攻击者配置文件
根据已知的安全要求，创建审查检查表
CR 1 对高风险代码执行定点审查
确定安全问题的联络点
VM 1 创建非正式的安全响应团队
捕获关于部署的重要安全信息
OE 1 记录典型应用程序警报的步骤

第 3 阶段

对员工进行针对特定角色的应用程序安全培训
EG 2 聘用安全指导员，增强项目团队
根据业务风险对数据和应用程序进行分类
SP 2 建立每个分类的安全目标
根据业务功能，制定安全要求
SR 1 使用指导原则、标准和标准遵从资源
检查是否完整提供了安全机制
AR 2 为项目团队部署设计审查服务
利用自动安全性测试工具
ST 2 将安全性测试集成到开发流程
建立基础架构补丁管理流程
IH 2 监视基准基础架构配置状态

第 4 阶段

创建应用程序安全支持门户网站
EG 3 建立基于角色的考试/认证制度
DD 1 维护推荐软件框架的列表
将安全原则显式应用于外围界面
AR 3 为敏感资源制定数据流图表
为体系结构审查建立发布关卡
CR 2 利用自动代码分析工具
将代码分析集成到开发流程
VM 2 建立一致的事件响应流程
采用安全问题披露流程
IH 3 确定和部署相关操作保护工具
扩展基础架构配置的审核程序

第 5 阶段

SC 2 建立安全和标准遵从的内部标准
建立项目审核实践
SP 3 定期进行业界范围的成本比较
收集历史安全费用的标准
TM 2 用应用程序特定的数据详细描述攻击树
采用一种衡量威胁的权重系统
SR 2 为每个项目建立和维护滥用案例模型
根据已知风险，指定安全要求
ST 3 采用应用程序特定的安全测试自动化
建立安全测试发布关卡
OE 2 创建每次发布的变更管理步骤
维护正式的操作安全指南

对组织的路线图有影响的 OSP 具有一些共同的特征。

外包式开发

对于采用外部开发资源的组织，通常需要限制代码的访问权限，这会导致组织优先执行安全要求任务，而不会先执行代码审查任务。此外，如果在较早的阶段执行威胁建模，组织便可以向外包开发人员提出更明确的安全要求。由于有关软件配置的专业技术通常是外包组中最重要的，因此必须制定合同以说明与操作实现相关的措施。

符合 PCI 标准的组织

对于必须满足支付卡行业数据安全标准 (Payment Card Industry Data Security Standard, PCI-DSS) 要求的组织，应该将标准与遵从措施放在路线图中的早期阶段。这使组织能够借机制定满足 PCI-DSS 标准要求的措施，还可以对以后的路线图进行相应的定制。

Web 服务平台

对于构建 Web 服务平台的组织，设计方面的错误将会带来其他风险，并且修复的成本更高。因此，应该将威胁建模、安全性要求和防御设计放在路线图的早期阶段。

通过收购而发展壮大的组织

在通过收购而发展壮大的组织中，通常存在遵循不同开发模式的若干个项目团队，其安全相关任务的等级也不同。对于这类组织，如果正在开发不同类型的软件，那么需要为各个部门或项目团队分别制定相应的路线图，以解决各个项目的起点和特定于项目的关注点不同这一问题。