软件保证成熟度模型

将软件的安全融入到软件开发过程的指导手册

版本－1.0

欲获得最新版本和其他信息，请访问项目网页 http://www.opensamm.org

鸣谢

软件保证成熟度模型（SAMM）最初是由独立软件安全顾问：Pravir Chandra（chandra@owasp.org）开创、设计、并编写的。该文档初稿的创作由Fortify软件公司赞助。该文档目前由Pravir Chandra所领导的OpenSAMM项目进行维护和更新。自SAMM最初版本的发布以来，这个项目已成为开放Web应用安全项目（OWASP）的一部分。另外，感谢那些列举在封底上给予我们支持的组织。

贡献者和审核者

如果没有那么多的审核人员和专家所给予的支持以及重要的反馈，这项工作就不会完成。他们是（以姓的英文首字母排序）：

Fabio Arciniegas	Brian Chess	Matteo Meucci	John Steven
Matt Bartoldus	Dinis Cruz	Jeff Payne	Chad Thunberg
Sebastien Deleersnyder	Justin Derry	Gunnar Peterson	Colin Watson
Jonathan Carter	Bart De Win	Jeff Piper	Jeff Williams
Darren Challey	James McGovern	Andy Steingruebl	

该中文版本参与人员

翻译及审核:王颉

（因笔者水平有限，欢迎大家指出存在的翻译错误。在以后发布的正式版本中，将修正指出的错误。）

该中文版发布说明

本文档为“Software Assurance Maturity Model (Version 1.0)”的中文Alpha版发布。该版本尽量提供原英文版本中的图片，并与原版本尽量保持相同的风格。存在的差异，尽情谅解。

这是一个OWASP的项目

开放Web应用安全项目（OWASP）是一个致力于改善应用软件安全的自由和开放的全球性社区。我们的任务是使应用程序的安全能够“看得见”，所以使人和机构可以针对应用安全的风险作出明智的决策。每个人都可以免费加入OWASP，我们所有的材料都基于免费和开放的软件许可证。OWASP基金会是一家501（c）3非营利的慈善机构，以确保持续的可用性和支持我们的工作。在线访问OWASP的网站http://www.owasp.org。

执行摘要

软件保证成熟度模型（SAMM）是一个开放的框架，用以帮助组织制定并实施针对组织所面临来自软件安全的特定风险的策略。由SAMM提供的资源可作用于以下方面：

✦评估一个组织已有的软件安全实践；
✦建立一个迭代的权衡的软件安全保证计划；
✦证明安全保证计划带来的实质性改善；
✦定义并衡量组织中与安全相关的措施。

SAMM以灵活的方式定义，以使它可被大、中、小型组织使用于任何类型的软件开发中。另外，此模型可适用于全组织范围内，从整个组织，或者甚至是一个单一的项目。除了这些特点，SAMM还建立在以下原则：

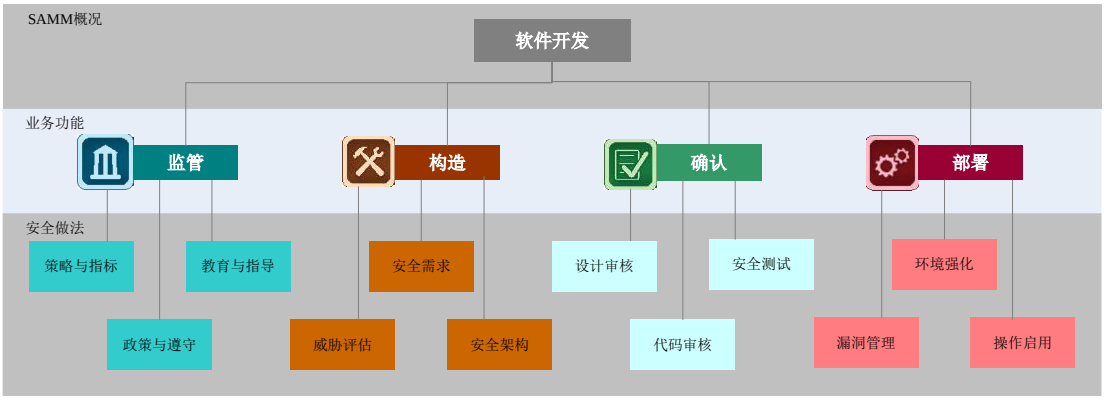
✦一个组织的行为随着时间的推移而缓慢的改变——一个成功的软件安全计划，应当详细说明每一个小的迭代步骤，以使能够提供有形的保证收益，并向项目的长期目标前进。

✦没有单一的方法可作用于所有的组织——一个软件安全框架必须是灵活的，并允许组织可以根据他们的风险承受能力和他们开发和使用软件的方式，去改进他们的选择。

✦与安全措施相关的指导必须是规范的——所有建立和评估保证计划的步骤应是简单的、明确的和可测量的。该模型还为普遍类型的组织提供了路线图模版。

该模型的基础建立于软件开发的每个核心业务职能上，每个职能都有相应的安全实践（见下图）。该模型为十二个安全实践中的每一个实践，都定义了三个成熟等级。这些定义了一个组织可以进行的各种实践，以降低安全风险并加强软件的保证。其他的细节还包括：衡量成功实践的性能表现、了解相关保证的益处、评估人员和其他成本。

作为一个开放的项目，SAMM的内容应始终保持普遍代表性，并对所有可用的资源免费提供使用。



目录

我需要…

评估现有软件的保证措施

3 ✦执行摘要
8-9 ✦业务功能
10-11✦监管
12-13✦构造
14-15✦验证
16-17✦部署
21-25✦评估执行
26 ✦创建记分卡
20 ✦使用成熟度等级
34-37✧策略与指标
38-41✧政策与遵守
42-45✧教育与指导
46-49✧威胁评估
50-53✧安全需求
54-57✧安全架构
58-61✧设计审核
62-65✧代码审核
66-69✧安全测试
70-73✧漏洞管理
74-77✧环境强化
78-81✧操作实现
27-31✧建立保证计划
84-95✧VirtualWare

建立一个组织的战略路线图

3 ✦执行摘要
8-9 ✦业务功能
10-11✦监管
12-13✦构造
14-15✦验证
16-17✦部署
20 ✦使用成熟度等级
27-31✦建立保证计划
21-25✧评估执行
26 ✧创建记分卡
84-95✧VirtualWare
34-37✧策略与指标
38-41✧政策与遵守
42-45✧教育与指导
46-49✧威胁评估
50-53✧安全需求
54-57✧安全架构
58-61✧设计审核
62-65✧代码审核
66-69✧安全测试
70-73✧漏洞管理
74-77✧环境强化
78-81✧操作实现

实施或执行安全活动

3 ✦执行摘要
8-9 ✦业务功能
10-11✦监管
12-13✦构造
14-15✦验证
16-17✦部署
20 ✦使用成熟度等级
34-37✧策略与指标
34-37✧策略与指标
38-41✧政策与遵守
42-45✧教育与指导
46-49✧威胁评估
50-53✧安全需求
54-57✧安全架构
58-61✧设计审核
62-65✧代码审核
66-69✧安全测试
70-73✧漏洞管理
74-77✧环境强化
21-25✧评估执行
26 ✧创建记分卡
27-31✧建立保证计划
84-95✧VirtualWare

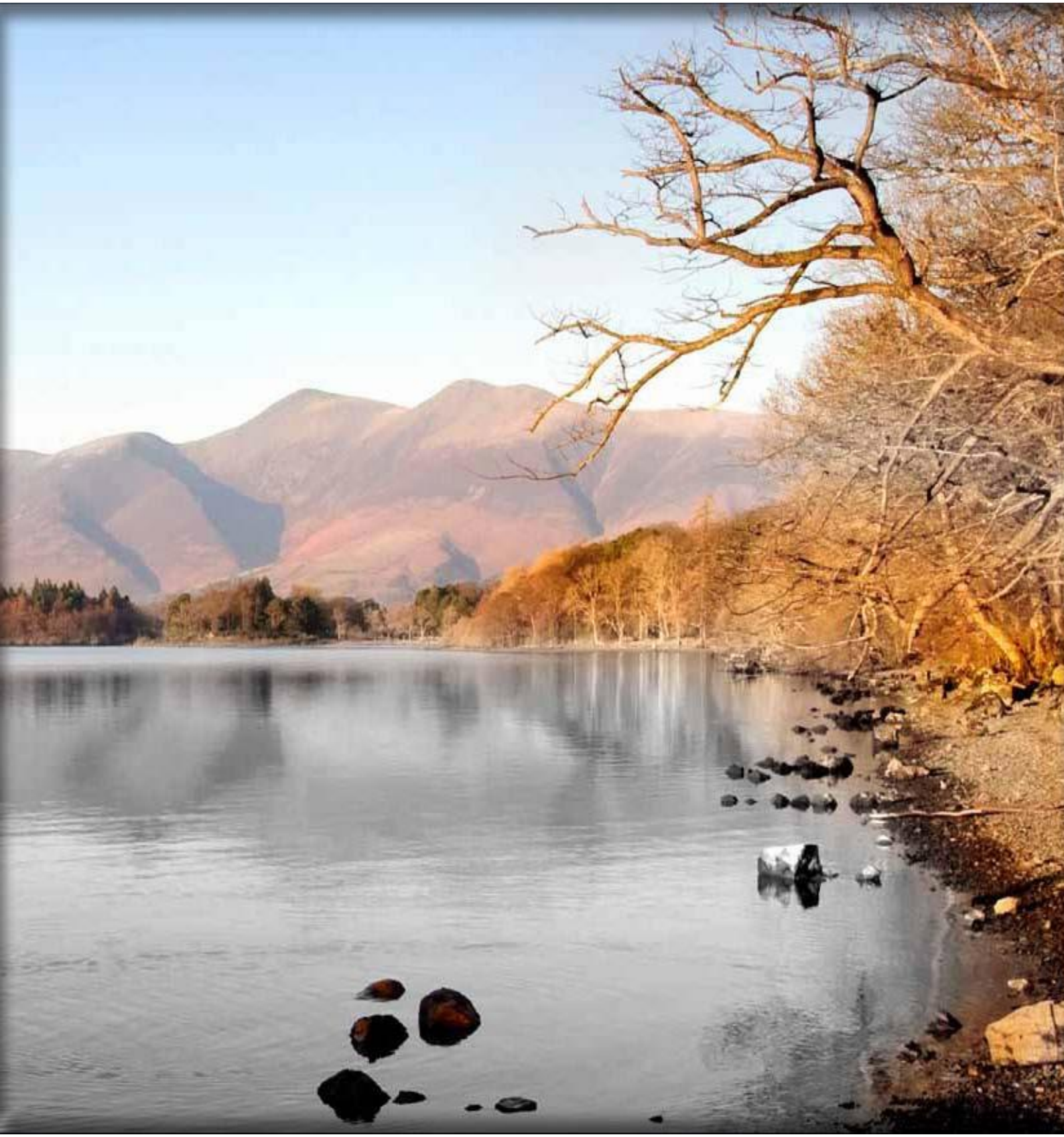
✦ 细读
✧ 略读


private $dbname;
private $charset;
public function connect()
$link = null;
mysqli_connect


Performance over 3 months

理解模型

从宏观的角度



SAMM建立在一个安全实践的集合上，这些安全实践关联到软件开发过程中的核心业务。本节介绍这些业务功能以及每个相应的安全实践。先在对框架进行总体介绍后，再对每个安全实践的成熟度等级进行简单地讨论，从而描绘出循序渐进提高每个成熟度等级的总体图。

业务功能

在最高等级上，SAMM设置了四种关键业务功能。每种业务功能（在下文中列出）是一组软件开发过程中具体细节的相关措施；换句话说，任何涉及了软件开发的组织，必须在一定程度上实现每一个业务功能。

对于每一个业务功能，SAMM设置了三个安全措施。每个安全措施（在下页中列出）是一个与安全相关的措施的领域，以为相关业务功能建立保证。所以，从总体来说，这十二个安全措施都是改进软件开发业务功能的独立部分。

对于每一个安全实践，SAMM设置了三个成熟度等级作为目标。安全实践中的每个等级，通过设置特定的活动和比先前等级更加严格的成功指标，设定了一个更加复杂的目标。此外，每个安全实践可被独立改善，虽然相关的措施可导致优化。

“监管”集中在于与一个组织如何管理所有软件开发活动相关的处理过程和措施。更具体地说，这包括关注多小组参与的开发过程，以及在组织级别上建立的业务处理过程。

更多信息请见第10页

“构造”关注于与一个组织对于开发项目中设置目标和创建软件相关的处理过程和措施。在一般情况下，这将包括产品管理、需求收集、高级别架构说明、详细设计和执行。

更多信息请见第12页

“确认”注重于与一个组织如何检查和测试软件开发过程中生产产物相关的处理过程和措施。这通常包括质量保证工作，比如测试，但它也可以包括其他审核和评估措施。

更多信息请见第14页

“部署”需要与一个组织如何管理所开发软件的发布相关的处理过程和措施。这涉及到将产品运送给终端用户、将产品部署在内部或外部主机，以及在运行环境中软件的正常运作。

更多信息请见第16页

监管	**策略与指标**包含了软件保证计划的总体战略方向和采集一个组织安全态势的测量处理过程和措施。	**政策与遵守**包含了建立一个贯穿一个组织的安全和遵守控制，以及审计框架，以实现正在开发和运行软件的安全保证。	**教育与指导**包含了通过培训以及指导与独立功能相关的安全议题，来增强软件开发过程中人员的安全知识。
构造	**威胁评估**包含了准确确定并特征化在一个组织的软件中潜在的攻击，以便更好地了解风险，并为风险管理带来便利。	**安全需求**包含了在软件开发过程中，为在开始阶段明确指定正确的功能，敦促将与安全相关的要求包含其中。	**安全架构**包含了促进默认安全设计的措施，和在软件开发过程中对所有技术和构架的控制，以支持设计过程。
确认	**设计审核**包含了考察设计过程中的产物以确保提供足够的安全机制，并坚持一个组织的安全期望。	**代码审核**包含了对一个组织源代码的评估，以帮助发现漏洞及相关的弥补措施，以及建立一个安全编码的最低期望值。	**安全测试**包含了在运行环境下对组织的软件测试，以发现漏洞，并为软件发布建立一个最低标准。
部署	**漏洞管理**包含了为管理内部和外部的漏洞报告建立一个一致的处理过程，以限制漏洞地对外暴露并采集数据以加强软件安全计划。	**环境强化**包含了围绕组织软件的操作环境，执行相关地控制，以加强对已部署应用程序的安全态势。	**操作实现**包含了识别并获得操作人员正确配置、部署和运行软件，而需要的与安全相关的信息。

成熟度等级

十二个安全实践中的每一个实践，都设置了三个成熟度等级和一个隐含的为零的起点。每个等级的细节在不同实践之间有所不同，但它们普遍代表：

0 隐起点代表在实践中的措施尚未实现

1 对安全实践有了初步了解并有所专门的提供

2 提高了安全实践的效率和（或）有效性

3 在一定规模上综合掌握了安全实践

注释

在整篇文档中，以下标注的术语被设定为在章节中设置的SAMM组件。如果这些术语没有被标注使用，它们应该在上下文的基础上被理解：

- **商务功能**也被称为**功能**
- **安全实践**也被称为**实践**
- **成熟度等级**也被称为**等级**、**目标**

监管

安全实践描述

策略与指标

策略与指标实践（SM）专注于为软件安全保证计划在组织中建立一个框架。这是以一种可衡量并和该组织实际业务风险相一致的方式，定义安全目标的最基本步骤。

从简便的风险概况出发，组织经过一段时间地工作，开发出针对应用程序和数据资产更先进的风险分类方法。通过对相关风险衡量的进一步了解，组织可以调整其项目级的安全目标和开发粒度路线图，使其安全计划更有效。

在这一实践的更高等级上，组织利用许多内部和外部的数据，去采集安全计划的度量指标和实质性的反馈信息。这允许在计划的级别上对成本支出和实现地受益进行调整。

政策与遵守

政策与遵守实践（PC）专注于理解并符合外部法律和规章制度的要求，同时也推动了内部安全标准，以确保遵守与该组织的业务目的一致。

为得到改善，该实践专注于项目级别的审计，收集有关该组织行为的信息，以检查期望是否达到。通过从引入简便的常规审计开始，随着时间逐步深入，组织以循序渐进的方式发生改变。

在一个复杂的形式下，提供这种实践需要理解全组织的内部标准和外部遵守驱动因素，并同时保证尽量按计划准时检查项目团队，以确保没有项目在没有被发现的情况下进行非期望的操作。

教育与指导

教育及指导实践（EG）专注于增加软件生命周期中涉及到人员的知识，以及设计、开发和部署安全软件需要使用的资源。有了更好的信息，项目团队将能够更好地主动识别并降低他们组织的特定安全风险。

为达到改善的目标，一个主要的方法是为员工提供培训：或是由专家指导的讲义课程，或是基于计算机的上机课程。作为组织的改善进展，一个广泛的培训由开发人员开始，并为组织中其他的角色提供培训，最后以基于角色的认证结束，从而确保了培训的综合性。

除了培训，该实践也需要使与安全相关的信息成为指导，以作为工作人员的参考信息。这个实践为建立组织安全实践的基准期望奠定了基础；并被允许一旦采纳该指导，就将逐步改善。

监管

措施概观

策略与指标

更多信息请见第34页

	SM1	SM2	SM3
目标	为组织内的软件安全建立统一的战略路线图。	衡量数据和软件资产的相对价值，并选择风险容忍度。	使安全成本与相关业务指标和资产价值相一致。
措施	A. 评估整体业务风险概况； B. 建立并维护保证计划路线图。	A. 根据业务风险将数据和应用程序分类； B. 建立并衡量每个分组的安全目的。	A. 引导周期性地全行业成本比较； B. 为以前的安全成本收集度量标准。

政策与遵守

更多信息请见第38页

	PC1	PC2	PC3
目标	了解组织的相关监管和遵守要求。	建立安全和遵守的基准线，并了解每个项目的风险。	要求遵守标准，并衡量项目是否符合全组织的政策和标准。
措施	A. 确定并监控外部的遵守驱动因素； B. 建立并维护遵守指导。	A. 为安全和遵守建立政策和标准； B. 建立项目审计实践。	A. 为项目建立遵守关卡； B. 为审计数据的采集，采用解决方案。

教育与指导

更多信息请见第42页

	EG1	EG2	EG3
目标	为开发人员提供关于以安全编程和部署为主题的资源。	为软件生命周期中所有的人员提供基于角色的安全开发详细指导。	实施综合的安全培训，并为员工进行基本知识的认证检验。
措施	A. 实施技术安全意识的培训； B. 建立并维护技术指导。	A. 实施针对特定角色的应用程序安全培训； B. 聘用安全指导专家增强项目团队。	A. 建立正式的应用程序安全支持门户网站； B. 建立基于角色的考试或认证制度。

构造

安全实践描述

威胁评估

威胁评估实践（TA）专注于在以开发软件的功能和运行环境的特点的基础上，识别并了解项目级别的风险。根据每个项目的威胁和可能遭受到的攻击的详细信息，组织通过对安全举措的优先次序，有效地做出更好的决策。另外，针对风险地认同做出更明智的决定，从而能更好地与业务相协同。

通过从简单的威胁模型开始，并建立更加详细的威胁分析和加权方法，组织随着时间的推移得到不断的改善。最终，复杂的组织将以一种把弥补方法和来自外部实体的已通过风险紧密结合的方式，维护这类信息。这就为将来潜在的项目，扩大了对安全问题的理解，同时对该组织目前抵抗已知威胁的表现保持密切关注。

安全需求

安全需求实践（SR）注重于积极阐述软件关于安全的预期行为。通过在项目层面的额外分析活动，根据软件的高级别业务目的，安全需求得到初步收集。作为一个组织的进步，更先进的技术得到了使用，例如，访问控制的规范说明用以发现在开发时没有发现的新安全需求。

在一种复杂的形式下，这种实践的提供也促使将组织的安全需求与供应商的关系联系起来，然后审计所有的项目以确保它们遵守了安全需求规范的期望。

安全架构

安全架构实践（SA）注重于在默认的情况下，组织采取积极措施设计并开发安全的软件。通过使用可重复使用的服务和组件，加强软件的设计过程，从而大大降低来自软件开发过程中的的整体安全风险。

从软件框架的简单建议和安全设计原则的明确考虑出发，组织开始为安全功能持续使用设计模式。此外，措施还鼓励项目团队对集中的安全服务和基础设施提高使用率。

随着组织的发展，这种复杂实践得使用使得组织建立了参照平台，以涵盖其建立的普遍类型软件。这些类型软件作为安全的框架，使开发人员可以以尽量低的漏洞风险开发软件。

构造

措施概观

威胁评估

更多信息请见第46页

	TA1	TA2	TA3
目标	确定并了解组织和单个项目的高级别威胁。	提高威胁评估的准确性，并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	A. 建立并维护特定应用程序的威胁模型； B. 根据软件架构建立攻击者概况。	A. 建立并维护每个项目的滥用用例模型； B. 为威胁的度量采用一个权重系统。	A. 明确评估来自第三方组件的风险； B. 用补偿控制详细描述威胁模型。

安全需求

更多信息请见第50页

	SR1	SR2	SR3
目标	在软件需求分析阶段明确地将安全考虑在内。	根据业务逻辑和已知风险增加安全需求的深度。	为所有软件项目和第三方的附属项目强制要求安全需求。
措施	A. 从业务功能推导出安全需求； B. 为需求评估安全和遵守指导。	A. 为资源和能力建立一个访问控制矩阵； B. 根据已知风险指定安全需求。	A. 将安全需求写入供应商协议中； B. 为安全需求扩展审计计划。

安全架构

更多信息请见第54页

	SA1	SA2	SA3
目标	将主动安全指导的想法引入到软件设计过程中。	将软件设计过程引导向已知安全服务和默认安全设计。	正式控制软件设计过程并验证安全部件的使用。
措施	A. 维护推荐的软件框架列表； B. 将安全原则明确运用到设计中。	A. 明确并促进安全服务和基础设施； B. 明确来自架构的安全设计模式。	A. 建立正式的参照架构和平台； B. 验证框架、模式和平台的使用。

验证

安全实践描述

设计审核

设计审核实践（DR）注重于针对与安全相关的问题评估软件设计和架构。这使得组织能够在软件开发的早期检测到架构层面的问题，从而避免因安全问题而导致以后由于重构所带来地潜在昂贵成本。

从简便的活动开始，了解有关架构的安全细节，组织进行了更加正式的检验方式，以验证安全机制提供的完整性。在组织层面上，设计审核服务建立并提供给利益相关者。

在一个复杂的形式下，该实践的提供涉及了数据级别的详细设计检验，以及期待基准线的实施。为在发布以前，执行设计评估和审核检验。

代码审核

代码审核实践（CR）注重于在源代码级别上的软件检验，以发现安全漏洞。代码级的漏洞一般在概念上简单易懂，但即便是开发人员也可以很容易地犯下错误，使软件存在潜在的安全漏洞而被利用。

首先，组织使用简便的检查列表，并针对有效性，仅检查软件的最关键模块。但是，组织使用自动化的技术，可以大大提高覆盖面和代码审核的效果。

该复杂实践的提供，将代码审核深深地整合到开发过程中，使项目团队能够及早地发现问题。这也使组织能够更好地进行审计，并在发布以前为代码审核结果设立期望。

安全测试

安全测试实践（ST）注重于在实时环境中检验软件，以发现安全问题。这些测试活动通过在与预期运行环境相同的情况下运行，检查软件的保证用例，从而避免那些不易发现的操作配置错误或业务逻辑中的错误再次发生。

从渗透测试和基于软件功能的高级别测试用例开始，组织使用自动化的安全测试方法以覆盖更多的测试用例，以发现系统中可能存在的漏洞。

在一个高级的形式下，该实践的提供涉及了自定义的自动化测试，以建立一个覆盖了详细的安全应用特定问题的测试机制。由于提供了在组织级别的其他可视性，安全测试使组织在发布以前能为安全测试结果设置一个最低预期。

验证

措施概观

	设计审核		更多信息请见第58页
	DR1	DR2	DR3
目标	为软件设计提供专门的审核，以确保排除已知风险的最低线。	根据安全的最佳实践为软件设计审核提供评估服务。	需求评估并验证已完成部分，以详细了解保护机制。
措施	A. 确定软件攻击层面； B. 根据已知安全需求分析设计。	A. 检查提供安全机制的完整性； B. 为项目团队部署设计审核服务。	A. 为敏感资源开发数据流图； B. 为设计审核建立发布关卡。

	代码审核		更多信息请见第62页
	CR1	CR2	CR3
目标	随机查找基本的代码级漏洞和其他高风险安全问题。	通过自动化方式在开发过程中使代码审核更加准确和有效。	必须进行全面的代码审核过程，以发现语言级别和特定应用程序的风险。
措施	A. 根据已知安全需求建立审核检查列表； B. 为高风险代码执行定点审核。	A. 使用自动化的代码分析工具； B. 将代码分析集成到开发流程当中。	A. 为特定应用程序问题自定义代码分析； B. 为代码审核建立发布关卡。

	安全测试		更多信息请见第66页
	ST1	ST2	ST3
目标	根据编程和软件需求，建立处理过程以执行基本的安全测试。	通过自动化使在开发过程中的安全测试更加完善和有效。	在部署前要求进行特定应用程序的安全测试以确保基本的安全。
措施	A. 从已知安全需求推出测试用例； B. 为软件发布执行渗透测试。	A. 使用自动化的安全测试工具； B. 将安全测试整合到开发过程中。	A. 为特定应用程序使用自动化的安全测试； B. 为安全测试建立发布关卡。

部署

安全实践描述

漏洞管理

漏洞管理实践（VM）注重于处理组织内漏洞报告和操作事故的处理过程。通过这些处理过程，组织的项目将会对这些事件的发生保有一致的预期，并不断提高效率，而不是混乱和无知的响应。

从在事件发生时对角色的简便分配开始，组织发展了一个更为正式的事件响应处理过程，以确保对发生事件的可见性，并能对其跟踪。交流也有所改善，以提高处理过程的整体理解。

在一个高级的形式下，漏洞管理包括了更加深入的事件分析和漏洞报告，以收集到详细的指标和其他根本的原因信息，用以反馈到组织的下游行为中。

环境强化

环境硬化实践（EH）注重于为组织软件运行的环境建立保证。由于应用程序的安全运行操作可因外部组件的问题而受到损坏，强化基础设施可直接提高软件的整体安全状况。

开发团队通过简单的跟踪和分部，更好地了解关于操作环境的信息。另外，组织通过使用可测量的方法，管理安全补丁的部署，并在操作环境中安装检测器，为潜在的安全事件造成破坏以前产生早期警报。

作为组织的进步，通过部署保护工具，以添加防御层和安全网来减少当任意漏洞被利用时所造成的破坏和损失。因此，操作环境得到了进一步的审核和强化。

操作实现

操作实现实践（OE）注重于从项目团队正在开发的软件收集关键的安全信息，并将这些信息传递给软件的用户和操作人员。如果没有这些信息，即使是设计最安全的软件，也会因为无法知道部署地点的重要安全特性和选择，而包含不必要的风险。

从为用户和操作人员编写简便的文档以获得最有效的细节信息开始，组织逐渐为每个版本的发布编写完整的操作安全指南。

在一个高级的形式下，操作实现也需要对每个项目团队进行组织级别的检查，以确保根据预期获得信息并共享。

部署

措施概观

漏洞管理

更多信息请见第70页

	VM1	VM2	VM3
目标	理解对于漏洞报告或事件的高级别计划。	为响应过程阐述期望，以改善一致性和交流。	在响应过程中为积极规划提供反馈，而改善分析和数据收集。
措施	A. 为安全事件确定联络点； B. 建立非正式安全响应团队。	A. 建立一致的事件响应流程； B. 采用安全事件报告流程。	A. 为事件执行根源分析； B. 收集每一事件的度量指标。

环境强化

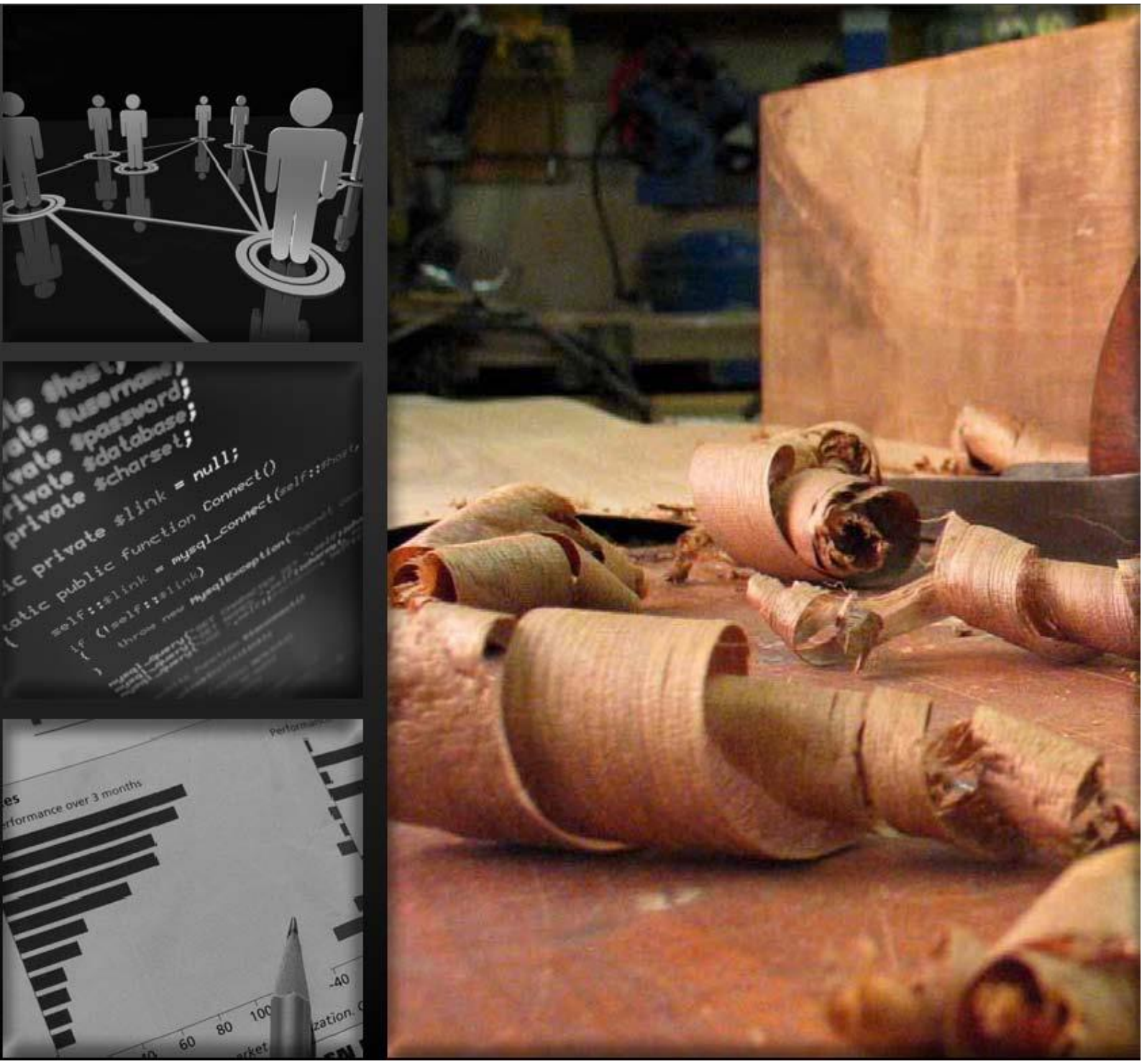
更多信息请见第74页

	EH1	EH2	EH3
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。	以已知最佳实践验证应用程序的健康和操作环境状态。
措施	A. 维护操作环境说明； B. 确定并安装关键的安全软件升级和补丁。	A. 建立常规补丁管理流程； B. 监控基准基础架构配置状态。	A. 确定并部署相关操作的保护工具； B. 为环境配置扩展审计计划。

操作实现

更多信息请见第78页

	OE1	OE2	OE3
目标	实现开发团队和操作人员之间对于与安全相关的关键数据的沟通交流。	通过提供详细的步骤为持续的安全操作提高期望。	针对完整性而对安全信息和部件检验进行强制宣传。
措施	A. 为部署获得的重要的安全信息； B. 为典型的应用程序警报记录流程。	A. 创建每次发布的变更管理流程； B. 维护正式的操作安全指南。	A. 为操作信息扩展操作审计计划； B. 对应用程序组件执行代码签名。



应用模型

使用方法

本节涵盖了 SAMM 的几个重要的实际应用。由于提供了模型本身的核心设计，组织可以将 SAMM 作为基准来衡量其安全保证计划，并创建记分卡。通过使用记分卡，组织能够证明通过迭代执行一项保证计划得到循序渐进改善。而最重要的是，组织还可以使用 SAMM 路线图模板，以指导建立或改善一个安全保证计划。

使用成熟度等级

十二个安全实践中的每一个实践都有三个成熟度等级。每个等级都有几个为理解并实现相关等级而详细说明关键因素的部分。除此之外，这些详细规定的细节内容使用了安全实践的定义，甚至可以在超出 SAMM 的使用范围情况下，建立一个软件保证计划。

目标

目标是一段总体说明，以陈述实现相关等级的保证目的。随着一个给定实践等级得不断提高，就为软件开发和部署的保证建立而言，目标也被设定得更加复杂。

措施

措施是实现等级的核心条件。有些措施是在全组织范围内进行的，有些措施则对应了个别项目团队的行动。在任何情况下，措施都获得了关键的安全功能，并且组织也可以自由得决定如何执行这些措施。

结果

结果体现了能力，并通过实现给定的等级来提交获得的结果。在某些情况下，结果被详细说明；而在其他情况下，为体现能力得提高，做出更多的定性声明。

成功指标

成功指标指明了衡量的样本，用以检查组织是否执行在给定的等级。组织可以选择是否执行数据的自我采集和自我管理，但是必须采用相应推荐的数据和阈值。

成本

成本是关于由组织为达到给定的等级而产生开销的定性说明。虽然具体的数值会因每个组织而不同，但这些成本数据只是参考意见，以说明为一个特定等级所进行的操作而带来的成本开销。

人员

人员表示在给定的级别上，从人力资源的角度来估算持续的开销。

✦开发人员——负责软件的详细设计和编码的人员；
✦架构师——负责高级别的设计工作和大型系统工程技术的人员；
✦经理——负责日常管理开发人员的人员；
✦QA 测试员——负责软件质量保证测试和发布前检验的人员；
✦安全审计员——具备软件技术安全知识的人员；
✦业务拥有者——负责对软件及其业务需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。

相关等级

相关等级是根据其组织的结构和正在建立保证计划的进度，而决定的一些潜在重叠的实践等级的参考。从功能上看，如果有关等级也是一个目标或已经被满足，那么这些等级都指明了实践应用的协同效应或者应被优化实施。

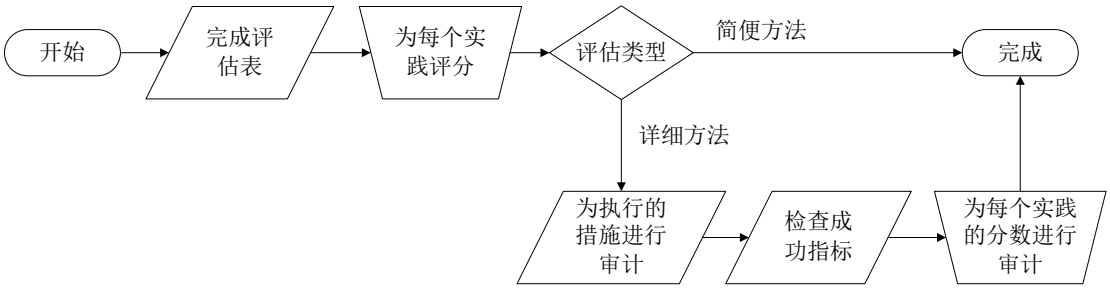
评估执行

按照已定义的安全实践来衡量一个组织，并建立其安全保证活动的整体状况。对于了解目前组织内已设置的安全活动的广度，这类评估非常有用。此外，它还使该组织能使用 SAMM 创建一个未来的发展路线，并以循序渐进的方式逐步得到改善。

一项评估过程的执行，仅仅是确定组织目前正在执行情况的成熟度等级。根据评估的原动力，通常会扩展对组织表现检验的方法。但是，有两种推荐的方法：

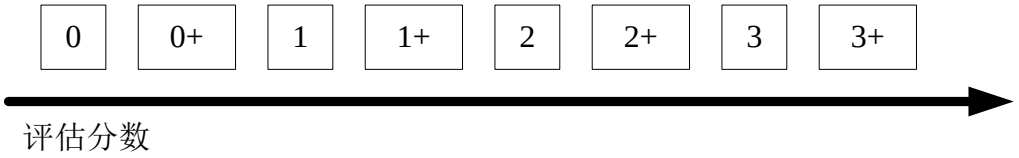
✦简便方法——每项实践的评估表都被评估分析，并为得到的回答评定分数。这种评估方法通常对于一个正尝试将已有的保证计划与 SAMM 相匹配，以及仅希望能立马了解其已达到保证计划等级大概情况的组织，是足够的。

✦详细方法——完成评估表以后，执行额外的审计工作，以确保该组织每个实践中规定的每一项措施都已执行。此外，因为每一个实践也规定了成功指标，相应的数据应被收集，以确保该组织正在按照预期那样执行。



使用评估表给组织打分是很直接的。在回答完问题后，评估答案栏中的分数以确定相关等级。所有问题的回答需为肯定的回答，并填写在相应实践标号的上方和问题栏的右方。

已有保证计划的组成措施，有可能并非总处于成熟度等级的边界值，比如：一个组织将一个给定的实践评估为等级 1,但可能实施了超出等级 1 的额外措施，且并没有达到等级 2。对于这种情况，组织的得分应注明一个“＋”号，以表示额外的保证措施在该等级已获得。例如，一个组织为“操作实现”实践，执行了所有等级 1 的措施，与此同时，一个等级 2 或等级 3 的措施的执行，则为等级 1 赋予了“1＋”的得分。同样，一个组织为一个“安全实践”实践执行了所有的措施，包括了一些超出 SAMM 范围的措施，因此得到一个“3＋”的得分。



监管

评估表

策略与指标	是/不是	
✦是否已经有了一个软件安全保证计划？		SM1
✦大多数业务拥有者是否了解组织的风险概况？		
✦大多数开发人员是否对保证项目的未来计划有所意识？		
✦大多数应用程序和资源是否根据风险被分类？		SM2
✦风险评级是否被用于调整需求的保证措施？		
✦组织是否知道基于风险评级需要什么？		
✦是否采集了每个项目保证措施成本的数据？		SM3
✦你的组织是否经常和其他组织比较安全成本？		

政策与遵守	是/不是	
✦是否大多数的项目拥有者知道他们项目的政策和标准的遵守状态？		PC1
✦遵守需求是否被项目团队特别考虑到？		
✦组织是否使用了一套政策和标准来控制软件开发？		PC2
✦项目团队是否可以对政策和标准的遵守要求审计？		
✦项目是否定期审计以确保政策和标准的遵守底线？		PC3
✦组织是否系统性得使用审计以采集并控制遵守的证据？		

教育与指导	是/不是	
✦大多数的开发人员是否接受了高级别的安全意识培训？		EG1
✦每个项目团队是否都接触了安全开发的最佳实例和指导？		
✦开发过程中的大多数角色是否受到了根据角色的培训和指导？		EG2
✦大多数的利益相关者是否能够在项目中安排安全指导员？		
✦与安全相关的指导是否由组织集中控制并在组织中一致分部？		EG3
✦大多数的人员是否为安全开发进行了测试以保证达到要求的最低技能底线？		

构造

评估表

威胁评估	是/不是	
✦组织中的大多数项目是否考虑并记录可能的威胁？		TA1
✦组织是否了解并记录可能面对的攻击者类型？		
✦项目团队是否定期为可能的滥用分析功能需求？		TA2
✦项目团队是否为相关比较使用了一个威胁等级评定方法？		
✦利益相关者是否意识到相关的威胁和等级评定？		
✦项目团队是否具体考虑了来自外部软件的风险？		TA3
✦所有的保护机制和控制是否能捕获并匹配威胁？		

安全需求	是/不是	
✦大多数项目团队在开发过程中是否明确阐述的一些安全需求？		SR1
✦项目团队是否从最佳实例和遵守的指导中引导需求？		
✦大多数的业务拥有者是否审核相关项目的访问控制矩阵？		SR2
✦项目团队是否根据来自于其他安全活动的反馈来详细阐述需求？		
✦大多数的业务拥有者是否为安全需求审核厂商的协议？		SR3
✦项目团队描述的安全需求是否被审计？		

安全架构	是/不是	
✦项目团队是否提供一个推荐的第三方部件列表？		SA1
✦大多数的项目团队是否意识到安全设计原则并应用它们？		
✦您是否为项目团队宣传可共享的安全服务？		SA2
✦项目团队是否提供了基于应用程序架构的规范设计模式？		
✦项目团队是否从中央控制平台和框架建立软件？		SA3
✦项目团队是否为安全架构部分的使用进行审计？		

确认

评估表

设计审核	是/不是	
✦项目团队是否记录软件设计的攻击范围？		DR1
✦项目团队是否按照已知的安全风险检查软件设计？		
✦大多数项目团队是否为安全机制具体分析设计元素？		DR2
✦大多数项目业务拥有者是否知道如何获得一个正式的设计审核？		
✦设计审核过程中是否包括详细的数据级分析？		DR3
✦常规项目审计是否需要一个设计审核结果的最低线？		

代码审核	是/不是	
✦大多数的项目团队是否拥有基于普遍问题的审核清单列表？		CR1
✦项目团队是否通常针对选择的高风险代码执行审核？		
✦大多数项目团队是否可以使用自动的代码分析工具去查找安全问题？		CR2
✦大多数项目业务拥有者是否持续要求并审核来自代码审核的结果？		
✦项目团队是否根据应用程序特定的编码标准，使用自动化的方式去检查代码？		CR3
✦常规项目审核是否需要一个为发布代码审核结果的最低线？		

安全测试	是/不是	
✦项目是否明确指定一些基于需求的安全测试？		ST1
✦大多数项目是否在发布以前执行渗透测试？		
✦大多数利益相关者是否意识到需在发布以前进行安全测试？		
✦项目是否使用了自动化工具去评估安全测试用例？		ST2
✦大多数项目是否遵守一个一致的过程，为利益相关者评估并记录安全测试？		
✦安全测试用例是否为应用程序特定的逻辑而综合制作的？		ST3
✦常规项目审核是否需要来自于安全测试的最低标准结果？		

部署

评估表

漏洞管理	是/不是	
✦大多数项目是否都有对于安全问题的联络点?		VM1
✦组织是否已设立一个安全响应团队？		
✦大多数项目团队是否知道他们的安全联络点和响应团队?		
✦组织对于事件的报告和处理是否使用了一种一致的流程？		VM2
✦大多数利益相关者是否意识到公开的相关安全事件与他们的软件项目有关系?		
✦大多数调查了根本起因的安全事件是否有了更多更深入的建议？		VM3
✦大多数项目是否持续收集并报告关于安全事件的数据和衡量标准?		

环境强化	是/不是	
✦大多数项目是否为操作环境记录需求？		EH1
✦大多数项目是否检查第三方软件部件的安全更新？		
✦是否使用一个持续连贯的处理流程以对于关键依赖的部件使用安全更新和补丁？		EH2
✦大多数项目是否权衡了应用程序的自动化检测和环境的健康状态？		
✦利益相关方是否意识到当进行操作时用于保护软件的额外工具的选择？		EH3
✦常规审计是否为大多数的项目检查环境健康状况？		

操作实现	是/不是	
✦你是否递交了关于多数软件发布的安全记录？		OE1
✦大多数项目是否都记录了与安全相关的警报和产生错误的条件？		
✦大多数项目是否正在使用一个已理解的变更管理处理流程？		OE2
✦项目团队是否为每个软件发布递交了一份操作安全手册？		
✦大多数项目是否为了恰当的操作安全信息，而去审计检查每个发布？		OE3
✦软件组件的代码签名是否按照一致的处理流程被经常进行？		

创建记分卡

根据为给每个安全实践填写的分数，一个组织可以创建一个记分卡来记录这些分数值。从功能而言，记分卡可以是某个特定的时间记录的12组分数。但是，通过每过一个时间间隔产生了一个记分卡，有利于了解在时间框架内，保证计划的总体变化。

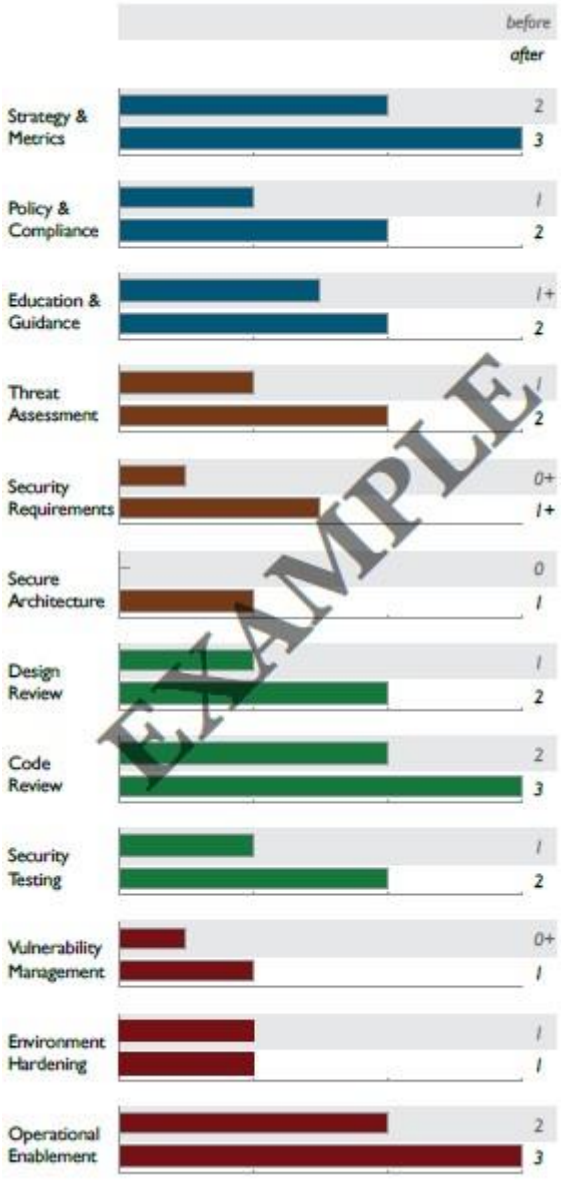
在以下几种情况中，建议使用时间间隔形式的记分卡：

✦差距分析——将获得的详细评估结果与预期的性能等级做比较，获得分数。

✦改善证明——在一次安全计划迭代建立完成的前后，获得分数。

✦持续衡量——为一个已经就位的保证计划，在持续的时间框架内获得分数。

右侧的图是一个记分卡样本，展现了一个组织的保证计划在一年中是如何改变的。如果该组织还保存了他们预期将在一年结束后获得的数据，那将是另一组有趣的数据，因为这将有助于显示在一年中计划改变的程度。



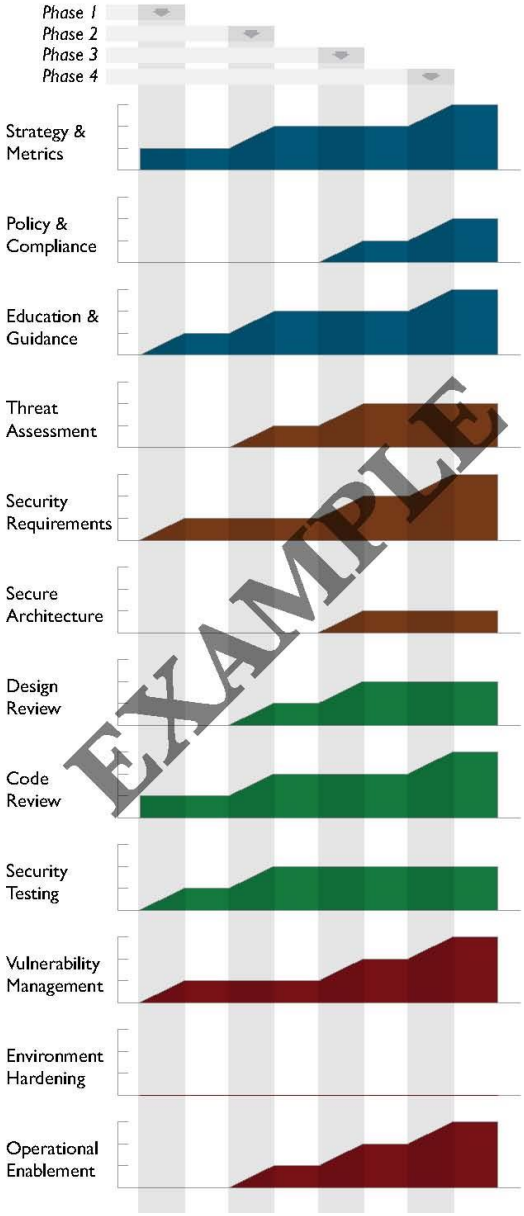
建立保证计划

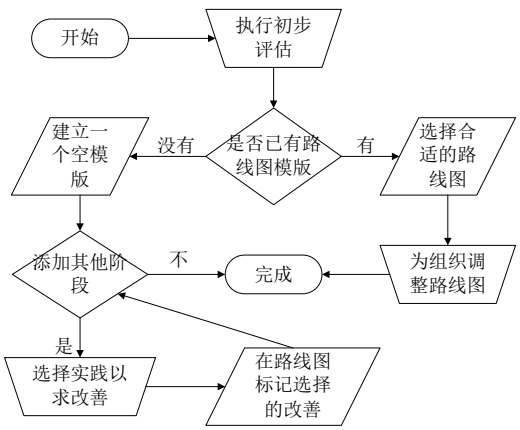
SAMM 的主要用法之一是帮助组织建立软件安全保证计划。这个过程很直接，通常，如果组织已经执行了一些安全保证措施，那么由评估开始。

我们为普遍类型的组织提供了几种路线图模板。因此，许多组织可以选择一个相匹配的合适模版，然后适当调整路线图模板以满足他们的需求。对于其他类型的组织，则可能需要建立一个自定义的路线图。

路线图（右图）由不同的阶段（垂直线条）组成，其中实践随着等级的提高而改善。因此，建立一个路线图，允许选择哪些实践在每个计划的阶段得到改善。组织可以根据他们的意愿自由地定制将来的计划，但是，鼓励根据业务驱动和组织特定的信息来循序执行，以确保保证计划的目标与组织的业务目标和风险承受能力相适应。

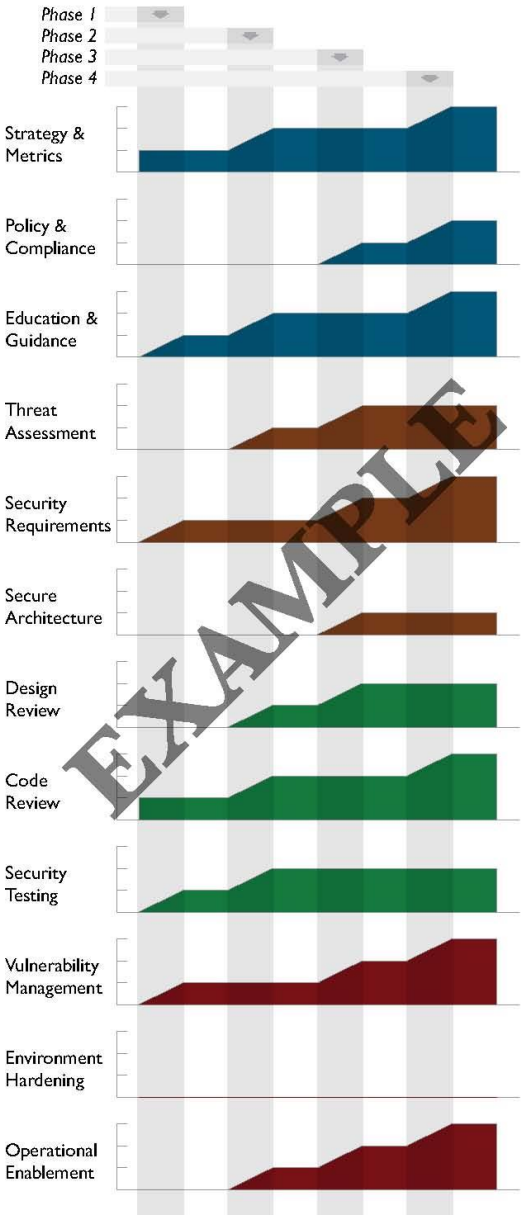
路线图建立以后，执行保证计划则很简单。一个组织从改善阶段开始，通过执行定制的措施来达到预期的等级。在该阶段结束时，应根据完成的实际情况调整路线图，然后开始下一阶段。





独立软件供应商

路线图模版



关系

独立软件供应商的核心业务包括了构建及销售软件组件和应用程序。

由于最初目的是限制会影响客户和用户业务的常见漏洞，因而早期行动专注于“代码审核”和“安全测试”的措施。

为了在产品说明中转向为积极主动地避免安全错误，组织会逐渐加强针对“安全需求”的措施。

此外，为了尽量降低已发现安全问题所造成的影响，组织还会逐渐加强“漏洞管理”措施。

随着组织逐渐成熟，来自“操作实现”的措施被逐渐增加，以更好地为客户和用户提供软件安全操作的信息。

额外的考虑

外包式开发

对于采用外部开发资源的组织，由于无法直接接触代码，因此，“安全需求”对于“代码审核”有着更为重要的优先级。此外，在早期阶段的高级“威胁评估”，可以允许组织能够更好的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团队中最有优势的，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

建立使用在线资源的应用程序的组织，存在一些额外的风险。这些风险源自于运行位于互联网基础设施构架中，面向互联网的系统。考虑到这方面的风险，组织应该将“环境强化”措施添加到他们的路线图中。

驱动程序与嵌入式开发

对于为嵌入式系统创建低层的驱动程序或软件的组织而言，在软件设计中的安全漏洞更有具破坏性，并且修复成本更高。因此，路线图必须被修改，以强调在早期阶段执行的“安全架构”和“设计审核”的措施。

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，而其与安全相关措施的等级也不同。对于这类组织，如果正在开发很多不同类型的软件，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关注点不同这一问题。

在线服务提供商

路线图模版

关系

在线服务提供商的核心业务包括创建网页应用程序和其他网络可访问的界面。

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，因而早期措施专注于“设计审核”和“安全测试”。

由于关键的系统将面向网络，还应尽早执行“环境强化”措施，并逐渐解决托管环境中存在的风险。

尽管组织的核心业务有所不同，但“政策与遵守”措施应当尽早启动，然后根据外部遵守驱动因素的关键性进行推进。

随着组织的逐渐成熟，“威胁评估”“安全需求”和“安全架构”措施逐渐添加，以便在基准安全预期建立以后支持主动的安全防御。

额外的考虑

外包式开发

对于采用外部开发资源的组织，由于无法直接接触代码，因此“安全需求”措施对于“代码审核”有着更为重要的优先级。此外，在早期阶段的高级“威胁评估”，可以允许组织能够更好的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有优势的，因此必须制定合同以说明与“操作实现”相关的措施。

在线支付处理

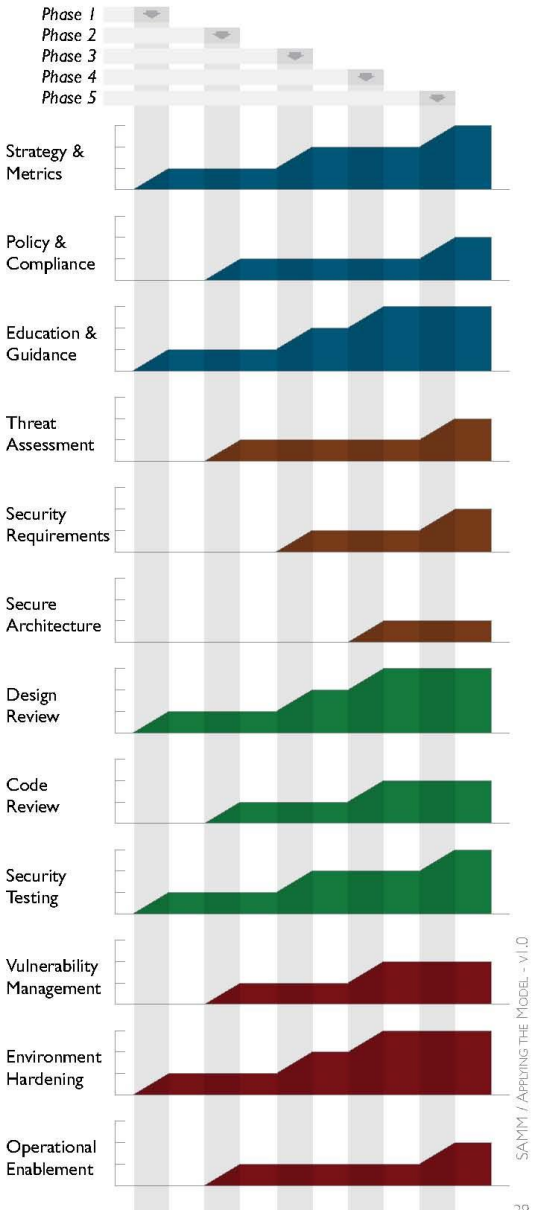
被要求遵守支付卡行业数据安全标准（PCI-DSS）或者其他在线支付标准的组织，应当在路线图的早期放入“政策与遵守”措施。这使得组织能够借机制定相关措施，以保证在将来的路线图中得到遵守和实现。

网页服务平台

对于正在构建网页服务平台的组织，设计方面的错误将会带来额外的风险，并且修复的成本更高。因此，应该将“威胁评估”“安全需求”和“安全架构”等措施排放在路线图的早期阶段。

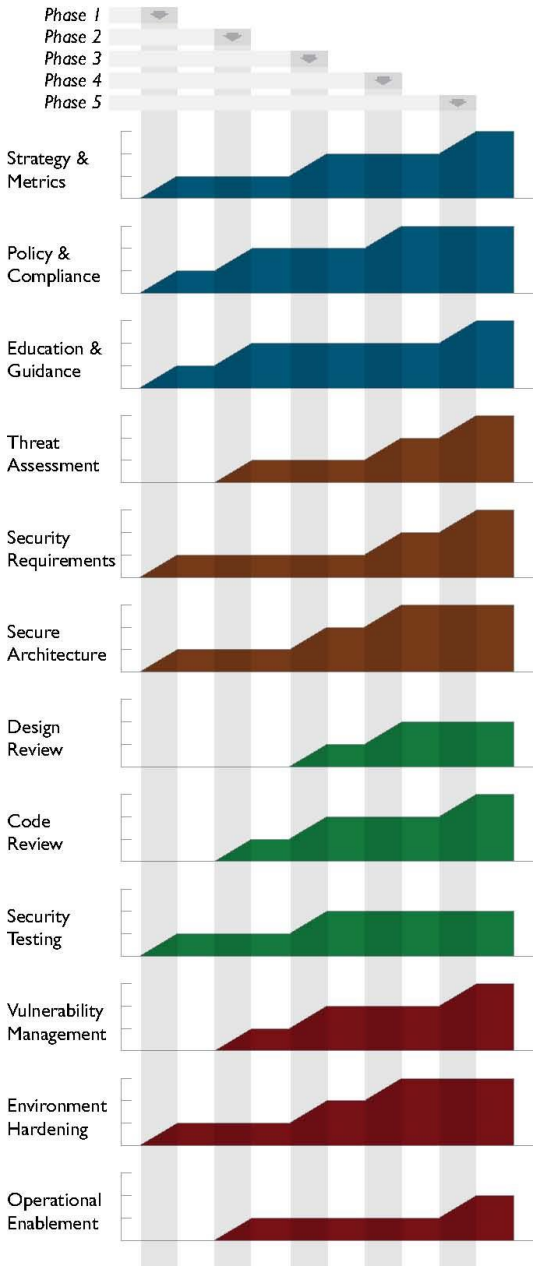
通过收购而发展壮大的组织

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，而其与安全相关措施的等级也不同。对于这类组织，如果正在开发很多不同类型的软件，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关注点不同这一问题。



金融服务机构

路线图模版



关系

金融服务机构的核心业务包括建立系统以支持金融交易和处理。通常情况下，这意味着一个非常集中的内部后端系统与外部不同的数据提供方相链接。

在初始阶段，工作的重点是改善那些与“监管”相关的实践，因为这些都是为保证计划设置基准和帮助组织到达遵守标准的关键服务。

因为主动建立安全且可靠的软件是一个总目标，在“构造”中的实践应尽早实施并能快速促使计划得成熟。

验证措施也同样能促使路线图计划的稳定执行，以处理现有系统不会创建不切实际的目标预期。此外，这还有助于确保使用了足够的循环来建立更为积极的实践。

由于金融服务机构往往操作自己开发的软件，那么重点则是在路线图的中期，当一些初始的“监管”实践完成以后，且在花更多精力在“构架”的实践以前，关注于“部署”中的实践。

额外的考虑

外包式开发

对于采用外部开发资源的组织，由于无法直接接触代码，因此“安全需求”对于“代码审核”有着更为重要的优先级。此外，在早期阶段的高级“威胁评估”，可以允许组织能够更好的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有优势的，因此必须制定合同以说明与“操作实现”相关的措施。

网页服务平台

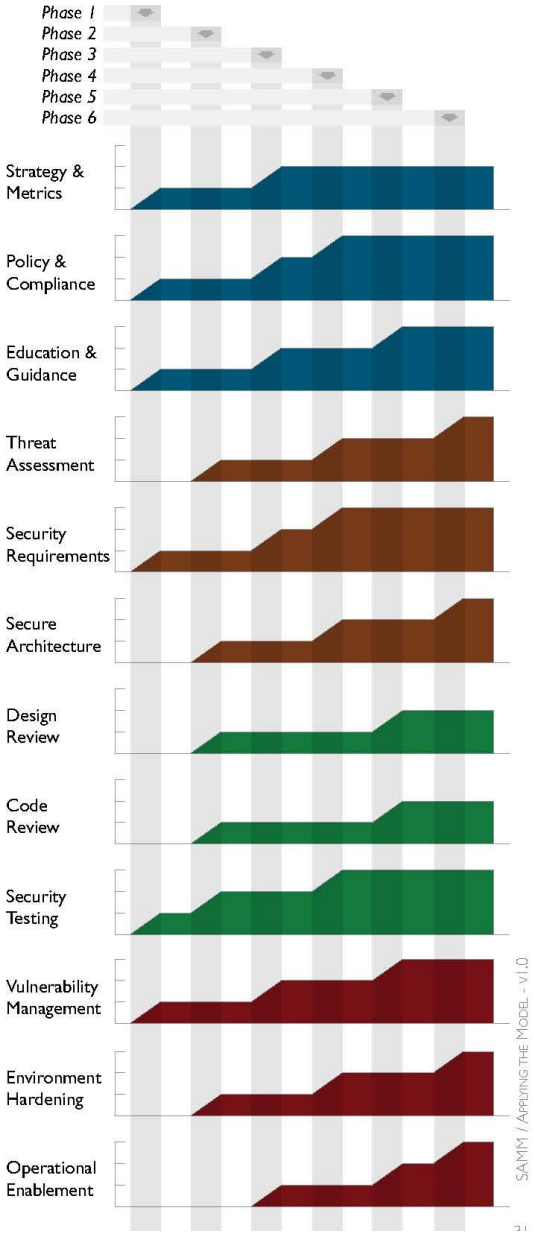
对于正在构建网页服务平台的组织，设计方面的错误将会带来额外的风险，并且修复的成本更高。因此，应该将“威胁评估”“安全需求”和“安全架构”等措施排放在路线图的早期阶段。

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，而其与安全相关措施的等级也不同。对于这类组织，如果正在开发很多不同类型的软件，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关注点不同这一问题。

政府组织

路线图模版



关系

政府组织的核心业务功能涉及到国家的附属机构，建立软件以支持公共部门的项目。

在初始阶段，“监管”实践被建立，并得到一个为组织改善具体路线图的遵守压力的大概想法。

由于公众曝光的风险以及大量已有代码的部署，在“验证”实践的早期强调“安全测试”，并在后期大量发展“代码审核”或者“设计审核”实践。

类似的重点同样放在“构造”和“部署”实践中。这有助于建立组织的漏洞管理，并有助于推动操作环境的安全姿态。与此同时，“构造”中的主动安全措施被建立，以帮助在软件开发过程中防止发生新的问题。

额外的考虑

外包式开发

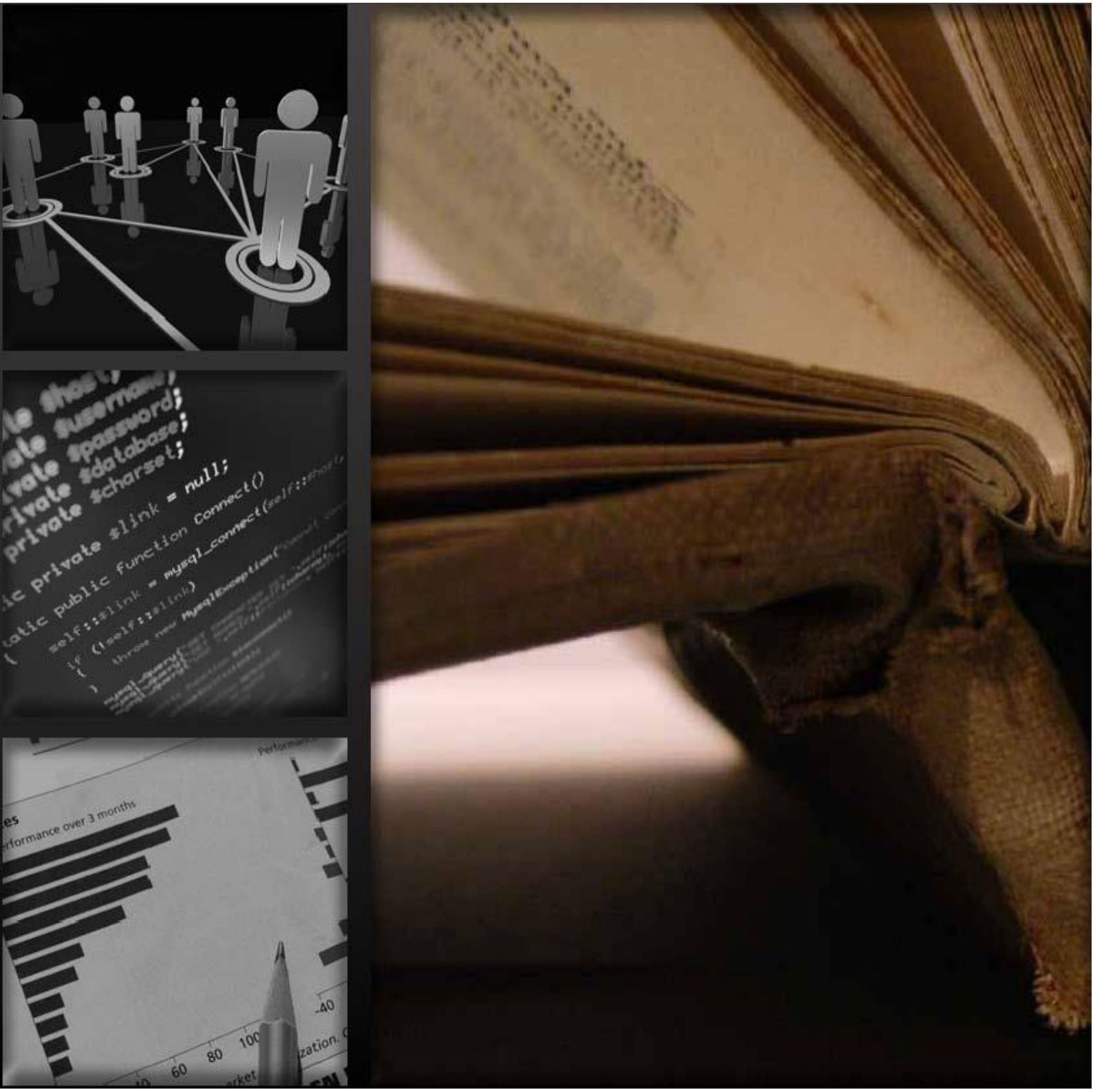
对于采用外部开发资源的组织，由于无法直接接触代码，因此“安全需求”对于“代码审核”有着更为重要的优先级。此外，在早期阶段的高级“威胁评估”，可以允许组织能够更好的向外包开发人员提出更明确的安全要求。由于有关软件配置的专业技术通常是外包组中最有优势的，因此必须制定合同以说明与“操作实现”相关的措施。

网页服务平台

对于正在构建网页服务平台的组织，设计方面的错误将会带来额外的风险，并且修复的成本更高。因此，应该将“威胁评估”“安全需求”和“安全架构”等措施排放在路线图的早期阶段。

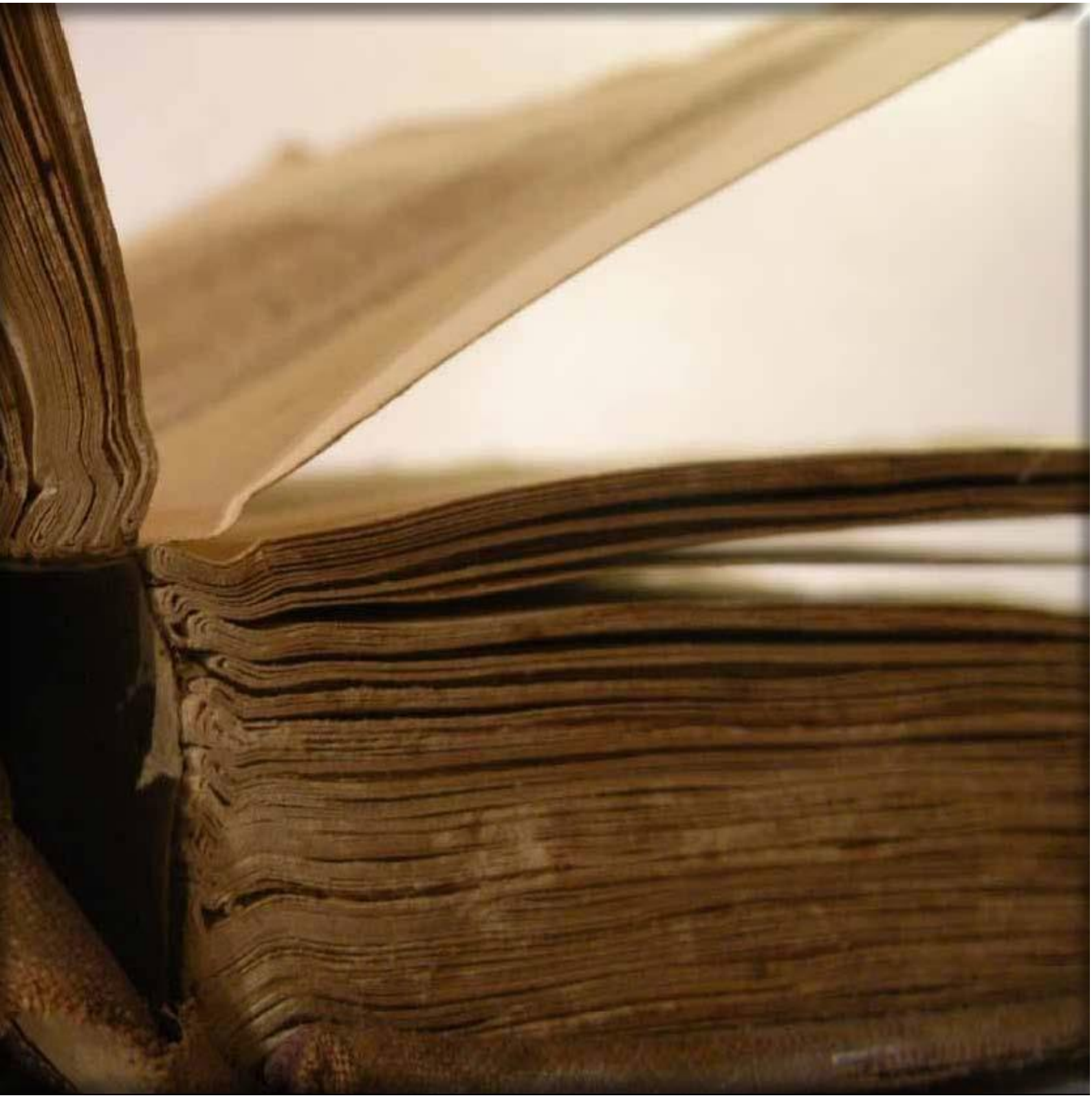
规章制度遵守

对于拥有大量需受规章制度限制业务的组织，建立“政策与遵守”的实践应加以调整，以适应外部驱动因素。同样，对于仅有少数遵守压力的组织，应抓住机会建立实践，以满足不同的需要。


private $host;
private $username;
private $password;
private $database;
private $charset;
public function connect()
$link = null;
mysql_connect(self::
self::$link
throw new MysqlException(
performance over 3 months

安全实践

详细说明



本节描述了 SAMM 中的细节：每个安全实践中的成熟度等级。对于每一个实践，汇总表包含了相关的三个等级。随后，每个等级的描述，包括详细地解释了所需的措施、组织从预期达到的等级获得的结果、衡量性能的成功指标、所需的持续人员投资以及其他相关的开销。

策略与指标

	SM1	SM2	SM3
目标	为组织内的软件安全建立统一的战略路线图。	衡量数据和软件资产的相对价值，并选择风险容忍度。	使安全成本与相关业务指标和资产价值相一致。
措施	A. 评估整体业务风险概况； B. 建立并维护保证计划路线图。	A. 根据业务风险将数据和应用程序分类； B. 建立并衡量每个分组的安全目的。	A. 引导周期性地全行业成本比较； B. 为以前的安全成本收集度量标准。
评估	✦是否已经有了一个软件安全保证计划？ ✦大多数业务拥有者是否了解组织的风险概况？ ✦大多数开发人员是否对保证项目的未来计划有所意识？	✦大多数应用程序和资源是否根据风险被分类？ ✦风险评级是否被用于调整需求的保证措施？ ✦组织是否知道基于风险评级需要什么？	✦是否采集了每个项目保证措施成本的数据？ ✦你的组织是否经常和其他组织比较安全成本？
结果	✦具体列举出了由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并有最小开销的路线图计划； ✦整个组织都了解了保证计划将会如何逐步进展。	✦根据业务的核心价值，为每个项目定制保证计划； ✦整个组织都了解到了与安全相关的数据和应用程序资产； ✦使利益相关者更好地了解并认同风险。	✦有助于制定每个案例安全支出决策的信息； ✦估算由于安全问题而在过去造成的损失； ✦考虑每个项目的安全支出与潜在的损失； ✦整个行业对于安全的勤奋付出。

策略与指标 SM1

为组织内的软件安全建立统一的战略路线图

措施

A. 评估整体业务风险概况

与业务拥有者和利益相关者进行面谈，并建立一个包括了组织各种应用程序和数据资产的最坏案例情况列表。根据组织开发、使用或销售软件的方式，最坏案例情况列表会有很大的不同。但常见的问题包括：数据盗窃或崩溃、服务中断、资金损失、逆向工程、帐户被盗等。

当获得了尽量多的最坏案例情况后，根据所收集到与核心业务相关的信息和知识，核对并选择最重要的情况。虽然可以选择任何数量的情况，但选择目标的范围应在3到7个，以便能更有效地利用时间，并保持工作的重点。

向组织详细阐释所选择的每一个选项，并记录有关最坏案例的情况、潜在导致发生的因素以及潜在牵制要素的详情。

最终的组织风险概况应由组织拥有者和其他利益相关者进行审核，以确保他们了解了这些内容。

B. 建立并维护保证计划路线图

为了了解组织的主要业务风险，针对十二项“实践”中的每一项，评估组织目前的状况。如果组织通过了所有累加的成功指标，那么可以根据相应的目标为每项“实践”以1、2、3这样的分数进行打分。如果没有达到成功指标，则为该实践打0分。

一旦完全了解了目前的状态后，下一个目标就是确定下一循环中需要改善的“实践”了。根据业务风险概况、其他业务驱动因素、遵守需求、预算容限等方面选择实践。一旦实践选择以后，重复操作的目标就是达到每个实践的“目的”。

在保证计划的改善方面，需要3到6个月的迭代操作，但应当至少每3个月召开一次保证策略会议，以审核计划的措施、参照成功指标的状况、以及可能要求更改计划改变的其他业务驱动因素。

结果

- ✦具体列举出了由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并有最小开销的路线图计划；
- ✦整个组织都了解了保证计划将会如何逐步进展。

成功指标

- ✦≥80％的利益相关者在过去6个月内都大概了解了业务风险概况；
- ✦≥80％的员工在过去3个月内大概了解了保证计划路线图；
- ✦≥1次以上的保证计划战略会议在过去3个月内被召开。

成本

- ✦扩充并维护组织风险概况；
- ✦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）
- ✦经理（4天/年）
- ✦业务拥有者（4天/年）
- ✦QA保证测试员（1天/年）
- ✦安全审计人员（4天/年）

相关等级

- ✦策略与指标－1
- ✦威胁评估－1
- ✦安全需求－2

策略与指标 SM2

衡量数据和软件资产的相对价值，并选择风险容忍度

措施

A. 根据业务风险将数据和应用程序分类

为应用程序建立一个简单的分类系统来表示风险等级。最简单的形式可以分为高/中/低等级。也可以使用更为复杂的分类方法，但所分的等级不应超过七个，并且它们应当从高到低表示对业务风险的影响。

从组织的业务风险概况开始，制定能将每个项目匹配到一个具体风险等级的项目评估准则。一个相似但是独立的分类模式也应当为数据资产建立。另外，应当根据对业务风险的潜在影响为每一项数据资产进行权衡和分类。

对收集的每个应用程序的信息进行评估，并根据总评估准则和目前使用数据资产的风险分类，为每个应用程序分配一个风险类别。这项工作可由安全团队集中进行；或由单独的项目团队，通过定制的问卷调查来收集必要的信息。

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

B. 建立并衡量每个分组的安全目的

为组织的应用程序文档实施了分类计划后，组织可以为直接的安全目标和保证计划路线图做出更细致地选择。

通过指定每个类别特定实践的重点，保证计划的路线图应为每个应用程序风险分类进行修改。对于每次执行保证计划时，通常会按照优先级将最高级别的目标设置在最高的风险应用程序等级上，并依次将不太紧要的目标排在较低或其他等级。

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，以换取接受权重的风险。但是，没有必要为每个风险类别构建单独的路线图，因为这样会使保证计划本身的管理效率低下。

结果

✦根据业务的核心价值，为每个项目定制保证计划；
✦整个组织都了解到了与安全相关的数据和应用程序资产；
✦使利益相关者更好地了解并认同风险。

额外成功指标

✦>90％的应用程序和数据资产在过去12个月内针对风险分类进行了评估；
✦>80％的员工在过去6个月内对相关应用程序和数据的风险评级有大致了解；
✦>80％的员工在过去3个月内对相关的保证计划路线图有大致了解。

额外成本

✦应用程序的建立或者许可证的支付以及数据风险分类计划；
✦计划更为精细的路线图计划开销。

额外人员

✦架构师（2天/年）
✦经理（2天/年）
✦业务拥有者（2天/年）
✦安全审计人员（2天/年）

相关等级

✦政策与遵守－2
✦威胁评估－2
✦设计审核－2

策略与指标 SM3

使安全成本与相关业务指标和资产价值相一致

措施

A. 引导周期性地全行业成本比较

以行业交流论坛、业务分析师和咨询公司或其他外部资源为来源，收集并研究关于安全成本的信息。特别是，需要确定一些关键因素。

首先，使用收集到的信息，以确定行业中同类组织在安全方面投入的平均成本。这可以通过自上往下的方式，通过对占预算的总百分比、收入等的估算来完成；或者使用自下往上的方式，通过确定被认为是组织正常的相关安全活动来完成。从整体上说，某些行业的情况很难估计，因此，需要从可访问的相关信息源尽可能多得收集信息。

研究安全成本的另一个目标，是确定您的组织是否可以从正在使用的第三方安全产品和服务中节约成本。当考虑决定更换供应商时，要将隐藏成本（如重新培训员工的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执行，并且每年至少进行一次。通过比较获得的信息应呈送给利益相关者，以更好地使保证计划与业务相一致。

B. 为以前的安全成本采集衡量标准

为以前的安全事件收集与项目特定相关的成本信息。例如，为弥补漏洞而耗费的时间和金钱、因系统停机而导致的经济损失、管理组织的罚款和收费、项目为特定的工具或服务一次性的安全支出等。

使用应用程序风险分类和为各类别分别规定的保证计划路线图，每个应用程序的基本安全成本最初可从与相应风险类别相关的成本估算出。

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，对风险等级不相称的数据值求和。这些信息表示在风险评估/分类中有错误；或为提供更有效的安全成本花销，需要调整组织的保证计划，以解决根本问题。

每个项目对安全支出的跟踪应当在每个季度的保证计划战略会议上完成，并且相关信息应由利益相关者每年进行至少一次审核和评估。为保证计划路线图的潜在影响，应讨论离群值和其他无法预见的成本。

结果

- ✦有助于制定每个案例安全支出决策的信息；
- ✦估算由于安全问题而在过去造成的损失；
- ✦考虑每个项目的安全支出与潜在的损失；
- ✦整个行业对于安全的勤奋付出。

额外成功指标

- ✦>80％的项目在过去3个月内报告了安全成本；
- ✦>1次的业界范围成本比较在过去1年内执行；
- ✦>1次的历史安全成本花销评估在过去1年内执行。

额外成本

- ✦为安全计划扩充或许可行业情报；
- ✦来自由成本核算、跟踪和评估而导致的计划开销。

人员

- ✦架构师（1天/年）
- ✦经理（1天/年）
- ✦业务拥有者（1天/年）
- ✦安全审计员（1天/年）

相关等级

- ✦漏洞管理－1

政策与遵守

	PC1	PC2	PC3
目标	了解组织的相关监管和遵守要求。	建立安全和遵守的基准线，并了解每个项目的风险。	要求遵守标准，并衡量项目是否符合全组织的政策和标准。
措施	A. 确定并监控外部的遵守驱动因素； B. 建立并维护遵守指导。	A. 为安全和遵守建立政策和标准； B. 建立项目审计实践。	A. 为项目建立遵守关卡； B. 为审计数据的采集，采用解决方案。
评估	✦是否大多数的项目拥有者知道他们项目的政策和标准的遵守状态？ ✦遵守需求是否被项目团队特别考虑到？	✦组织是否使用了一套政策和标准来控制软件开发？ ✦项目团队是否可以对政策和标准的遵守要求审计？	✦项目是否定期审计以确保政策和标准的遵守底线？ ✦组织是否系统性得使用审计以采集并控制遵守的证据？
结果	✦ 对于处理好的第三方审计结果而提高了保证； ✦ 根据遵守需求的优先顺序，调整内部资源； ✦ 及时发现影响组织的现行法律法规。	✦ 项目团队对安全和遵守的预期有了了解； ✦ 业务拥有者更好地了解其生产线中特定的遵守风险； ✦ 为将遵守与随机安全改进更有效地得到满足而优化方法。	✦ 整个组织都能了解到因不遵守政策与标准而导致的风险； ✦ 为项目级别的遵守提供明确保证； ✦ 准确地跟踪以前项目的遵守历史记录； ✦ 利用工具的有效审计流程，以减少人工工作量。

政策与遵守 PC1

了解组织的相关监管和遵守要求

措施

A. 确定并监控外部的遵守驱动因素

当组织有许多遵守要求时，该项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数据方式的法规和标准。如果可行的话，利用内部员工专门处理相关遵守事务。

根据组织的核心业务，研究并确定需要遵守或被认定为是业界规范的第三方监管标准。可能需要遵守的标准包括：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。在阅读并理解每项第三方标准后，收集与软件和数据相关的具体要求，并建立一个汇总列表，以将每个驱动因素（第三方标准）与特定的安全要求一一匹配对应。在这一阶段，尝试通过去除被认为是可选用的标准或仅使用推荐的标准，来限制数量要求。

组织需要至少每半年进行一次研究，以确保组织紧随第三方标准的最新更新。根据业界情况和遵守标准的重要性，这一措施可能对不同组织在工作量和涉及的人员等方面有所不同，但本措施必须被明确地执行。

B. 建立并维护遵守指导

根据软件的统一列表和来自遵守驱动因素与数据相关的要求，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。这个列表有时被称为控制声明，每个响应都应当获得组织的业务活动的概念，以确保达到了要求（或记录没能符合要求的原因）。

由于典型的审计实践常常涉及到检查一段控制声明是否充分，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、轻便的处理流程元素以涵盖组织能做到的基本遵守，来满足许多要求，继而使组织能够更好地得到保证。

研究具体的列表，确定主要的差距，以为在建立保证计划的未来规划工作中提供信息。就有关遵守差距的信息，与利益相关者进行交流，确保他们意识到不遵守政策与标准会带来哪些风险。

至少每半年需要与利益相关者更新并审核一次控制声明。由于标准遵守的驱动因素很多，因而经常进行更新是很有意义的。

结果

- ✦对于处理好的第三方审计结果而提高了保证；
- ✦根据遵守需求的优先顺序，调整内部资源；
- ✦及时发现影响组织的现行法律法规。

成功指标

- ✦>1次的遵守标准发现会议在过去6个月内举行；
- ✦在过去6个月内完成并更新了遵守标准检查列表；
- ✦>1次的与利益相关者的遵守标准审核会议在过去6个月内进行。

成本

- ✦初始创建并维护遵守标准检查列表。

人员

- ✦架构师（1天/年）
- ✦经理（2天/年）
- ✦业务拥有者（4天/年）

相关等级

- ✦策略与指标－1

政策与遵守 PC2

建立安全和遵守的基准线，并了解每个项目的风险

措施

A. 为安全和遵守建立政策和标准

从当前遵守标准指导开始，审核监管标准并记录任何可选的或建议的安全需求。同样，组织应当进行一些少量的研究，以发现其他一些在未来会发生改变的潜在相关遵守标准要求。

根据已知业务的安全驱动因素，添加一些额外要求来扩充该列表。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。

将普通的和相似的要求进行分类，并将每个组分类重新编写为更普遍的和更简单的声明，以满足所有的遵守驱动因素并提供一些额外的安全价值。通过每一次以建立一组内部政策和标准为目标的分组处理，可直接匹配回遵守标准的驱动因素和最佳实践。

值得一提的是，政策和标准不能包含对于项目团队难以遵循或遵循成本太高的要求。大约80%的项目应能够在中断最少的情况下遵守标准。这需要制定一个良好的交流计划，以宣传新的内部政策与标准，并在需要时帮助项目团队遵守标准。

B. 建立项目审计实践

为项目团队创建一个简单的审计流程，以要求并接受对内部标准的审计。审计通常由安全审计员执行，但是，如果有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，按优先级顺序将项目放入审计队列，这样，高风险软件的评估将能更早地执行或被更频繁地执行。此外，低风险项目可以放宽内部审计的要求，以使审计实践更加有效。

总的来说，每个进行的项目都应至少每半年进行一次审计。一般情况下，如果对应用程序保留了足够的审计信息，在初次审计完成以后执行的审计就会更简单。

向业务拥有者和其他利益相关者介绍这种服务，以便他们能够为他们的项目提出审计申请。根据内部标准，每个需求的详细通过或未通过的结果都应交给项目利益相关者进行评估。在可行的情况下，审计结果也应包含对影响的解释和修改建议。

结果

✦项目团队对安全和遵守的预期有了了解；
✦业务拥有者更好地了解其生产线中特定的遵守风险；
✦为将遵守与随机安全改进更有效地得到满足而优化方法。

额外成功指标

✦>75%的员工在过去6个月内大概了解了政策和标准；
✦>80%的利益相关者意识到了政策和标准的遵守状态。

额外成本

✦内部标准的扩充或许可；
✦每个项目来自遵守内部标准和审计的开销。

人员

✦架构师（1天/年）
✦经理（1天/年）
✦安全审计员（2天/项目/年）

相关等级

✦教育与指导－1&3
✦策略与指标－2
✦安全需求－1&3
✦安全架构－3
✦代码审核－3
✦设计审核－3
✦环境强化－3

政策与遵守 PC3

要求遵守标准，并衡量项目是否符合全组织的政策和标准

措施

A. 为项目建立遵守关卡

一旦组织针对安全性制定了内部标准后，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点。在项目通过内部标准审计并被认定遵守了标准以前，特殊点控制项目不能通过。

通常，遵守标准关卡位于软件发布的点上，这样在通过遵守标准检查之前，不允许发布该软件。留出足够的审核和修改时间很重要，因此，通常应较早开始进行审计，例如，在将发行版本交给QA的时候。

尽管遵守标准关卡是固定的标准，但旧项目或其他特定的项目可能无法遵守标准，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%。

B. 为审计数据的采集，采用解决方案

定期对项目团队进行审计的组织会逐渐积累大量的审计数据。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的审计数据。

对于内部标准的许多具体要求，现有的工具，比如：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守情况。自动检查遵守情况的目的在于既能提高审计的效率，又能实现更多的员工在正式审核前能够自我检查遵守情况。另外，自动检查不容易发生错误，并保证在发现问题后能尽快得到处理。

信息存储功能应当为每个项目支持对当前和历史审计数据的集中式访问。自动化解决方案另外还必须提供详细的访问控制功能，以仅让获得批准的、具有合法业务目的的个人访问审计数据。

所有关于访问遵守标准的数据以及申请访问权限的说明信息和处理流程应当向项目团队进行宣传。安全审计员初次引导项目团队时可能需要较长的时间。

结果

- ✦整个组织都能了解到因不遵守政策与标准而导致的风险；
- ✦为项目级别的遵守提供明确保证；
- ✦准确地跟踪以前项目的遵守历史记录；
- ✦利用工具的有效审计流程，以减少人工工作量。

额外成功指标

- ✦>80%的项目通过审计，以证明遵守了政策和标准；
- ✦每次自动审计使用的时间不到人工审计的50%。

额外成本

- ✦扩充或许可内部标准审计自动化工具；
- ✦审计关卡和例外处理流程的持续维护。

额外人员

- ✦开发人员（1天/年）
- ✦架构师（1天/年）
- ✦经理（1天/年）

相关等级

- ✦教育与指导－3
- ✦代码审核－2
- ✦安全测试－2

教育与指导

	EG1	EG2	EG3
目标	为开发人员提供关于以安全编程和部署为主题的资源。	为软件生命周期中所有的人员提供基于角色的安全开发详细指导。	实施综合的安全培训，并为员工进行基本知识的认证检验。
措施	A. 实施技术安全意识的培训； B. 建立并维护技术指导。	A. 实施针对特定角色的应用程序安全培训； B. 聘用安全指导专家增强项目团队。	A. 建立正式的应用程序安全支持门户网站； B. 建立基于角色的考试或认证制度。
评估	✦大多数的开发人员是否接受了高级别的安全意识培训？ ✦每个项目团队是否都接触了安全开发的最佳实例和指导？	✦开发过程中的大多数角色是否受到了根据角色的培训和指导？ ✦大多数的利益相关者是否能够在项目中安排安全指导员？	✦与安全相关的指导是否由组织集中控制并在组织中一致分部？ ✦大多数的人员是否为安全开发进行了测试以保证达到要求的最低技能底线？
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； ✦对安全问题的深入了解，鼓励更主动的安全计划。	✦有效地补救正在进行项目代码和旧版项目代码中的漏洞； ✦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。

教育与指导 EG1

为开发人员提供关于以安全编程和部署为主题的资源

措施

A. 实施技术安全意识的培训

采用来自组织内部或者外部的资源，对技术人员进行应用程序安全性基本原则的安全培训。通常，培训可通过两种方式进行：讲师讲授1－2天的培训课程；或者是基于计算机的培训课程，让每位开发人员以等量的学时学习各个方面的内容。

课程内容应包含概念和技术两方面的信息。适当的主题包括：关于高级别最佳实例的输入验证、输出编码、错误处理、日志记录、身份验证、授权。其他的培训内容还应包含普遍的软件漏洞，如在OWASP Top 10中列举的软件（Web应用程序、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）的常见漏洞。在可能的情况下，提供以特定编程语言编写的代码示例以及机房上机练习。

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人数进行授课（讲师授课或基于电脑的课程）。

B. 建立并维护技术指导

对于开发人员，收集并建立一个列表以包含有效的文件、网页和技术文档，用于提供特定于技术的安全建议。这些参考资料可以来自于互联网上许多可用的公开资源。如果开发环境中涉及到许多非常专业的技术或专利技术，可由熟悉安全问题的资深员工编写安全记录，并以一种专门的方式来创建知识库。

确保管理层了解了相关资源，并为员工介绍了其预期的用途。尽量简化指导，并及时更新参考列表，以避免混乱和不相关的内容。一旦建立了一种熟悉的等级后，技术参考资料可用作定性的检查表，以确保在开发过程中被阅读、理解并遵循。

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。

成功指标

✦>50%的开发人员在过去1年内大概了解了安全问题；
✦>75%的高级开发人员和架构师在过去1年内大概了解了安全问题；
✦在首次培训的3个月内，启动技术指导。

成本

✦培训课程的扩充或许可；
✦对技术指导的持续维护。

人员

✦开发人员（1－2天/年）
✦架构师（1－2天/年）

相关等级

✦政策与遵守－2
✦安全需求－1
✦安全架构－1

教育与指导 EG2

为软件生命周期中所有的人员提供基于角色的安全开发详细指导

措施

A. 实施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；另一种是计算机模块化的培训，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、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（讲师授课或基于计算机的培训）。

B. 聘用安全指导专家增强项目团队

聘用组织内部或外部的安全专家加入项目团队，以提供相关咨询。另外，应在组织内部公告安全专家的入职，以确保所有员工对此知晓。

安全专家可以在组织内部招募有经验的员工，通过让他们投入一些时间（最多10%）以进行安全指导。为提高效率，这些专家们应通过相互交流来确保他们了解彼此之间的专长，并将问题传递给相应的人员。

在软件生命周期中的任何时候都可能需要安全专家，需要这些安全专家的合适时候包括：产品的初步概念定义时、在完成功能或详细设计规范之前、开发过程中出现问题时、测试计划时、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作为在整个组织内沟通与安全相关信息的联络点，也可作为比纯粹集中式安全团队更熟悉当前项目团队的本地资源。

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；
✦对安全问题的深入了解，鼓励更主动的安全计划。

额外成功指标

✦>60%的开发人员在过去1年内接受了培训；
✦>50%的管理人员和分析师在过去1年内接受了培训；
✦>80%的高级开发人员和架构师在过去1年内接受了培训；
✦培训课程的有用性值>3.0 Likert。

额外成本

✦培训库的扩充或许可；
✦深谙安全知识的员工的实际训练。

额外人员

✦开发人员（2天/年）
✦架构师（2天/年）
✦经理（1-2天/年）
✦业务拥有者（1－2天/年）
✦QA测试员（1－2天/年）
✦安全审计员（1－2天/年）

相关等级

✦漏洞管理－1
✦代码审核－2
✦安全架构－2

教育与指导 EG3

实施综合的安全培训，并为员工进行基本知识的认证检验

措施

A. 建立正式的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，创建并公布一个集中式存储库（通常是一个内部网站）。可以用对组织有意义的任何方式创建指导准则，但是，必须要建立一个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以外，支持门户网站还应具有交互组件，比如：邮件列表、基于Web的论坛或wiki网页，以使内部资源交流与安全相关的主题并对信息进行分类，以备日后参考。

门户网站的内容应进行归类，以便根据一些常用的要素（比如：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）轻松地搜索。开发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计标准。

B. 建立基于角色的考试或认证制度

按角色或按培训班级/模块来创建并管理能力考试，以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，并将最低通过分数设置在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，但认证考试应至少每半年进行一次。

根据是否通过标准或例外绩效，应将员工分成不同等级。当有其他与安全相关的活动时，可要求已达到某个认证级别的个人，在实践完成之前结束该活动。比如，一个未通过认证的开发人员在未获得架构师认证的明确批准前，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，以通过个人责任来跟踪安全决策。从整体上看，这还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

- ✦有效地补救正在进行项目代码和旧版项目代码中的漏洞；
- ✦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。

额外成功指标

- ✦>80%的员工在过去1年内获得了认证。

额外成本

- ✦认证考试的扩充或许可；
- ✦对应用程序安全支持门户网站的持续维护和变更控制；
- ✦执行员工认证的人力资源和成本耗费。

额外人员

- ✦开发人员（1天/年）
- ✦架构师（1天/年）
- ✦经理（1天/年）
- ✦业务拥有者（1天/年）
- ✦QA测试员（1天/年）
- ✦安全审计员（1天/年）

相关等级

- ✦策略与遵守－2&3

威胁评估

	TA1	TA2	TA3
目标	确定并了解组织和单个项目的高级别威胁。	提高威胁评估的准确性，并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	A. 建立并维护特定应用程序的威胁模型； B. 根据软件架构建立攻击者概况。	A. 建立并维护每个项目的滥用用例模型； B. 为威胁的度量采用一个权重系统。	A. 明确评估来自第三方组件的风险； B. 用补偿控制详细描述威胁模型。
评估	✦组织中的大多数项目是否考虑并记录可能的威胁？ ✦组织是否了解并记录可能面对的攻击者类型?	✦项目团队是否定期为可能的滥用分析功能需求？ ✦项目团队是否为相关比较使用了一个威胁等级评定方法？ ✦利益相关者是否意识到相关的威胁和等级评定？	✦项目团队是否具体考虑了来自外部软件的风险？ ✦所有的保护机制和控制是否能捕获并匹配威胁？
结果	✦对可能产生负面结果的因素，有了深入的理解； ✦在项目团队中加强了对威胁的安全意识； ✦为组织记录了威胁。	✦为每个项目详细了解了可能的威胁； ✦为项目团队能更好得权衡决定的框架； ✦根据风险的权重，项目团队能对开发工作进行优先级排序的能力。	✦更深入的考虑对于每个软件项目的全部威胁概况； ✦详细的映射保证功能，以建立对每个软件项目的威胁； ✦根据每个软件项目的业务功能，建立文档以记录调查的对象。

威胁评估 TA1

确定并了解组织和单个项目的高级别威胁

措施

A. 建立并维护特定应用程序的威胁模型

根据每个软件项目的商业目的和商业风险（如果存在的话），确定每个项目团队在软件开发过程中可能的最坏情况。这可以通过简单的攻击树或者通过更正式的威胁建模方法实现，例如：微软的STRIDE，Trike等。

为建立攻击树模型，需用一句话描述每个最坏的情况，并将它们作为一个攻击者的高级目标标记出来。通过每个确定的攻击者目标，可以确定必须的先决条件，以便得以实现每个目标。这些信息应该在每个目标下面的分支中被捕获，每个分支下面的描述为逻辑与或者是逻辑或。一个“与”分支表明每个直接连接的子节点必须为真，以便实现父节点。一个“或”分行表明任意一个连接的子节点可为真，以便实现父节点。

除了威胁建模方法，还需审核每个当前和历史的功能要求，并扩展攻击树来表明每个相关的安全问题。反复地仔细研究攻击者可能达到的任何一个目标，分析每个错误情况的所有方式。当完成初步建立后，且软件有了明显的更改时，应用程序的威胁模型应该更新。这一评估过程应由高级开发人员和架构师以及一个或多个安全审计人员进行实施。

B. 根据软件架构建立攻击者概况

最初，根据软件项目进行评估，以确定对组织可能的威胁。对于这种评估，可能仅考虑了来自于攻击者有害意图的有限威胁，而忽略了其他的风险，比如：已知的威胁漏洞、潜在的弱点等。

根据普遍考虑到的外部攻击者和他们相关的攻击动机来设计。这个列表中，增加可能导致破坏的内部角色和内部攻击者的动机。基于正在考虑中的软件项目架构，它可以对每次一种架构类型进行更加有效的分析，而不是分析每个单独的项目中应用程序架构和业务目的都会受到类似的威胁。因为相同架构和业务目的的应用程序通常容易受到相同的攻击。

这一评估应由业务所有者和其他利益相关者执行，但也需要包括一个或多个安全审计员，以获得对威胁的更多了解。最后的目标是拥有一个简洁的威胁攻击者名单列表和相应的攻击动机。

结果

- ✦对可能产生负面结果的因素，有了深入的理解；
- ✦在项目团队中加强了对威胁的安全意识；
- ✦为组织记录了威胁。

成功指标

- ✦>50％的项目利益相关者在过去12个月内对相关项目的威胁模型有大概了解；
- ✦>75％的项目利益相关者对相关架构的攻击者概况有大致了解。

成本

- ✦建立并维护项目的威胁模型。

人员

- ✦业务拥有者（1天/年）
- ✦开发人员（1天/年）
- ✦架构师（1天/年）
- ✦安全审计员（2天/年）
- ✦经理（1天/年）

相关等级

- ✦策略与指标－1
- ✦安全需求－2

威胁评估 TA2

提高威胁评估的准确性，并深入了解每个项目的细节

活动

A. 建立并维护每个项目的滥用用例模型

进一步考虑组织的威胁，执行更正式的分析，以确定潜在的误用或滥用功能。通常，这一过程开始于正常使用情况的确定，比如：用例图。

如果没有使用正式的滥用实例建模技术，则使用常规的方法产生一个陈述，并以集思广益的方式决定陈述是否应该部分或者全部否定，从而为每种情况产生一组滥用实例。最简单的开始方法是以尽可能多的方式将“不”字插入在陈述之中，通常在名词和动词附近。每个使用情况应该产生一些可能被滥用的实例陈述。

进一步阐述滥用实例的陈述，以包括任何特定应用软件关注的业务功能。最终目标是为了完成滥用陈述，以形成一个软件不允许发生情况的模型。如果需要，这些滥用实例可以与现有的威胁模型相结合。

初步建立完成以后，滥用实例模型应该为新启动的项目在设计阶段更新。对于已有的项目，应为潜在的滥用分析新的需求，并应当为已建立的实际功能创建滥用实例。

B. 为威胁的度量采用一个权重系统

根据已创建的攻击概况，确定一个评级系统，以对威胁进行相对得比较。这个系统可以包含基于业务风险简单的高－中－低等级。但是，任何类型的度量方法都可以使用，只要其中所分的等级不超过5个。

确定评级系统后，应建立评估标准，并为每一个威胁分配一个等级。为了正确地进行该步骤，应在评级以前考虑威胁的各种其他重要因素。这些因素包括：资本和人力资源、内在的访问权限、技术能力、威胁模型的有关目标和攻击成功的可能性等。

为每一个威胁分配了一个等级后，优先考虑使用在开发生命周期中降低风险的措施。一旦为项目团队建立系统以后，应在新功能的设计或重构过程中予以更新。

结果

- ✦为每个项目详细了解了可能的威胁；
- ✦为项目团队能更好得权衡决定的框架；
- ✦根据风险的权重，项目团队能对开发工作进行优先级排序的能力。

额外的成功指标

- ✦>75％的项目团队确定并为威胁排序；
- ✦>75％的项目利益相关者在过去6个月里大概了解了相关项目的威胁和滥用模型。

额外成本

- ✦建立并维护威胁模型和攻击者概况的项目成本。

额外人员

- ✦安全审计员（1天/年）
- ✦业务拥有者（1天/年）
- ✦经理（1天/年）

相关等级

- ✦策略与指标－2
- ✦安全架构－2

威胁评估 TA3

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

活动

A. 明确评估来自第三方组件的风险

对组织开发的软件进行基于代码的评估，并确定任何来自于外部的组件。通常情况下，这包括了开源项目、购买现有的软件和组织开发的软件使用的在线服务。

对于每一个确定的组成部分，基于被攻破的潜在第三方部件，为软件项目详细描述攻击者概述。根据新确定的攻击者概述，并基于新攻击者的目的和能力，更新软件的威胁模型，以合并包含其他可能的风险。

除了威胁实例，还应当考虑可能影响你代码和设计的第三方软件威胁或者设计漏洞。根据来自威胁的潜在风险和已更新的攻击者概述的信息，详细描述你的威胁模型。

完成项目的最初进行后，必须在设计阶段或者每一个开发周期中更新并审核。这项活动应该由一个具备相关技术的安全审计员和业务利益相关者执行。

B. 用补偿控制详细描述威胁模型

执行评估，以正式确定威胁模型中能成功直接阻止攻击的前提条件因素。这些因素是正式解决来自软件直接风险的补偿性控制，也可以是软件本身的技术特点，还可以是在开发生命周期、基础设施功能等中的过程元素。

如果使用攻击树模型，每个分支所代表的逻辑关系可能是“与”或者“或”中的任意一个。因此，在一个“与”分支中通过减少一个前提条件，父节点和所有连接的叶节点可以被标记以作为减少。然而，在一个“或”节点的所有子节点必须防止在父节点以前被标记以作为减少。

除了威胁建模技术以外，确定补偿控制并直接批注威胁模型。其目标就控制而言，确定可被最大化标记数目的补偿控制。对于任何剩余可行的路径，需要确定反馈到组织策略中的潜在补偿控制。

在项目的最初执行后，这必须在设计阶段或每一个开发周期中更新和审核。这项活动应该由一个具备相关技术安全审计员和业务利益相关者执行。

结果

- ✦更深入的考虑对于每个软件项目的全部威胁概况；
- ✦详细的映射保证功能，以建立对每个软件项目的威胁；
- ✦根据每个软件项目的业务功能，建立文档以记录调查的对象。

额外的成功指标

- ✦>80%的项目团队在每一个执行周期以前更新了威胁模型；
- ✦>80%的项目团队在每次发布以前更新了第三方组件的目录清单；
- ✦>50%的所有安全事件在过去的12个月里由威胁模型提前检验出来。

额外成本

- ✦来自维护详细威胁模型和扩展攻击者概述的项目成本。
- ✦发现所有的第三方附属物。

额外人员

- ✦业务拥有者（1天/年）
- ✦开发人员（1天/年）
- ✦架构师（1天/年）
- ✦安全审计员（2天/年）
- ✦经理（1天/年）

相关等级

- ✦安全需求－2&3

安全需求

	SR1	SR2	SR3
目标	在软件需求分析阶段明确地将安全考虑在内。	根据业务逻辑和已知风险增加安全需求的深度。	为所有软件项目和第三方的附属项目强制要求安全需求。
措施	A. 从业务功能推导出安全需求； B. 为需求评估安全和遵守指导。	A. 为资源和能力建立一个访问控制矩阵； B. 根据已知风险指定安全需求。	A. 将安全需求写入供应商协议中； B. 为安全需求扩展审计计划。
评估	✦大多数项目团队在开发过程中是否明确阐述的一些安全需求？ ✦项目团队是否从最佳实例和遵守的指导中引导需求？	✦大多数的业务拥有者是否审核相关项目的访问控制矩阵？ ✦项目团队是否根据来自于其他安全活动的反馈来详细阐述需求？	✦大多数的业务拥有者是否为安全需求审核厂商的协议？ ✦项目团队描述的安全需求是否被审计？
结果	✦开发工作和业务风险高度一致； ✦专门为安全需求获得行业中的最佳实例； ✦利益相关者对于采取为降低来自软件风险的措施的意识。	✦详细了解对于业务逻辑的攻击实例； ✦根据可能遭受的攻击，对安全功能的开发工作设置优先级； ✦为特点和安全工作间的权衡，做出更有理论依据的决策； ✦利益相关者可以更好地避免存在安全漏洞的功能要求。	✦为安全期望正式从外部代码设定基线； ✦每个项目团队集中使用的安全措施信息； ✦根据应用程序风险和所需安全需求，调整项目资源。

安全需求 SR1

在软件需求分析阶段明确地将安全考虑在内

措施

A. 从业务功能推导出安全需求

对指明的业务逻辑和每个软件项目总体行为的功能需求进行审核。在收集了项目需求后，进行评估，以获得相关的安全需求。即使软件将由第三方完成，这些需求一旦确定，应包含在功能需求中交付给厂商。

对于每个功能需求，安全审计员应带领利益相关者了解过程中涉及到安全方面的期望。通常情况下，澄清每个需求的问题包括了对于数据安全、访问控制、处理的完整性、业务功能的关键性、职责分离、运行时间等的期望。

确保所有安全需求，在总体上为编写出好的需求而保持相同原则，是很重要的。具体来说，这些安全需求应该是具体的、可测量的、合理的。

为所有正在进行项目中的新需求进行这项处理。对于已有的功能，建议进行同样的处理以作为一个为将来安全重构的差距分析。

B. 为需求评估安全和遵守指导

确定业界的最佳实践，以作为项目团队的需求。这些最佳实践可以从公开的指导准则、内部或外部的指导准则/标准/政策、已建立的遵守需求中选择。

重要的是，不要尝试在每一个开发迭代过程中引入太多的最佳实践需求，因为需要对于设计和编程有一个时间权衡。建议的方法是在连续的开发周期中逐渐地增加最佳实践，以增强软件整体的安全保证。

对于现有的系统，重构安全最佳实践会是一个复杂的任务。如有可能机会，在添加新的特征时增加安全需求。至少，应当执行分析，以确定适用的最佳实践，从而帮助未来的项目规划。

这项审核工作应当由一个安全审计员和业务利益相关者执行。而高级开发人员、架构师和其他技术利益相关者也应当将他们的设计和编程知识经验带入到决策过程之中。

结果

✦开发工作和业务风险高度一致；
✦专门为安全需求获得行业中的最佳实例；
✦利益相关者对于采取为降低来自软件风险的措施的意识。

成功指标

✦>50%的项目团队有明确定义了的安全需求。

成本

✦来自每个开发周期额外安全需求的项目成本。

人员

✦安全审计员（2天/年）
✦业务拥有者（1天/年）
✦经理（1天/年）
✦架构师（1天/年）

相关等级

✦教育与指导－1
✦政策与遵守－2
✦设计审核－1
✦代码审核－1
✦安全测试－1

安全需求 SR2

根据业务逻辑和已知风险增加安全需求的深度

措施

A. 为资源和能力建立一个访问控制矩阵

根据应用程序的业务目的，确定用户和操作人员角色。此外，通过收集所有的有关数据资产和应用程序的特点，建立资源和能力列表，以保护任何形式的访问控制。

在一个以角色作为其中一个轴并以资源作为另外一个轴的简单矩阵中，考虑每种角色和每种资源的关系，并注意就访问控制而言，在每个交叉点上的正确行为。

需要注意的是，对于数据资源，访问权限是指创建、读取、更新和删除。对于资源的访问功能，访问权限分级很可能会是应用程序特定的，但至少注意，角色应允许使用这些功能。

此权限矩阵将作为一个为全部系统业务逻辑记录正确访问控制权限的产生物。就这点而言，它应由项目团队和业务利益相关者共同建立。经过初步建立，它应当由业务利益相关者在每次发布前进行更新，但通常是在设计阶段将要开始的时候进行。

B. 根据已知风险指定安全需求

明确审核现有表明特定于组织或具体项目的安全风险的对象，以便更好地了解该软件的整体风险概况。如果可以的话，利用以下资源：高层次的业务风险；私有的应用程序威胁模型；来自设计审核、代码审核、安全测试的结果等等。

除了审核现有的对象以外，为一个应用程序使用滥用实例模型，以确定模型的具体安全需求，直接或间接地促进减少滥用的情况。

这个过程应该由业务拥有者和安全审计员共同执行。最终，由风险概念导致的新安全需求，成为规划阶段内建的步骤，使新发现的风险专门由项目团队评估。

结果

✦详细了解对于业务逻辑的攻击实例；
✦根据可能遭受的攻击，对安全功能的开发工作设置优先级；
✦为特点和安全工作间的权衡，做出更有理论依据的决策；
✦利益相关者可以更好地避免存在安全漏洞的功能要求。

额外成功指标

✦75％的项目在过去6个月里更新了滥用实例模型。

额外成本

✦来自建立和维护滥用实例模型的项目开支。

额外人员

✦安全审计员（2天/年）
✦经理（1天/年）
✦架构师（1天/年）
✦业务拥有者（1天/年）

相关等级

✦威胁评估－1&3
✦策略与指标－1

安全需求 SR3

要求对所有软件项目和第三方附属物的安全需要处理

措施

A. 将安全需求写入供应商协议中

除了在前面的分析中已确定的各种安全需求外，还可以从第三方协议中获得其他的安全利益。通常情况下，在组织内部开发需求和可能的高层次设计，而详细的设计和编程执行则交给供应商完成。

由于每个外部开发的组成部分的具体分工不同，需要确定具体的安全措施和技术评估标准，以添加在供应商的合同中。通常，这是一组包含了“设计审核”“代码审核”“安全测试”实践的措施。

协议内容的修改应该是与供应商对每个实例逐一修改，因为添加额外的需求往往意味着成本的增加。每个潜在安全活动的成本，应该与每个组件的使用活动或者考虑到的系统的利益相权衡。

B. 为安全需求扩展审计计划

将安全需求完整性的检查纳入常规项目审计之中。由于在没有项目具体知识信息的情况下很难判断，审计应专注于检查项目的对象，比如需求或设计文档，以作为执行了适当种类分析的证据。

特别是，每一个功能需求，应用基于业务驱动力和期望的滥用实例的安全需求注明。整个项目需求应包含从最佳实践的指导和标准得到的一个需求列表。此外，应该还有一个明确的未完成的安全需求列表，以及一个未来版本公布的时间估计表。

这项审计应在每个开发迭代中进行，理想的情况是在需求处理快完成时结束，但必须在软件发布以前完成审计。

结果

✦为安全期望正式从外部代码设定基线；
✦每个项目团队集中使用的安全措施信息；
✦根据应用程序风险和所需安全需求，调整项目资源。

额外成功指标

✦>80%的项目在过去6个月里通过了安全需求审计；
✦>80%的供应商协议在过去12个月里为合同的安全需求进行了分析。

额外成本

✦因安全需求增加而增加的软件外包成本；
✦因安全需求的发行关卡造成的当前项目。

额外人员

✦安全审计员（2天/年）
✦经理（2天/年）
✦业务拥有者（1天/年）

相关等级

✦威胁评估－3
✦政策与遵守－2

安全架构

	SA1	SA2	SA3
目标	将主动安全指导的想法引入到软件设计过程中。	将软件设计过程引导向已知安全服务和默认安全设计。	正式控制软件设计过程并验证安全部件的使用。
措施	A. 维护推荐的软件框架列表； B. 将安全原则明确运用到设计中。	A. 明确并促进安全服务和基础设施； B. 明确来自架构的安全设计模式。	A. 建立正式的参照架构和平台； B. 验证框架、模式和平台的使用。
评估	✦项目团队是否提供一个推荐的第三方部件列表？ ✦大多数的项目团队是否意识到安全设计原则并应用它们？	✦您是否为项目团队宣传可共享的安全服务？ ✦项目团队是否提供了基于应用程序架构的规范设计模式？	✦项目团队是否从中央控制平台和框架建立软件？ ✦项目团队是否为安全架构部分的使用进行审计？
结果	✦专门防止意外的依赖性和选择一次性实现； ✦利益相关者意识到因选择的文档和框架而增加的项目风险； ✦为将安全机制积极得应用到设计中，而在开发过程中建立协议。	✦将资产详细地映射到用户角色，以鼓励在设计时更好的区分； ✦为提供安全保证和功能，而可重复使用的设计模块； ✦使用已建立的安全设计技术，以增加对软件项目的信心。	✦自定义的应用程序开发平台，以提供内置的安全保护； ✦整个组织都期望在开发过程中针对安全做出更积极的努力； ✦根据业务需要，利益相关者可以为安全设计做出更好的权衡决定。

安全架构 SA1

将主动安全指导的想法引入到软件设计过程中

措施

A. 维护推荐的软件框架列表

在整个组织内贯穿软件项目，确定正在使用的常用第三方软件库和框架。一般来说，这种需要不是对于依赖软件组件的详尽搜索，而是为了获得最常用的高级别组件。

根据第三方组件的核心特点，从部件列表将这些组件分为不同的功能组。此外，注意到跨越项目团队的每个组件的普遍使用情况， 以权衡第三方代码的信赖度。使用此加权列表作为指导，创建一个组件清单，以宣传作为开发组织的推荐组件。

对于推荐列表中包含的内容，有几个决定因素。虽然列表可以不进行专门研究就可创建列表，但是我们建议，调查每个事件的历史、跟踪记录漏洞响应、对组织而言功能的适当性、第三方组件使用的复杂性等等。

这份列表应由高级开发人员和架构师创建，但还应包括经理和安全审计员的介入。创建后，这个满足功能分组的推荐组件列表应当向开发组织进行宣传。最终目标是为项目团队提供为人熟知的默认资源。

B. 将安全原则明确运用到设计中

在设计过程中，项目团队的技术人员应当使用一个简短的安全指导原则列表，作为系统详细设计的核对表。通常情况下，安全原则包括深度防御、最薄弱环节的保护、安全功能的简单设计、安全故障、安全性和实用性的平衡、最低权限的使用、模糊安全的避免等。

特别是在边缘界面，设计团队应当考虑整个系统范围中的每一项原则，并确定可添加在每个这样的界面上的特点，以加强安全性。一般来说，这些应该受到限制，使其只需在功能要求常规实现成本之外做出少量额外工作，其他较大的工作量应在未来发布中提及并规划。

这个过程应该在安全意识培训后，由每个项目团队执行，这有助于和更加安全、精明的工作人员进行合作，以作出设计决定。

结果

✦专门防止意外的依赖性和选择一次性实现；
✦利益相关者意识到因选择的文档和框架而增加的项目风险；
✦为将安全机制积极得应用到设计中，而在开发过程中建立协议。

额外成功指标

✦>80％的开发人员在过去1年里大概了解了软件框架建议；
✦>50％的项目设计了安全原则的自我报告应用。

额外成本

✦对软件框架建议的建立、维护和了解；
✦项目对安全原则的持续分析和应用。

额外人员

✦架构师（2－4天/年）
✦开发人员（2－4天/年）
✦安全审计员（2－4天/年）
✦经理（2天/年）

相关等级

✦教育与指导－1

安全架构 SA2

将软件设计过程引导向已知安全服务和默认安全设计

措施

A. 明确并促进安全服务和基础设施

组织应当确定与安全功能相关的共享基础设施或服务。这通常包括：单点登录服务、企业目录系统、访问控制或有权限的服务、身份验证系统。通过收集和评估可重复使用的系统，建立这种资源的清单，并根据系统履行的安全机制将它们分类。这也有助于考虑为什么一个开发团队想要整合这种资源，比如：使用共享资源的好处。

如果每个类别中存在多种资源，一个组织应该在该类别中选择并规范化一个或多个类别的共享服务。由于未来软件的开发将依赖于这些选择的服务，每项服务都应该进行彻底地审计，以确保理解了安全态势的底线。对于每个选定的服务，应由开发团队建立设计指导，以了解如何与系统集成。这种指导在集成后，应当向开发团队提供培训、指导、指导方针和标准。

这样做的好处包括：促进已知的安全系统、简化项目设计团队的安全指导、为围绕使用共享安全服务的应用程序建立保证并明确方向。

B. 明确来自架构的安全设计模式

就常用架构类型而言，组织中的每个软件项目都应该被归类。常见的类别包括：客户服务器应用程序、嵌入式系统、桌面应用程序、面向Web的应用程序、Web服务平台、事务处理中间件系统、主机应用程序等。根据组织的专业性，可能需要根据开发语言、处理器架构、甚至部署的时间安排，做出更详细的分类。

对于普通的软件架构类型，可衍生出一系列代表执行安全功能可靠方法的常用设计模式，并应用到组织软件项目的每个设计之中。这些安全设计模式代表可研究或购买的通用设计元素，如果这些自定义的模式根据你的组织要求更加具体，它通常会更有效。示例模式包括：单点登录子系统、跨层代表模型、加强的界面设计、职能分离授权模型、集中化的日志模式等。

由架构师、高级开发人员和其他技术利益相关者，在设计阶段确定适用的和适当的身份验证工作模式。

结果

✦将资产详细地映射到用户角色，以鼓励在设计时更好的区分；
✦为提供安全保证和功能，而可重复使用的设计模块；
✦使用已建立的安全设计技术，以增加对软件项目的信心。

额外成功指标

✦>80％的项目在过去6个月里更新了访问权限矩阵；
✦>80％的项目团队在过去6个月里大概了解了应用程序的安全模式。

额外成本

✦扩充或者许可应用程序安全模；。
✦持续维护项目的访问权限矩阵。

额外人员

✦架构师（2－4天/年）
✦开发人员（1－2天/年）
✦经理（1－2天/年）
✦业务拥有者（1天/年）
✦安全审计员（1－2天/年）

相关等级

✦教育与指导－1

安全架构 SA3

正式控制软件设计过程并验证安全部件的使用

措施

A. 建立正式的参照架构和平台

当与共享的安全服务集成、并使用每种架构具体的安全模式后，项目团队应选择一组执行这些功能的代码集，以作为一个共享代码库的基础。这些共享的基本代码最初可作为每个项目都需使用的一组普遍推荐库集合，它可以随着时间的推移，发展成为一个或多个软件架构，以代表项目团队建立他们自己软件的参考平台。参考平台的示例包括： Web应用程序视图控制器架构、支持事务处理的后端系统库文件、Web服务平台框架、客户端服务器应用程序框架、中间件的可插拔业务逻辑框架等。

另一种建立初始参考平台的方法是在生命周期的早期选择一个特定的项目，并让安全人员使用该生命周期，以一个通用的方式建立安全功能，以便从项目中提取该功能并利用到该组织的其他项目中。

不管采用什么样的建立方式，参照平台在审计和与安全相关的审核的速度方面有优势，并可以提高开发效率，降低维护费用。

架构师、高级开发人员和其他技术利益相关者应参与设计和建立参照平台。创建后，必须有一个团队对其进行持续的维护和更新。

B. 验证框架、模式和平台的使用

在项目的例行审计中，需要对项目工件进行额外分析，以衡量推荐框架、设计模式、共享的安全服务以及引用参考平台的使用情况。虽然在例行审计中进行，这一活动的目的是尽可能多地从项目团队中收集反馈信息，以衡量他们各自积极的安全工作。

总体而言，在项目团队中验证几个因素是很重要的。确定非推荐框架的使用，以确定在推荐框架与该组织的功能需求之间是否存在差异。检查未使用或不正确使用的设计模式和引用参考平台模块，以确定是否需要更新。另外，随着组织的发展，项目团队希望看到在引用参考平台上执行更多不同的功能。

这项分析由任何具有安全常识的技术人员执行。而收集的度量标准，则由经理和利益相关者从每个项目分析整理。

结果

✦自定义的应用程序开发平台，以提供内置的安全保护；

✦整个组织都期望在开发过程中针对安全做出更积极的努力；

✦根据业务需要，利益相关者可以为安全设计做出更好的权衡决定。

额外成功指标

✦>50％的项目使用了参考平台；

✦>80％的项目在过去6个月里报告了框架、模式和平台的使用反馈信息；

✦对于项目组报告指南和平台有用性>3.0 Likert值。

额外成本

✦建立或者许可参考平台；

✦对参考平台正在进行的维护和支持；

✦在审计过程中验证使用的项目开支。

额外人员

✦经理（1天/年）

✦业务拥有者（1天/年）

✦架构师（3－4天/年）

✦开发人员（2－3天/年）

✦安全审计员（2天/年）

相关等级

✦政策与遵守－2

✦设计审核－3

✦代码审核－3

✦安全测试－3

设计审核

	DR1	DR2	DR3
目标	为软件设计提供专门的审核，以确保排除已知风险的最低线。	根据安全的最佳实践为软件设计审核提供评估服务。	需求评估并验证已完成部分，以详细了解保护机制。
措施	A. 确定软件攻击层面； B. 根据已知安全需求分析设计。	A. 检查提供安全机制的完整性； B. 为项目团队部署设计审核服务。	A. 为敏感资源开发数据流图； B. 为设计审核建立发布关卡。
评估	✦项目团队是否记录软件设计的攻击范围？ ✦项目团队是否按照已知的安全风险检查软件设计？	✦大多数项目团队是否为安全机制具体分析设计元素？ ✦大多数项目业务拥有者是否知道如何获得一个正式的设计审核？	✦设计审核过程中是否包括详细的数据级分析？ ✦常规项目审计是否需要一个设计审核结果的最低线？
结果	✦非常了解架构的隐含安全影响； ✦实现开发团队对于安全最佳实践而进行设计自检； ✦执行项目级设计审核的简便处理。	✦正式提供评估服务，以持续审核架构的安全； ✦在维护模式和原有系统中准确找到安全漏洞； ✦项目利益相关者对于软件如何提供保护保证有深入的了解。	✦系统设计过程中弱点的细化粒度，以鼓励更好的区分； ✦整个组织都注意到项目的架构应符合基本安全期望； ✦在已知缺陷减少方面，对各项目的效率和进展进行比较。

设计审核 DR1

为软件设计提供专门的审核，以确保排除已知风险的最低线

措施

A. 确定软件攻击层面

为每个软件项目建立一个整体架构的简化视图。通常情况下，应该基于项目对象，比如：高层次的需求和设计文档、与技术人员面谈或基于代码模块级的审查。捕获系统中的高层次模块非常重要，一个良好的细化略缩图规则可确保整个系统的审核图在一页上就能显示。

根据每个单页的视图架构，分析每个组件的界面对于授权用户、匿名用户、操作人员和应用程序特定角色等的可访问性。根据单页视图的上下文，应考虑提供界面的组件，以在图表上查找功能委派或传递到其他组件的数据。将具有类似可访问性文档的界面和组件进行分组，以获得作为软件的攻击层面。

对于每一个界面，进一步阐述单页视图，以记录任何与安全相关的功能。根据已确定界面组组成的攻击面，检查模型设计的一致性，以及如何保护具有类似访问权限的界面。任何违反一致性的信息都会记录在评估结果中。

这一分析应由项目组内部或外部通晓安全问题的技术人员执行。通常情况下，在初始建立完成后，当在设计阶段对边缘系统界面进行添加或者更改时，只需在图表和攻击层面更新即可。

B. 根据已知安全需求分析设计

无论是已正式还是非正式确认的安全需求，都应确定并得到收集。另外，确定并包含系统安全操作所依靠的任何安全假设。

根据系统架构的单页图表，审核已知安全要求列表上的每一项。阐述该图表，显示针对每条安全需求的设计级功能。如果系统较大或比较复杂时，则可创建独立的细分图表，以简化捕获此信息的操作。总体目标是由系统设计解决验证每一条已知的安全需求。任何未在设计级明确提供的安全需求都应作为评估结果被记录。

这一分析应由通晓安全问题的技术人员执行，同时还需架构师、开发人员、管理人员和组织拥有者的介入。当安全需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

✦非常了解架构的隐含安全影响；
✦实现开发团队对于安全最佳实践而进行设计自检；
✦执行项目级设计审核的简便处理。

成功衡量标准

✦>50%的项目在过去12个月里更新了攻击层面分析；
✦>50%的项目在过去12个月里更新了安全需求设计级分析。

成本

✦扩充并维护每个项目的架构图；
✦攻击层面和安全需求设计调查的持续项目开销。

人员

✦架构师（2—3天/年）
✦开发人员（1—2天/年）
✦经理（1天/年）
✦安全审计员（1天/年）

相关等级

✦安全需求—1

设计审核 DR2

根据安全的最佳实践为软件设计审核提供评估服务

措施

A. 检查提供安全机制的完整性

对于高层次体系结构图中模块的每个界面，通过安全机制列表和系统分析，提供正式的迭代。这种类型的分析应该在内部界面（例如：层之间）和外部界面（例如：组成攻击层面的界面）上进行。

需要考虑的六个主要安全机制包括：身份认证、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，还需考虑加密和会话管理机制。对于每一个界面，确定在系统设计中是否提供了每种机制，并记录遗漏或者不明确的功能。

这一分析应由通晓安全问题的技术人员执行，并由项目团队提供具体特定应用的知识。这种分析应在每次发行时进行，通常是在设计阶段临近结束的时候。经初步分析后，后续版本都需在开发周期修改的基础上更新结果。

B. 为项目团队部署设计审核服务

制定一个处理流程，以便项目的利益相关者可以要求进行设计审核。这项服务可在组织内部集中提供，或由目前的工作人员分布式提供。但所有的审核人员必须针对执行审核的完整性和一致性进行培训。

审核服务应在审核请求队列中被集中管理，以由熟悉组织的整体业务风险概况的高级管理人员、架构师和利益相关者分类筛选。这使得项目审核的优先级与整体业务风险相对应。

在一个设计审核中，审核团队应与项目团队合作，以收集足够的信息去系统地阐述对攻击层面的理解、项目特定的安全需求与设计元素的匹配，并验证模块界面的安全机制。

结果

✦正式提供评估服务，以持续审核架构的安全；
✦在维护模式和原有系统中准确找到安全漏洞；
✦项目利益相关者对于软件如何提供保护保证有深入的了解。

额外成功指标

✦>80％的利益相关者在过去6个月里大概了解了审核请求的状态；
✦>75％的项目在过去12个月里进行了设计审核。

额外成本

✦扩充、培训并维护设计审核团队；
✦项目审核活动的持续开销。

额外人员

✦架构师（1－2天/年）
✦开发人员（1天/年）
✦经理（1天/年）
✦安全审计员（2－3天/年）

相关等级

✦教育与指导－2
✦策略与指标－2

设计审核 DR3

需求评估并验证已完成部分，以详细了解保护机制

措施

A. 为敏感资源开发数据流图

根据软件项目的业务功能，进行分析，以识别关于高风险功能的系统行为的详细信息。通常情况下，高风险的功能将关联到进行创建、获取、更新和删除敏感数据。除了数据，高风险功能还包括拒绝服务或危害项目特定的关键业务逻辑。

对于每个已确定的数据源或业务功能，选择并使用标准化的符号来获取相关的软件模块、数据源、角色以及它们之间的信息流。通常，从高级别的设计图开始，并不断充实相关细节信息，同时删除与敏感资源不对应的元素部分。

当项目数据流图创建以后，对其进行分析并决定设计过程中的内部堵塞点。一般来说，这些点都将是能处理不同敏感等级数据的独立软件模块，或是不同业务功能关键性级别的访问关卡。

B. 为设计审核建立发布关卡

建立一致的设计审核计划以后，下一步骤就是在软件开发生命周期中设置一个特定点。在执行设计审核，且审核结果被审核并接受以前，项目无法通过这个特定点。为通过该点，应设置期望的基准水平，例如：不允许任何具有高危险性调查结果的项目通过，并且所有的结果必须被业务拥有者所接受。

一般来说，设计审核应发生在设计阶段快结束的时候，以协助及早发现安全问题。但它必须在项目团队生成发布以前执行。

对于旧系统或暂未活动的项目，应建立例外处理流程，以允许项目的持续操作，但需要有一个明确的审核时间计划安排，以阐明现有系统中存在的任何隐藏漏洞。例外应限制于不超过所有项目的20%。

结果

✦系统设计过程中弱点的细化粒度，以鼓励更好的区分；
✦整个组织都注意到项目的架构应符合基本安全期望；
✦在已知缺陷减少方面，对各项目的效率和进展进行比较。

额外成功指标

✦>80%的项目在过去的6个月里更新了数据流图；
✦>75%的项目在过去6个月里通过了设计审核审计。

额外成本

✦维护正在进行项目的数据流图的开销；
✦由于设计审核审计失败，而导致的组织项目延迟开支。

额外人员

✦开发人员（2天/年）
✦架构师（1天/年）
✦经理（1-2天/年）
✦业务拥有者（1-2天/年）
✦安全审计人员（2－3天/年）

相关等级

✦安全架构－3
✦代码审核－3

代码审核

	CR1	CR2	CR3
目标	随机查找基本的代码级漏洞和其他高风险安全问题。	通过自动化方式在开发过程中使代码审核更加准确和有效。	必须进行全面的代码审核过程，以发现语言级别和特定应用程序的风险。
措施	A. 根据已知安全需求建立审核检查列表； B. 为高风险代码执行定点审核。	A. 使用自动化的代码分析工具； B. 将代码分析集成到开发流程当中。	A. 为特定应用程序问题自定义代码分析； B. 为代码审核建立发布关卡。
评估	✦大多数的项目团队是否拥有基于普遍问题的审核清单列表？ ✦项目团队是否通常针对选择的高风险代码执行审核？	✦大多数项目团队是否可以使用自动的代码分析工具去查找安全问题？ ✦大多数项目业务拥有者是否持续要求并审核来自代码审核的结果？	✦项目团队是否根据应用程序特定的编码标准，使用自动化的方式去检查代码？ ✦常规项目审核是否需要一个为发布代码审核结果的最低线？
结果	✦检验普通代码中可能被发现或导致攻击的漏洞； ✦为导致严重安全影响的编码错误而进行的简便审核； ✦为安全保证的基础代码级审慎调查。	✦开发过程实现了代码级安全漏洞的持续自我检查； ✦将常规分析的结果，以对每个团队安全编码习惯的历史数据进行编译； ✦利益相关者注意到没有缓和的漏洞，从而做出更好的权衡分析。	✦增加对代码分析的准确性和适用性的信心结果； ✦整个组织对安全编码期望的基线； ✦项目团队为判断代码级安全设立了目标。

代码审核 CR1

随机查找基本的代码级漏洞和其他高风险安全问题

措施

A. 根据已知安全需求建立审核检查列表

根据项目已知的安全需求，推导出针对安全性的简便代码检查列表。这些检查可以具体到围绕功能需求的安全问题的检测，或针对基于执行语言、平台、代表性技术堆栈等的安全编码最佳实践的检测。由于这些变化，往往需要一套检查列表以覆盖在组织内部不同类型的软件开发。

不论是用公开可用的资源进行创建或者购买，技术利益相关者，比如：开发经理、架构师、开发人员和安全审计员，应审核清单列表的有效性和可行性。重要的是要保持列表简短并且简单，以通过对于代码的人工或者简单搜索工具查找高优先级问题。自动化代码分析工具也可以用来实现同一目标，但为了使扫描和审核过程更加有效，也应该对分析工具进行定制，以将总体安全检查的整体集合缩小为一个有价值的小集合。

开发人员应为他们的工作职能大概了解列表清单的目标。

B. 为高风险代码执行定点审核

由于代码级的漏洞在关系到软件安全性的关键部分发生，会带来明显的影响，项目团队应针对常见漏洞进行高风险模块的审核。高风险的功能的常见例子包括：身份验证模块、访问控制增强点、会话管理模块、外部界面、输入校验器和数据解析器等。

利用代码审核检查列表，该分析可作为开发流程的一个普通部分执行，但进行更改时，项目团队成员会被分配给要审核的模块。安全审计员和自动审核工具也可以被用于进行审核处理。

在更改和审核高风险代码的开发周期中，开发经理应当在其他项目利益相关者的介入下，将找到的结果进行分类，并适当的将修复工作进行优先级排序。

结果

✦检验普通代码中可能被发现或导致攻击的漏洞；
✦为导致严重安全影响的编码错误而进行的简便审核；
✦为安全保证的基础代码级审慎调查。

成功指标

✦>80%的项目在过去6个月里大概了解了相关代码审核检查表；
✦>50%的项目在过去6个月里对高风险代码执行了代码审核；
✦开发团队代码审核列表的有用性值，大于3.0 Likert值。

成本

✦扩充或者许可代码审核检查列表；
✦高风险代码审核活动的项目持续开销。

人员

✦开发人员（2－4天/年）
✦架构师（1－2天/年）
✦经理（1－2天/年）
✦业务拥有者（1天/年）

相关等级

✦安全需求－1

代码审核 CR2

通过自动化方式在开发过程中使代码审核更加准确和有效

措施

A. 使用自动化的代码分析工具

很多代码级的安全漏洞因复杂而非常难以理解，往往需要仔细检查才能被发现。但是，有很多有用的自动化解决方案，可针对bug和漏洞进行自动得代码分析。

有商业的和开源的产品覆盖了常见的编程语言和框架。对于选择一个适当的代码分析解决方案，需要根据以下一些要素，包括：检查的深度和准确性、产品的可用性和使用模型、可扩展性和自定义功能、组织架构和技术堆栈的适用性等。

在选择过程中，由通晓安全知识的技术人员以及开发人员和开发管理人员介入，并与利益相关者一起审核总体结果。

B. 将代码分析集成到开发流程当中

一旦选中代码分析解决方案后，它必须被纳入到开发过程中，以鼓励项目团队利用它的功能。要做到这一点的一个有效方法是设置基础架构，在代码开发时直接进行自动扫描；或是从项目代码库获取代码进行自动扫描。这样，可以及早获得结果，从而使开发团队能够在发布以前进行自我检查。

在旧系统或正在进行的大型项目中，一个潜在的问题是，代码扫描器通常会报告发布以前尚未更新模块的结果。如果自动扫描设置为定期运行，避免审核开销的有效策略是：考虑限制那些已被添加、删除、或自上次扫描后发生的改变。如果不能忽略剩下的结果，那么开发经理应当与安全审计员、利益相关者和项目团队合作，制订一个针对其他结果的解决计划。

如果在发布以前仍然存在代码审核未能解决的结果，那么项目利益相关者必须审核并接受这些结果。

结果

✦开发过程实现了代码级安全漏洞的持续自我检查；
✦将常规分析的结果，以对每个团队安全编码习惯的历史数据进行编译；
✦利益相关者注意到没有缓和的漏洞，从而做出更好的权衡分析。

额外成功指标

✦>50％的项目在过去6个月里进行了代码审核并由利益相关者认可；
✦>80％的项目在过去1个月里查看了自动代码审核的结果。

额外成本

✦代码分析解决计划的研究和选择；
✦自动化整合的初步花销和维护；
✦自动化代码审核和降低风险的项目持续开销。

额外人员

✦开发人员（1－2天/年）
✦架构师（1天/年）
✦经理（1－2天/年）
✦安全审计员（3－4天/年）

相关等级

✦无

代码审核 CR3

必须进行全面的代码审核过程，以发现语言级别和特定应用程序的风险

措施

A. 为特定应用程序问题自定义代码分析

代码扫描工具采用了基于知识的内置检测规则，在常用语言的API和库的基础上检查代码。但是，这类工具对于理解自定义API和适用于类似检查的设计能力有限。然而，通过自定义，代码扫描器可成为一个功能强大且通用的分析引擎，以寻找组织和具体项目的安全问题。

就简单性和自定义分析的能力而言，虽然不同工具的细节有所不同，但是自定义代码扫描器通常涉及到在特定的API和对函数调用地方进行指定检查。检查可以包括：分析内部编码标准的遵守、被传递给自定义界面的未检测感染数据、对敏感数据处理的跟踪和验证、内部API的正确使用等。

从使用共享代码库的检查器自定义扫描器是非常有效的，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，安全审计员应检查代码和高层次的设计，以确定可用的检测器，并与开发人员和利益相关者讨论相关的执行。

B. 为代码审核建立发布关卡

为了给所有的软件项目设立代码级的安全底线，应在软件开发生命周期中建立一个特殊的点作为检查点，以满足达到发布的代码审核最低标准。

首先，这个标准应该比较直接，例如，选择一个或两个漏洞类型来设定标准，而具有相应结果的项目不能通过。随着时间的推移，应通过添加额外的通过检查点的标准而改进。

一般来说，代码审核检查点应发生在执行阶段快结束以前，并必须在发布前执行。

对于旧系统或暂未活动的项目，应建立例外处理流程，以允许项目的持续操作，但需要有一个明确的审核时间计划安排以抑制结果。例外应限制于不超过所有项目的20%。

结果

- ✦增加对代码分析的准确性和适用性的信心结果；
- ✦整个组织对安全编码期望的基线；
- ✦项目团队为判断代码级安全设立了目标。

额外成功指标

- ✦>50%的项目使用了代码分析自定义；
- ✦>75%的项目在过去6个月里通过了代码审核审计。

额外成本

- ✦扩充和维护自定义代码审核检查；
- ✦代码审核审计的持续项目开销；
- ✦由未通过代码审核审计，而导致项目延迟的企业组织开支。

额外人员

- ✦架构师（1天/年）
- ✦开发人员（1天/年）
- ✦安全审计员（1－2天/年）
- ✦业务拥有者（1天/年）
- ✦经理（1天/年）

相关等级

- ✦政策与遵守－2
- ✦安全架构－3

安全测试

	ST1	ST2	ST3
目标	根据编程和软件需求，建立处理过程以执行基本的安全测试。	通过自动化使在开发过程中的安全测试更加完善和有效。	在部署前要求进行特定应用程序的安全测试以确保基本的安全。
措施	A. 从已知安全需求推出测试用例； B. 为软件发布执行渗透测试。	A. 使用自动化的安全测试工具； B. 将安全测试整合到开发过程中。	A. 为特定应用程序使用自动化的安全测试； B. 为安全测试建立发布关卡。
评估	✦项目是否明确指定一些基于需求的安全测试？ ✦大多数项目是否在发布以前执行渗透测试？ ✦大多数利益相关者是否意识到需在发布以前进行安全测试？	✦项目是否使用了自动化工具去评估安全测试用例？ ✦大多数项目是否遵守一个一致的过程，为利益相关者评估并记录安全测试？	✦安全测试用例是否为应用程序特定的逻辑而综合制作的？ ✦常规项目审核是否需要来自于安全测试的最低标准结果？
结果	✦独立验证围绕关键业务功能的预期安全机制； ✦为达到安全性测试的高级别审慎调查； ✦对于每个软件项目安全测试的特定完善。	✦更加深入以及更加连贯地验证软件的安全功能； ✦在发布以前开发团队实现自我检查和错误更正； ✦利益相关者在做出风险认同决定时，能更好的意识到开放的威胁漏洞。	✦应用程序对于抵抗预期攻击的表现的组织整体基准线； ✦为提高自动化分析准确性的自定义安全测试组件； ✦项目团队意识到了攻击抵抗的目的。

安全测试 ST1

根据软件编码和软件需求建立为执行基本安全测试的过程

措施

A. 从已知的安全需求中得到测试实例

根据一个项目的已知安全要求，确定一组测试用例，以检查软件功能是否正确。通常，这些测试用例从围绕功能要求和系统业务逻辑的安全问题中获得，但还应包括针对基于编程语言或技术堆栈的常见威胁漏洞的通用测试。

通常，最有效的方法是由项目团队来建立应用程序特定的测试用例，并采用可用的资源或者以购买的知识库，为安全性选择可用的常见测试用例。虽然不是必需的，也可以使用自动化安全测试工具覆盖常见的安全测试用例。

这个测试用例计划应该执行在需求分析和设计阶段，且必须在发布以前的最终测试之前执行。选用的测试用例应由相关的开发人员、安全人员和QA人员针对其适用性、有效性、可行性进行审核。

B. 为软件发布引导进行渗透测试

使用为每个项目选定的安全测试用例进行渗透测试，以评估该系统对于每个用例的表现。这种测试在测试阶段应经常被执行。

渗透测试用例应包括应用程序特定的测试，以检查业务逻辑的完整性；以及一般威胁漏洞测试，以检查设计和程序代码。一旦指定，安全测试用例可以由具有安全常识的质量保证或开发人员执行。但项目小组第一次使用安全测试用例时，应当在安全审计员的监控下进行，以协助并指导团队成员。

在发布或部署之前，利益相关者必须审核安全测试的结果，并在发布时期接受由未通过的安全测试检测显示出的风险。在后一种情况下，一个具体的时间表应当被建立，以弥补相关的漏洞缺陷。

结果

✦独立验证围绕关键业务功能的预期安全机制；
✦为达到安全性测试的高级别审慎调查；
✦对于每个软件项目安全测试的特定完善。

成功指标

✦>50%的项目在过去的12个月里指定了安全测试实例；
✦>50%的利益相关者在过去6个月里大概了解了项目安全测试的状态。

成本

✦扩充或许可安全测试实例；
✦维护和评估安全测试实例的项目持续开销。

人员

✦QA测试员（1－2天/年）
✦安全审计员（1－2天/年）
✦开发人员（1天/年）
✦架构师（1天/年）
✦业务拥有者（1天/年）

相关等级

✦安全需求－1

安全测试 ST2

通过自动化使开发过程中的安全测试更加完善和有效

措施

A. 使用自动化的安全测试工具

为了测试安全性，需要对每个软件界面进行大量输入用例的检测，但这样会使安全用例的有效手动测试实施和执行难以控制。因此，应该使用自动化安全测试工具自动测试软件，以使安全测试更加有效，并获得更高质量的结果。

针对组织的适用性，审核商业的和开源的产品。选择一个合适工具，应基于以下一些因素，其中包括：内置安全测试用例的健康性和准确性；对组织重要的测试架构种类的有效性；改变或添加测试用例的自定义；开发组织结果的质量和可用性等。

在选择过程中，由具有安全知识的技术人员和开发人员、质量保证人员参与介入，并和利益相关者一起审核全部结果。

B. 将安全测试整合到开发过程中

通过使用自动化工具进行安全测试，组织内的项目在开发过程中应当经常执行安全测试并审核结果。为了保持低开销，安全测试工具应该在经常执行的基础上被配置为自动运行，例如：每晚或每周，并检查获得的结果。

只要对需求或者设计有利，就应尽早执行安全测试。虽然在传统意义上，使用了功能性测试用例。但这种测试驱动的开发方法，包含了在开发生命周期的早期阶段（通常是设计阶段），识别并运行安全测试用例。随着安全测试用例自动化的执行，项目在一些针对并不存在的功能而没有通过测试的情况下，进入了编码执行阶段。当所有的测试通过以后，编码执行完成。这在开发生命周期中为开发人员提供了一个清晰的早期目标，从而降低了对于安全考虑而延迟发布的风险，或为赶上项目的最后期限而被迫接受安全漏洞的风险。

对于每个项目的发布，通过自动和手动安全检测方式获得的结果，应当由管理人员和业务利益相关者进行审核。如果在发布内容中仍然有未解决的风险，利益相关者和开发管理人员应共同努力，建立解决这些问题的具体时间表。

结果

✦更加深入以及更加连贯地验证软件的安全功能；
✦在发布以前开发团队实现自我检查和错误更正；
✦利益相关者在做出风险认同决定时，能更好的意识到开放的威胁漏洞。

额外成功指标

✦>50％的项目在过去的6个月里执行了安全测试并由利益相关者认可；
✦>80％的项目在过去1个月里查看了自动化安全测试的结果。

额外成本

✦自动化安全测试方案的研究和选择；
✦自动化整合的初始开支和维护；
✦正在进行的项目执行自动化安全测试和抑制的项目持续开销。

额外人员

✦开发人员（1天/年）
✦架构师（1天/年）
✦经理（1－2天/年）
✦安全审计员（2天/年）
✦QA测试员（3－4天/年）

相关等级

✦

安全测试 ST3

要求应用程序特定的安全测试以在部署以前确保安全的基准线

措施

A. 为特定应用程序使用自动化的安全测试

通过自定义安全测试工具、增强通用测试用例执行工具或者通过扩充自定义用户测试工具，项目小组通过安全要求建立一组自动的检查器，开始正式迭代工作以测试所执行业务逻辑的安全。

另外，如果对自动安全测试工具进行自定义，以使它们能够理解关于正在进行测试的项目中的特定软件界面的更多信息，就可以明显改善测试的精度和覆盖深度。还有，可将特定于组织的规范或技术标准的特定问题，整理为一个可重复使用、集中的测试系列，以使能够更容易地查看审计数据集合和每个项目的管理情况。

项目小组应在他们的软件业务功能基础上，着重于安全测试用例的建立；另外，一个由安全审计员领导的组织级别团队，应着重于详细阐述对符合规定和内部标准的自动化测试。

B. 为安全测试建立发布关卡

为了避免在即将发布的软件中存在极易被发现的安全bug，应在软件开发生命周期中设立一个的特定检测点，已建立的安全测试用例必须通过该点才能发布项目。这就建立了一个所有项目都应该通过的各种安全测试的基准。

由于在初期添加太多的测试用例可能会导致一个间接成本增加，因此，在开始时选择一到两个安全问题，包括对每种问题的各种测试用例，并以项目必须通过所有的测试为期望。随着时间的推移，这个基准应随着额外安全问题的选择以及添加各种相应测试用例而被改善。

总的来说，这个安全测试检测点应在编程阶段快结束时或是测试阶段执行，且必须在发布以前。

对于旧系统或暂未活动的项目，应建立例外处理流程，以允许项目的持续操作，但需要有一个明确的审核时间计划安排以抑制结果。例外应限制于不超过所有项目的20%。

结果

✦应用程序对于抵抗预期攻击的表现的组织整体基准线；
✦为提高自动化分析准确性的自定义安全测试组件；
✦项目团队意识到了攻击抵抗的目的。

额外成功指标

✦>50%的项目正在使用自定义的安全测试；
✦>75%的项目在过去6个月里通过了所有的安全测试。

额外成本

✦为自动化安全测试而扩充和维护自定义；
✦安全测试审计过程的项目持续开支；
✦由于安全测试审计失败而导致项目延迟的组织开支。

额外人员

✦架构师（1天/年）
✦开发人员（1天/年）
✦安全审计员（2天/年）
✦QA测试员（1－2天/年）
✦业务拥有者（1天、年）
✦经理（1天/年）

相关等级

✦政策于遵守－2
✦安全架构－3

漏洞管理

	VM1	VM2	VM3
目标	理解对于漏洞报告或事件的高级别计划。	为响应过程阐述期望，以改善一致性和交流。	在响应过程中为积极规划提供反馈，而改善分析和数据收集。
措施	A. 为安全事件确定联络点； B. 建立非正式安全响应团队。	A. 建立一致的事件响应流程； B. 采用安全事件报告流程。	A. 为事件执行根源分析； B. 收集每一事件的度量指标。
评估	✦大多数项目是否都有对于安全问题的联络点? ✦组织是否已设立一个安全响应团队？ ✦大多数项目团队是否知道他们的安全联络点和响应团队?	✦组织对于事件的报告和处理是否使用了一种一致的流程？ ✦大多数利益相关者是否意识到公开的相关安全事件与他们的软件项目有关系?	✦大多数调查了根本起因的安全事件是否有了更多更深入的建议？ ✦大多数项目是否持续收集并报告关于安全事件的数据和衡量标准?
结果	✦处理高优先级漏洞或事件的简便处理过程就为； ✦供利益相关者通知和报告影响安全的事件的框架； ✦处理安全事件的高级审慎调查。	✦处理第三方漏洞报告的沟通计划； ✦为软件操作员发布安全补丁的清晰流程； ✦关于事件跟踪、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； ✦对由漏洞和损坏带来成本损失的粗略估计； ✦利益相关方能够更好地根据历史事件趋势做出权衡的决策。

漏洞管理 VM1

理解对于漏洞报告或事件的高级别计划

措施

A. 为安全事件确定联络点

为组织中的每一个部门或每个项目团队建立一个联络点，以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监管。

由安全事件造成的使用示例包括:接收来自外部实体的漏洞报告、现场发生的软件损坏或其他软件的安全故障、内部发现的高风险漏洞等。如果发生安全事件，则最近的联络点将作为受影响项目团队的额外资源以及顾问，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。这些分配的安全联络点的名单列表应集中维护，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能够就安全问题更直接地向其他成员请求帮助并进行合作。

B. 建立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，选出一小组人，以作为一个集中式的安全响应技术团队。该团队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。

除具有上述职责外，安全响应团队的成员还需要负责在事件发生期间执行行政报告并与上级沟通。在大多数情况下，安全响应团队可能不会执行这些职责，但是他们必须具有足够的灵活性以能够快速响应，或必须与其他团队组员一起流畅地处理安全事件。

响应团队应每年至少举行一次会议，以简要介绍响应处理流程的安全联络点，以及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处理过程就为;
✦供利益相关者通知和报告影响安全的事件的框架;
✦处理安全事件的高级审慎调查。

成功指标

✦>50%的组织在过去6个月中了解了最近的安全联络点;
✦在过去12个月中，安全响应团队和联络点举行了至少一次会议。

成本

✦项目中由人员担任安全联络点角色而造成的持续开销;
✦确定恰当的安全响应团队。

人员

✦安全审计员（1 天/年）
✦架构师（1 天/年）
✦经理（1 天/年）
✦企业所有者（1 天/年）

相关等级

✦教育与指导－2
✦策略与指标－3

漏洞管理 VM2

为响应过程阐述期望，以改善一致性和交流

措施

A. 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处理过程。此外，每个安全响应团队的成员必须每年进行至少一次针对这些材料的培训。

合理的事件响应流程需具有几条原则，它们包括：初始分类以避免额外的损失、更改管理和补丁应用程序、管理项目人员以及与事件相关的其他人员、证据收集和保存、限制将该事件透露给利益相关者、明确的向利益相关者报告机制和通信树等。

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，当项目团队检测到安全事件或高风险漏洞时，他们应遵循一个内部流程，从而与安全响应小组的成员取得联系。

B. 采用安全事件报告流程

对于大多数组织而言，它们不愿意公布关于安全问题的消息，但这里有一些应遵循的重要方法，以将安全问题从内部沟通到外部。

第一个，也是最常见的一个方法，是组织通过为其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，则这种方法不那么重要；但是对于在组织以外操作该软件的所有环境而言，必须存在补丁发布流程。它应该提供多个因素，包括：发布补丁前的变更管理和回归测试、通知操作员/用户为补丁分配关键性类别、分散详细的技术信息从而使攻击者无法直接加以利用，等等。

另一个方法是与第三方进行外部沟通，以报告组织软件中的安全漏洞。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信息公开步骤实践。

最后，许多国家/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以及组织利益相关者合作，确定下一步适当的操作。

结果

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

额外成功指标

✦ >80%的项目组在过去六个月中大概了解了事件响应流程；
✦ >80%的利益相关者在过去6个月中大概了解了安全问题的公开情况。

额外成本

✦ 当前组织对于安全事件响应流程的开销。

额外人员

✦安全审计员（3-5天/年）
✦经理（1-2天/年）
✦业务拥有者（1-2天/年）
✦支持/操作人员（1—2天/年）

相关等级

✦

漏洞管理 VM3

在响应过程中为积极规划提供反馈，而改善分析和数据收集

措施

A. 为事件执行根源分析

虽然可能会花费大量时间，但应当将安全事件响应处理流程扩大以进行额外的分析，从而确定导致安全故障的关键。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：社会工程学、未能遵循程序等）。

一旦确定事件根源后，应将其作为工具使用，以在组织中查找可能发生类似事件的其他潜在缺陷。对于每一个已确认的缺陷，还应该给出其他预先主动抑制发生的建议，以作为结束原始事件响应工作的一部分。

所有根据造成事件的根本原因分析所得出的建议，应由管理层和相关利益相关者审查，以安排相应的牵制工作或记录所能接受的风险。

B. 收集每一事件的度量指标

通过采用集中处理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所提出的影响和效率。

应记录并审核过去至少每6个月中的安全事件信息。对相似的事件进行分组，并简单统计每种类型事件的总数。对于事件采取的额外措施还因包括：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、处理和清除安全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。对于造成技术故障的根本问题，确定哪种类型的主动措施、审核或操作实践，以对最初检测到安全事件或减轻其损坏也是很有帮助的。

此信息是对程序计划处理流程的具体反馈，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；
✦对由漏洞和损坏带来成本损失的粗略估计；
✦利益相关方能够更好地根据历史事件趋势做出权衡的决策。

额外成功衡量标准

✦>80%的事件在过去6个月中记录了根本起因和进一步的建议；
✦>80％事件过去6个月中对度量指标进行了整理。

额外成本

✦组织对安全事件持续执行深入研究和分析而造成的开销。
✦组织对安全事件度量标准的持续收集和审核而造成的开销。

额外人员

✦安全审计员（3天/年）
✦经理（2天/年）
✦业务拥有者（2天/年）

相关等级

✦政策与度量标准－3

环境强化

	EH1	EH2	EH3
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。	以已知最佳实践验证应用程序的健康和操作环境状态。
措施	A. 维护操作环境说明； B. 确定并安装关键的安全软件升级和补丁。	A. 建立常规补丁管理流程； B. 监控基准基础架构配置状态。	A. 确定并部署相关操作的保护工具； B. 为环境配置扩展审计计划。
评估	✦大多数项目是否为操作环境记录需求？ ✦大多数项目是否检查第三方软件部件的安全更新？	✦是否使用一个持续连贯的处理流程以对于关键依赖的部件使用安全更新和补丁？ ✦大多数项目是否权衡了应用程序的自动化检测和环境的健康状态？	✦利益相关方是否意识到当进行操作时用于保护软件的额外工具的选择？ ✦常规审计是否为大多数的项目检查环境健康状况？
结果	✦开发团队清楚理解了操作预期； ✦根据一个已充分理解的时间表，削减现有基础架构下的高优先级风险； ✦软件操作人员对于基础架构的关键安全维护而配备的高级别计划。	✦对操作的系统中的安全特性进行了粒度验证； ✦正式预测对于削减基础架构风险时间表； ✦利益相关者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； ✦建立并测量了操作维护和操作性能目标； ✦通过外部依赖性的缺陷降低了成功攻击的可能性。

环境强化 EH1

了解应用程序和软件组件的基本操作环境

措施

A. 维护操作环境规范说明

对于每一个项目，应创建并维护一个预期操作平台的具体定义。根据组织的组成，此规范说明应由开发人员、利益相关者、支持和操作团队等共同创建。

开始创建该规范应首先根据软件的商业功能获得有关此操作环境的所有真实详细信息。这些信息包括以下因素，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，还需记录影响软件行为方式的关于操作环境的任何已知用户或操作人员配置选项。

另外，确定在项目设计和实施阶段中提出的有关操作环境的假设，并在规范说明中使用这些假设。

对于正在进行的项目，应至少每6个月对该规范说明进行审核和更新。如果软件设计或预期的操作环境发生了更改，则应更频繁地执行审核和更新。

B. 确定并安装关键的安全更新和补丁

大多数应用程序运行于另一大型软件堆栈之上，这些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。因为这个大型软件堆栈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，所以必须安装技术堆栈模块的重要安全更新。

同样，应执行对高风险依赖性的常规研究和持续监控，以保证安全漏洞安装了最新补丁。确定可影响软件项目安全状态的重要更新和补丁后，应制定相应计划，安排受影响的用户和操作人员安装更新。根据软件项目的类型，此操作的具体步骤和内容可能不同。

结果

✦开发团队清楚理解了操作预期；
✦根据一个已充分理解的时间表，削减现有基础架构下的高优先级风险；
✦软件操作人员对于基础架构的关键安全维护而配备的高级别计划。

成功指标

✦>50%的项目在过去6个月中更新了操作环境规范说明；
✦>50%的项目在过去6个月中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说明的项目开销；
✦持续监控并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）
✦经理（2－4 天/年）
✦支持/操作人员（3－4 天/年）

相关等级

✦操作实现－2

环境强化 EH2

通过强化操作环境提高对应用程序操作的信心

措施

A. 建立常用补丁管理流程

从重要更新和补丁的专用应用程序转移到一个更加正式的专门处理流程，应在组织中创建一个持续的流程，以在操作环境中对软件基础架构持续使用安全补丁。

在最基本的形式上，此处理流程致力于针对安全补丁的发布和应用时间间隔做出保证。为使此处理流程高效地工作，通常组织对低优先级补丁接受高延迟，例如：低优先级补丁的期限最多可以为30天，而重要补丁的延迟最多仅为2天。

这项工作应主要由支持和操作人员执行，但是应当与开发团队召开例会，以使项目与过去的更改和预定的升级更新保持一致。

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控这些组件，并提示开发团队在何时需要进行升级。

B. 监控基准环境配置状态

对于一个软件项目的基础架构，由于监控和管理各种组件的补丁很复杂，应利用自动化工具来自动监控系统，以保证配置的有效性。

很多商业工具和开源工具可提供这种类型的功能，因此，项目团队应选择一个适合的基于组织需要的解决方案。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、度量标准的收集和趋势追踪等等。

除对主机和平台检测以外，应对监控自动化进行自定义，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，以共同确定一个给定软件项目的最佳监控数量。

最后，当部署了一个用于监控基础架构的配置状态解决方案后，应由项目利益相关者在每周或者至少每季度对意外警报或配置更改，进行一次收集和审核。

结果

✦对操作的系统中的安全特性进行了粒度验证；
✦正式预测对于削减基础架构风险时间表；
✦利益相关者持续关注软件项目的当前操作状态。

额外成功指标

✦ >80%的项目团队在过去12个月中了解了补丁管理处理流程。
✦ >80%的利益相关方在过去6个月中意识到了当前补丁状态。

额外成本

✦组织进行补丁管理和监视的持续开销；
✦扩充和许可基础架构监控工具。

额外人员

✦架构师（1－2天/年）
✦开发人员（1－2天/年）
✦业务拥有者（1－2天/年）
✦经理（1－2天/年）
✦支持/操作人员（3－4天/年）

相关等级

✦

环境强化 EH3

以已知最佳实践验证应用程序的健康和操作环境状态

措施

A. 确定并部署相关操作保护工具

为了在软件的操作环境中为软件构建一个更好的保证用例，可使用额外的工具来增强系统的整体安全状态。不同的操作环境会有很大的不同，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、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、证据调查的日志聚合工具、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，技术利益相关者应与支持和操作人员合作，为业务利益相关者确定和推荐选定的操作保护工具。如果在降低风险对比实施成本方面认为一项投资有价值，则利益相关者应同意试进行、广泛展示和持续维护的计划。

B. 为环境配置扩展审计程序

当执行例行的项目级审计时扩展审核，以纳入对与强化操作环境相关的工件的检查。除了操作环境的最新说明以外，审计过程还应包括对于当前补丁状态和自上次审计的历史数据的检查。通过接入监控工具，审计过程还可以验证有关应用程序配置管理和历史更改的关键因素。审计过程还应当针对该软件架构类型可用的操作保护工具，检查这些工具的使用情况。

基础架构的审计可在项目初始发布和部署后的任何时间点进行，但应至少每6个月进行一次。对于旧系统或无开发活动的项目，基础架构审计仍应由业务利益相关者执行并审核。一个例外处理流程应被创建，以允许有特殊案例的项目继续进行，但必须为抑制结果处理明确地分配时间计划。例外应限制在不超过所有项目的20%。

结果

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

额外成功指标

✦ >80%的利益相关方在过去6个月中了解了相关操作保护工具；
✦ >75%的项目在过去6个月中通过了基础架构审计。

额外成本

✦ 研究和选择操作保护解决计划；
✦ 操作保护工具的建立和许可；
✦ 维护保护工具的持续操作开销；
✦ 关于基础架构审计的持续项目开销。

额外人员

✦ 业务拥有者（1天/年）
✦ 经理（1－2天/年）
✦ 支持/操作人员（3－4天/年）

相关等级

✦ 政策与遵守－2

操作实现

	OE1	OE2	OE3
目标	实现开发团队和操作人员之间对于与安全相关的关键数据的沟通交流。	通过提供详细的步骤为持续的安全操作提高期望。	针对完整性而对安全信息和部件检验进行强制宣传。
措施	A. 为部署获得的重要的安全信息； B. 为典型的应用程序警报记录流程。	A. 创建每次发布的变更管理流程； B. 维护正式的操作安全指南。	A. 为操作信息扩展操作审计计划； B. 对应用程序组件执行代码签名。
评估	✦你是否递交了关于多数软件发布的安全记录？✦大多数项目是否都记录了与安全相关的警报和产生错误的条件？	✦大多数项目是否正在使用一个已理解的变更管理处理流程？ ✦项目团队是否为每个软件发布递交了一份操作安全手册？	✦大多数项目是否为了恰当的操作安全信息，而去审计检查每个发布？ ✦软件组件的代码签名是否按照一致的处理流程被经常进行？
结果	✦通过更好地理解正确操作，为软件安全状态进行特定的改进； ✦操作人员和用户意识到各自为确保安全部署的职责； ✦对于关键的安全信息，改善了软件开发人员和用户之间的交流。	✦针对与安全相关的更新，随软件发布而提供的详细指南； ✦每个应用程序为安全操作步骤更新了的信息库； ✦对开发人员、操作人员和用户的操作预期保持一致。	✦整个组织都理解了对于与安全相关文档的期望； ✦利益相关者能根据部署和操作的反馈，做出更好的权衡决策； ✦操作人员和/或用户能够独立验证软件发布的完整性。

操作实现 OE1

实现开发团队和操作人员之间对于与安全相关的关键数据的沟通交流

措施

A. 为部署操作捕获重要的安全信息

利用特定软件的知识，项目团队应当确认与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的软件内置安全特征列表开始。根据该列表，还应获得关于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，以更好地让用户和操作人员知道选择后所带来的影响。

整体而言，此列表应该言简意赅，并旨在提供最关键的信息。一旦初始建立列表后，为取得同意，应由项目团队和组织利益相关方进行审核。另外，为确保可理解和操作相关信息，与选定的操作人员和用户共同审核该列表会非常高效。项目团队应对每个发布和更新审核该信息，且必须至少每6个月进行一次。

B. 为典型的应用程序警报记录过程

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，应明确并获得用户/操作人员为响应此事件而采取适当操作的信息。

对于软件可能产生的很大一组事件，就软件的业务目标而言，选择优先级最高的集合。该集合应该包括与安全相关的任何事件，但也可能包含与软件运行健康状况和配置状态相关的关键错误和警报。

为每一个事件捕获可操作的建议，以为通知用户和操作人员所需的下一步操作和导致事件的潜在根本原因。这些处理流程必须由项目团队审核，并每6个月为每个主要的产品发布更新一次，但也可以进行得更频繁，例如在每次发布时。

结果

✦通过更好地理解正确操作，为软件安全状态进行特定的改进；
✦操作人员和用户意识到各自为确保安全部署的职责；
✦对于关键的安全信息，改善了软件开发人员和用户之间的交流。

成功指标

✦>50%的项目在过去6个月中更新了安全信息部署；
✦>50%的项目在过去6个月中更新了事件的操作流程。

成本

✦来自维护安全信息部署的持续项目开销；
✦来自维护关键操作流程的持续项目开销。

人员

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

相关等级

✦

操作实现 OE2

通过提供详细的步骤为持续的安全操作提高期望

措施

A. 创建每次发布变更的管理过程

为了在软件中正式地更新与用户和操作人员相关的改变，每个发布版本都必须包括与升级和首次安装相关联的变更管理处理流程程序。总而言之，目标是确定预期的随附步骤，以确保部署成功，且不会导致过多的停机时间或安全状态的下降。

要在开发过程中建立这些流程，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。在开发周期的早期使用此流程会非常高效，这样可将这些信息保留，并在需求分析、设计和编码阶段时确定。

在每次发布前，项目团队应将此列表作为一个整体进行审核，以检查完整性和可行性。对于一些项目，一个给定的版本发布伴随着大量的变更流程，对这些变更可进行授权特殊处理，例如构建自动升级脚本以避免部署中发生错误。

B. 维护正式的操作安全指南

从获得的关键软件事件信息以及处理每个事件的流程开始，项目团队应建立并维护正式的指南，以捕获用户和操作人员需要了解到的所有的与安全相关的信息。

最初，应根据有关系统的已知信息建立该指南，这些信息包括：与安全相关的配置选项、事件处理流程、安装和升级指南、操作环境规范说明、有关部署环境的安全相关假设，等等。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，这样大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，这可能会很挑战，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部署的建议。

操作安全性指南初始创建后，应由项目团队进行审核，并随每次发布进行更新。

结果

✦针对与安全相关的更新，随软件发布而提供的详细指南；
✦每个应用程序为安全操作步骤更新了的信息库；
✦对开发人员、操作人员和用户的操作预期保持一致。

额外成功指标

✦>50%的项目在过去6个月中更新了更改管理处理流程；
✦>80%的利益相关方在过去6个月中大概了解了操作安全指南的状态。

额外成本

✦维护改变管理程序的持续项目开销；
✦维护操作安全指南的持续项目开销。

额外人员

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

相关等级

✦环境强化－1

操作实现 OE3

针对完整性而对安全信息和部件检验进行强制宣传

措施

A. 为操作信息扩展审计程序

当执行常规的项目级审计时扩展审核，以包括对涉及安全性操作实现组件的检查。应对项目进行检查，以确保其具有涉及软件细节的已更新并完整的操作安全指南。

这些审计应在开发周期快要结束并接近发布的时候开始进行，但是必须在发布前完成并通过。对于旧系统或不活动的项目，这种审计应当被执行，完成一次性的解决结果，并验证审核标准遵守，随后则不再需要针对操作实现进行另外的审核。

发布前，必须由利益相关者对审计结果进行审核。一个例外流程应当被创建，以允许未通过审计的项目继续进行发布，但是这些项目应具有修正错误的具体时间期限。例外应限制在不超过所有活动项目的20%。

B. 为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，代码签名允许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发布版本的真实性。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，从而避免已部署的软件在其操作环境中被破坏或修改。

对代码进行签名会造成组织管理签名凭证的开销。一个组织必须遵循安全的密钥管理流程，以确保签名密钥的持续保密性。当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的计划，例如：密钥循环、密钥泄露或密钥丢失。

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计员和组织利益相关方合作，确定应进行签名的软件部件。随着项目的发展，每次发布都应对此列表进行审核，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

✦整个组织都理解了对于与安全相关文档的期望；
✦利益相关者能根据部署和操作的反馈，做出更好的权衡决策；
✦操作人员和/或用户能够独立验证软件发布的完整性。

额外成功指标

✦>80%的项目在过去6个月中更新了操作安全指南；
✦>80%的利益相关方在过去6个月中大致了解了代码签名选项和状态。

额外成本

✦操作指南审计的持续项目开销；
✦维护代码签名凭证的持续组织开销；
✦代码模块确定和签名的持续项目开销。

额外人员

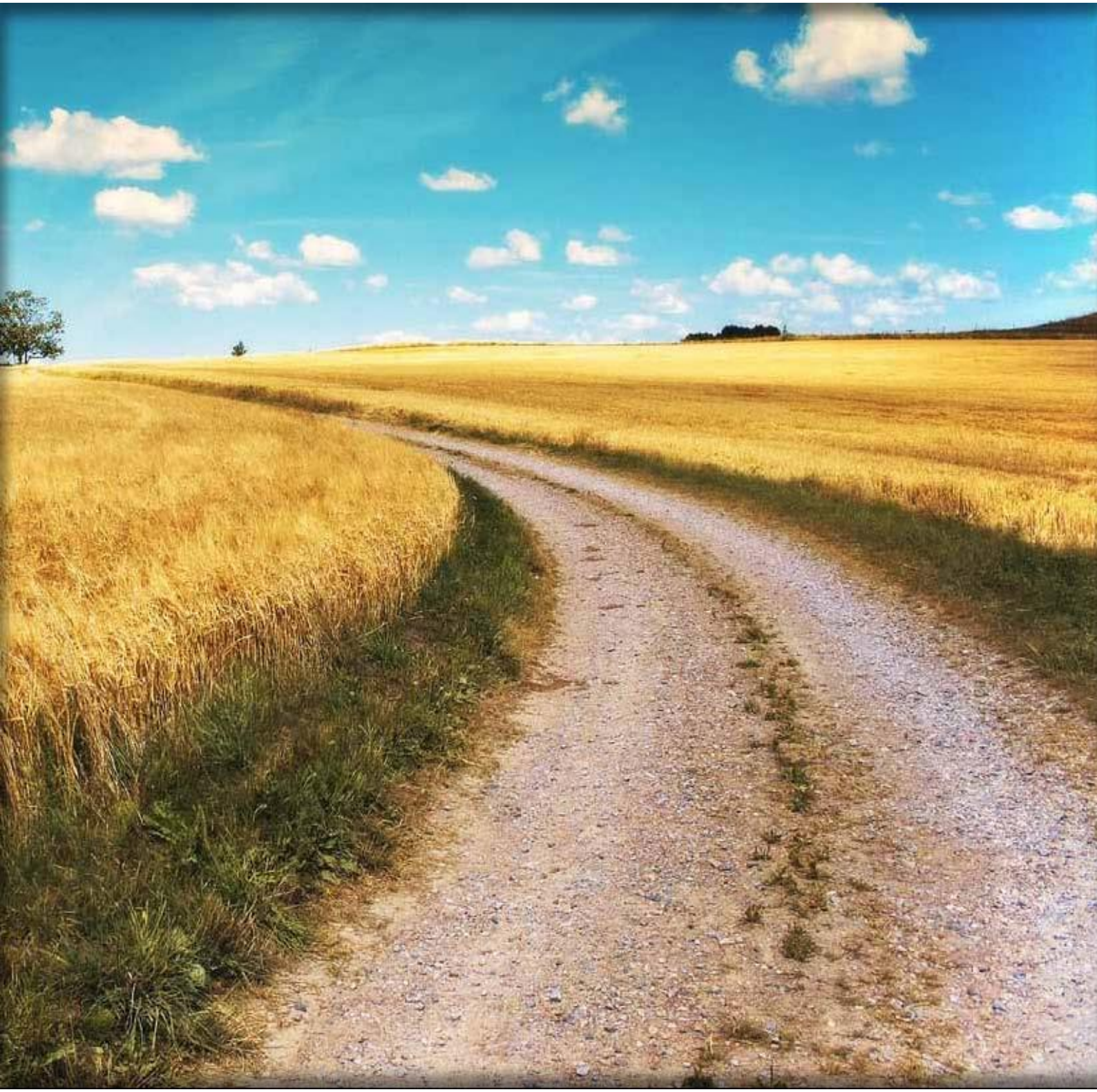
✦开发人员（1天/年）
✦架构师（1天/年）
✦经理（1天/年）
✦安全审计员（1－2天/年）

相关等级

✦

案例分析

对一个案例进行从头到尾的讲解



本节详细分析了一个使用SAMM的特定商业案例。该案例描述了组织在建立安全保证方案时，如何将路线图模版作为指导，适应可能的最佳实践，并考虑组织的特定风险。

VirtualWare

案例分析：中型独立软件供应商

业务概况

VirtualWare是其所在市场的主导者，专门提供虚拟化的综合应用平台，以帮助企业将他们的应用程序界面整合到一个单一的环境中。他们的技术为多个操作环境提供了服务器应用程序和内置的桌面客户端，其中包括微软、苹果和Linux平台。

该组织是一个中型组织企业（200-1000名员工），并已在世界各主要国家设立了分公司以开展全球业务。

组织

VirtualWare已使用了超过8年的时间一直在开发自己的核心软件平台。在此期间，由于网页界面较少被使用，他们仅需面对少量的普通Web漏洞风险。大多数VirtualWare的平台运行在一个基于服务器的系统或者运行在桌面上的客户端。

最近，VirtualWare开始了一系列的新项目，以通过Web技术实现他们的客户端和服务器界面。通过了解有关Web的常见攻击程度，使该组织审核其软件的安全战略并确保充分解决可能存在的威胁，以推动组织向前发展。

此前，该组织已开展应用程序代码的基本审核，但更加注重于性能和功能，而非安全性。VirtualWare开发人员已使用一些代码质量分析工具来发现错误并在代码中解决它们。

考虑到这一点，高层管理团队已经制定了一个战略目标，以审核他们应用程序目前的安全状况，并在其中识别、去除、预防漏洞的最好方法。

环境

VirtualWare使用混合的Java、C++和Microsoft. NET技术开发他们的虚拟化技术。他们的核心应用虚拟化技术已用C++编写，并已有了针对错误和安全性的一系列审核，但目前还没有正式的处理流程已识别和修复已知或未知的安全错误。

虽然后端系统使用了微软和C++技术，VirtualWare选择了Java以支持他们的Web技术。开发团队专注于的网络界面则主要由Java开发人员编制。

VirtualWare雇用了超过300名开发人员，在他们工作的项目基础上，与其他工作人员分成各个团队。共有12支团队，每队大约有20-40名开发人员。每个团队关于软件安全性都仅有很少的经验，虽然高级开发人员对于代码执行了基本的评估，但是安全性并没有作为组织的一个重要目标。

VirtualWare中的每个团队采用了不同的开发模型。目前使用的两个主要方法是Agile SCRUM和迭代瀑布方法。对于软件安全性，IT部门或项目架构师都没有指导意见。

核心挑战

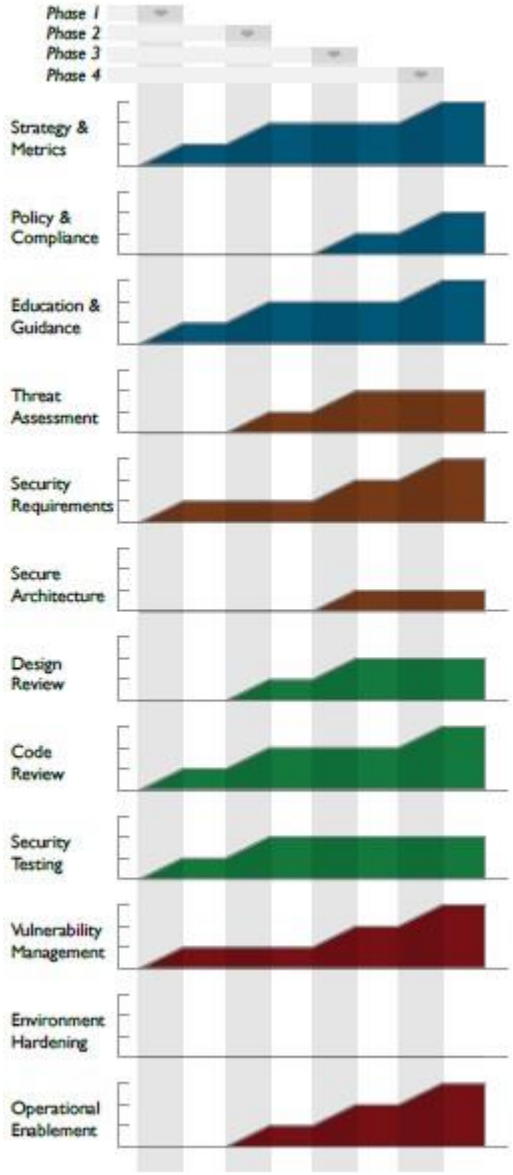
✦应用程序功能的快速发布，以确保保持对他们竞争对手的竞争优势；
✦有关软件安全性概念的有限经验—目前仅有很少的活动任务是与安全相关的；
✦有经验的开发人员离开该组织，并由缺乏经验的开发人员取代；
✦多种技术被用于应用程序，而旧的应用程序自初始建立以后没有被更新；
✦不了解现有的安全情形或组织正面临的风险。

VirtualWare希望把重点放在如何为客户安全地开发新Web应用程序。因此，对于实施安全保证计划的最初重点在于对他们的开发团队进行教育和意识培养，以及提供一些关于安全编码的基础技术指导和测试标准。

该组织此前曾通过他们的电子邮箱地址：support@virtualware.net，收到了来自客户关于获得错误和安全漏洞信息的请求邮件。但是，由于这只是一个普通的支持邮箱地址，现有的请求在组织内并不是总能传递到正确的团队并得到正确处理。另外，VirtualWare还明确了实施一个正式的安全漏洞响应计划的需求。

执行策略

在一个组织内执行一项安全保证计划是一个长期的策略，不仅对开发人员的文化会带来显着的影响，而且会影响对于开发并交付应用程序采取的企业处理流程。这一策略的执行被设置在12个月内完成，但由于该组织的规模，在这一指定时间内将会比较容易地实施。



第一阶段（第1至3个月）— 意识培养和规划

VirtualWare先前已发现，他们组织对于应用程序安全性威胁仅有有限的知识和意识，以及有限的安全编码经验。在VirtualWare中部署计划的第一阶段集中于对开发人员的培训，并执行相关指导和计划，以确定当前的安全漏洞。

VirtualWare的开发团队在安全编码技术方面仅有有限的安全经验，因此，开发的初始培训计划可以是向组织内的开发人员提供防御性的编程技术。

在组织内有超过300名的开发人员并支持多种开发语言，对于VirtualWare关键的挑战之一是提供一个教育课程，从纯技术的角度，向开发人员讲授一些基本的安全编码概念。这个初始教育课程的主要目的在于编码技术和测试工具。组织并进行为期一天的课程，以讲授涵盖安全编码的基础知识。

VirtualWare意识到，他们的很多应用程序都有漏洞，并且缺少真正的策略，用以明确已有的漏洞，并在一个合理的时间框架内解决这些问题。一个基本的风险评估方法已被实施，该组织并对于现有的应用程序平台进行了审核。

这一阶段还包括执行一组概念以强化开发团队的安全工具。开发团队已经有了许多工具用于执行质量类型的评估。对于代码审核和安全测试工具进行的额外调查也已被执行。

预期目标

在该项目的这一阶段，VirtualWare执行了以下SAMM的实践与措施。

SM1	A. 评估整体业务风险概况； B. 建立并维护保证计划路线图。
EG1	A. 实施技术安全意识的培训； B. 建立并维护技术指导。
SR1	A. 从业务功能推导出安全需求； B. 为需求评估安全和遵守指导。
CR1	A. 根据已知安全需求建立审核检查列表； B. 为高风险代码执行定点审核。
ST1	A. 从已知安全需求推出测试用例； B. 为软件发布执行渗透测试。
VM1	A. 为安全事件确定联络点； B. 建立非正式安全响应团队。

为了实现这些成熟度等级，VirtualWare在这一阶段实施了一系列活动计划。下列措施被执行：

✦为所有开发人员进行1天的（高级别的）安全编码课程；

✦建立一个为在组织内所使用技术的应用程序安全的技术指导白皮书；

✦建立一个风险处理流程，并为应用程序平台执行高级别的商业风险评估和业务风险审核；

✦为开发人员准备初步的技术指导和标准；

✦为组织目前的重大风险在应用程序平台上执行短代码审核；

✦为项目开发测试和使用用例，并为应用程序评估用例；

✦为应用程序安全初始化任命一个角色；

✦为下一阶段的保证计划产生一个战略路线图的草案。

由于VirtualWare内部对于专业知识技术的匮乏，公司与第三方安全咨询公司展开合作，以协助建立培训计划，并协助编写组织的威胁模型和战略路线图。

在这一阶段所面临的主要挑战之一，就是让所有300名开发人员通过为期一天的培训课程。为了实现这一目标，VirtualWare举行了20天的课程，每次只有每个开发团队的少数人员参加课程。这减少了在培训期间对人力资源的整体影响。

在项目的这一阶段，VirtualWare投入了大量的资源到风险审核处理过程和组织的业务风险审核过程中。虽然为这些任务付出了相当大的努力，但是必须确保VirtualWare执行的下一步骤，是与企业组织所面临的风险联接在一起的。

VirtualWare的管理层收到了积极的反馈信息，其中大多数来自参加了培训项目的组织内部开发人员。虽然培训课程没有提供足够详细的信息，但是开发人员认为，该初始培训已提供了一些可以帮助他们立即在每天编写安全代码的基本技能。

执行成本

在项目的这个阶段投入了大量的内部资源和成本。这一阶段的相关成本有三种不同类型。

内部资源需求

内部资源使用在该阶段应用程序安全初始活动的内容创建、研讨会组织和审核之中。资源使用以每个角色总天数的方式展现。

开发人员	14天	业务拥有者	8天
架构师	10天	QA测试人员	3天
经理	8天	安全审计员	9天

培训资源需求（该阶段每人的培训）

VirtualWare内的每个开发人员要求参加一个培训课程，因此，每一个开发人员分配到了为期一天的应用程序安全计划。

开发人员（每人） 1天

外包资源

由于VirtualWare内部知识的匮乏，外部资源被用来协助内容创建，以及为开发人员创建/提供培训计划。

安全顾问	15天	培训顾问	22天

第二阶段（第3至6个月）— 培训和测试

VirtualWare在第一阶段确定，他们的很多应用程序都包含了可能受到来自外部威胁的漏洞。因此，这一阶段的主要目标之一是进行基本能力测试和审核，以在代码中确定漏洞并解决它们。

在这一实施阶段，最大的挑战之一是引进自动化工具以尽可能多得覆盖代码并发现弱点。在过去，开发人员对于使用自动化工具非常困难，因此，使用新的工具被视作为一个巨大的挑战。

为了确保在组织内成功推出自动化工具，VirtualWare开始分阶段的进行。这些工具将首先提供给团队的高级领导人，并在一段时间内在网上向开发人员发布。团队被鼓励使用这些工具，但是，对于他们的使用没有进行正式的处理。

这一实施阶段也被看作为一个更加正式的教育和安全意识培训计划。开发人员被要求在Web服务和数据验证方面进行相比于前次培训更加具体的培训。为期六个小时的具体培训新课程在这两个重点领域被制作。VirtualWare还为架构师和经理提供了的额外训练课程，并在通过组织内举行了一个宣传活动。

预期目标

在该项目的这一阶段，VirtualWare执行了以下SAMM的实践与措施。

SM2	A. 根据业务风险将数据和应用程序分类； B. 建立并衡量每个分组的安全目的。
EG2	A. 实施针对特定角色的应用程序安全培训； B. 聘用安全指导专家增强项目团队。
TA1	A. 建立并维护特定应用程序的威胁模型； B. 根据软件架构建立攻击者概况。
DR1	A. 确定软件攻击层面； B. 根据已知安全需求分析设计。
CR2	A. 使用自动化的代码分析工具； B. 将代码分析集成到开发流程当中。
ST2	A. 使用自动化的安全测试工具； B. 将安全测试整合到开发过程中。
OE1	A. 为部署获得的重要的安全信息； B. 为典型的应用程序警报记录流程。

为了实现这些成熟度等级，VirtualWare在这一阶段实施了一系列活动计划。下列措施被执行：

✦为质量保证测试人员、经理和架构师提供额外的教育及培训课程；

✦将数据资产进行分类，并设置安全目标；

✦将威胁评估方法开发成一种利用攻击树和档案的威胁建模方法；

✦审核并确定每个应用程序平台的安全需求；

✦介绍自动化工具以协助代码覆盖和现有应用程序以及新代码库的安全分析；

✦审核并强化现有的渗透测试计划；

✦加强现有的软件开发生命周期，以支持安全测试作为开发处理过程的一部分。

VirtualWare调整了现有的应用程序安全培训计划，以提供一个较小的技术版本，作为一个商业应用程序的安全意识计划。这是一个只有4个小时的短课程，并将培训人员扩大到组织的经理和业务拥有者。

对现有的代码审核和渗透测试计划的一个高级别审核，确定该处理过程不足，并需要得到加强，以为应用程序安全漏洞提供更好的测试和结果。一个团队已着手实施执行渗透测试和代码审核的新计划。作为该计划的一部分，每个项目团队的每个高级开发人员被分配大约4天时间对于他们的应用程序去执行高级别的源代码审核。

VirtualWare的管理层了解到，基础设施和应用程序紧密集成在一起，并在这一阶段，对应用程序平台（基础设施）的操作方面进行了审核。这个阶段的重点在于基础设施需求，以及在推荐部署的硬件和应用程序接口之间的应用程序集成特征。

在这一阶段，战略路线图和应用程序安全的方法由项目团队进行了审核。本次审核和更新的目的是对数据资产进行正式地分类，并设置与数据资产和应用程序相关业务风险的适当等级。因此，项目团队能够为这些应用程序设定安全目标。

执行成本

在项目的这个阶段投入了大量的内部资源和成本。这一阶段的相关成本有三种不同类型。

内部资源需求

内部资源使用在该阶段应用程序安全初始活动的内容创建、研讨会组织和审核之中。资源使用以每个角色总天数的方式展现。

开发人员	8天	业务拥有者	5天
架构师	10天	QA测试员	3天
经理	8天	安全审计员	15天
支持操作	2天		

培训资源需求（该阶段每人的培训）

更多VirtualWare内的人员被要求参加这一培训课程，因此，不同的人员角色被分配到了时间以参见应用程序安全的培训。

架构师（每人）	1天	经理（每人）	1/2天
业务拥有者（每人）	1/2天		

外包资源

由于VirtualWare内部知识的匮乏，外部资源被用来协助内容创建，以及为开发人员创建/提供培训计划。

安全顾问	22天	培训顾问	5天

第三阶段（第6至9个月）－ 架构和基础设施

在VirtualWare内实施保证计划的第三阶段，建立于前面的实施阶段基础上，并将重点设定于风险建模、架构、基础设施和业务实现能力上。

该阶段的主要挑战是将应用程序平台和组织的操作方面更紧密的结合在一起。在前一阶段，VirtualWare的团队引入了漏洞管理和有关应用程序安全性的操作。在这一阶段，VirtualWare执行这些领域的下一步骤，并介绍了事故清除应急处理，以及详细的变更控制程序。

VirtualWare为此实施措施选择了启动两个新的领域。虽然VirtualWare不受法律法规的影响，但是他们很多的客户已经开始询问这些平台是否可以通过了相关法律法规。一个小团队已经在VirtualWare内建立，以确定相关的规定驱动因素并建立一个驱动列表。

在前一阶段，VirtualWare引入了一系列新的自动化工具，以协助审核并确定漏洞。虽然不是集中在这一阶段，但开发团队都采用了新的工具，并报告说，他们已经开始从这些工具的使用中获益。

预期目标

在该项目的这一阶段，VirtualWare执行了以下SAMM的实践与措施。

PC1	A. 确定并监控外部的遵守驱动因素； B. 建立并维护遵守指导。
TA2	A. 建立并维护每个项目的滥用用例模型； B. 为威胁的度量采用一个权重系统。
SR2	A. 为资源和能力建立一个访问控制矩阵； B. 根据已知风险指定安全需求。
SA1	A. 维护推荐的软件框架列表； B. 将安全原则明确运用到设计中。
DR2	A. 检查提供安全机制的完整性； B. 为项目团队部署设计审核服务。
VM2	A. 建立一致的事件响应流程； B. 采用安全事件报告流程。
OE2	A. 创建每次发布的变更管理流程； B. 维护正式的操作安全指南。

为了实现这些成熟度等级，VirtualWare在这一阶段实施了一系列活动计划。下列措施被执行：

✦为组织内的项目定义并公布关于安全需求和安全架构的技术指南；
✦确定并记录合规审查和监管需求；
✦为应用程序基础设施的安全性确定并创建指南；
✦创建一个已通过的发展框架的定义列表；
✦加强在VirtualWare内使用现有的威胁建模过程；
✦采用一个事件响应计划，并准备一个安全公布处理程序；
✦为所有项目引入变更管理程序和正式的指南。

为配合在上一阶段针对开发人员引入的自动化工具，安全编码技术的正式技术指导也被引入了组织。这些技术指导都是关于编程语言和技术的详细技术文档，以及对于每个相关语言/应用程序安全编码技术所提供的指导。

随着将教育和意识宣传计划、技术指导、为协助开发人员而引进自动化工具作为一个混合的方法，VirtualWare开始注意到明显不同的代码被纳入到其应用程序的交付产品版本中。开发人员将根据使用工具和参见培训所提供的积极反馈在计划中变成了现实。

VirtualWare的项目团队有史以来第一次为他们的安全性以及应用程序平台的设计负责。在这一阶段，一个正式地审核处理程序以及对于最佳实践的验证由每个团队执行。有些团队发现在安全性和业务设计之间存在差距，这些差距需要加以审核。一个正式的计划已部署，以确保解决这些差距问题。

一个正式的事故响应计划和变更管理处理流程在项目的这一阶段被引入。这是一个艰难的实现过程，VirtualWare团队最初很难适应这些处理流程，因为影响到他们的企业文化，另外，业务操作十分困难。然而，随着时间的推移，每个团队成员在新的处理流程中明确了新的价值，并且，在执行期间所产生的改变也被团队得到了认可。

执行成本

在项目的这个阶段投入了大量的内部资源和成本。这一阶段的相关成本有两种不同类型。

内部资源需求

内部资源使用在该阶段应用程序安全初始活动的内容创建、研讨会组织和审核之中。资源使用以每个角色总天数的方式展现。

开发人员	5天	业务拥有者	6天
架构师	7天	QA测试员	10天
经理	9天	运营支持人员	3天

外包资源

由于VirtualWare内部知识的匮乏，外部资源被用来协助内容创建，以及为开发人员创建/提供培训计划。

安全顾问	20天

第四阶段（第9至12个月）— 监管和操作安全

在VirtualWare内实施保证计划的第四阶段，通过强化组织内现有的安全功能，继续先期的阶段。现在，VirtualWare已实施了很多关键的应用安全程序安全处理流程和机制，以确保应用软件安全地开发和维护。

本阶段的核心是支持调整和监管规定。这三项功能对于一个有效的长期应用程序安全策略的基础起着至关重要的作用。一个完整的培训计划已经实施，与此同时，长期的战略路线图也在VirtualWare内部署就位。

这个阶段的另外一个重点在于计划实施的业务操作方面。 VirtualWare的管理层先前明确，事件响应计划的需求和专门的变更管理流程，对于长期的战略是至关重要的。

VirtualWare将这个阶段视为他们长远将来的踏脚石。这个阶段看到了组织实施了很多最终的度量措施，以巩固先前阶段获得的成果。从长远来看，这将确保计划实施的处理流程、概念和控制在组织内被合理的执行，以确保他们应用程序平台是最安全的结果。

VirtualWare选择这个阶段向他们的客户介绍了他们的新应用程序安全措施，并向客户提供了关于应用程序安全、安全地部署应用程序和在应用程序中报告漏洞等一系列措施的详细信息。这些活动的主要目的是向他们的客户群逐渐灌输信心，以相信VirtualWare的应用程序是安全的，并且VirtualWare可以协助客户以确保他们所用技术的应用程序环境是安全的。

预期目标

在该项目的这一阶段，VirtualWare执行了以下SAMM的实践与措施。

SM3	A. 引导周期性地全行业成本比较； B. 为以前的安全成本收集度量标准。
PC2	A. 为安全和遵守建立政策和标准； B. 建立项目审计实践。
EG3	A. 建立正式的应用程序安全支持门户网站； B. 建立基于角色的考试或认证制度。
SR3	A. 将安全需求写入供应商协议中； B. 为安全需求扩展审计计划。
CR3	A. 为特定应用程序问题自定义代码分析； B. 为代码审核建立发布关卡。
VM3	A. 为事件执行根源分析； B. 收集每一事件的度量指标。
OE3	A. 为操作信息扩展操作审计计划； B. 对应用程序组件执行代码签名。

为了实现这些成熟度等级，VirtualWare在这一阶段实施了一系列活动计划。下列措施被执行：

✦为所有的项目建立良好定义的安全需求和测试计划；

✦建立并实施一个事故响应计划；

✦为应用程序审核已有的报警处理过程，并为事件捕获记录处理流程；

✦为部署应用程序安全创建一本客户安全白皮书；

✦审核项目中已有的安全支出，并确定每个项目是否为安全分配了恰当的预算；

✦为应用程序角色执行最终的教育和意识宣传计划；

✦为组织完成一个长期应用程序安全战略路线图。

在以前的阶段中，VirtualWare已为客户发布了一个正式的事件响应计划，以提交在他们的代码中发现的漏洞。在该阶段，VirtualWare提取了提交的漏洞结果，并执行相关的评估，以分析造成问题的原因、如何并试图从提交的漏洞中鉴别并确定常见的问题。

由于不断地努力，以确保应用程序在内部和在客户网络中安全地部署，VirtualWare为客户编写并提供了一系列基于行业标准而建议的环境强化白皮书。这些指导的目的是为客户部署应用程序的最好方法提供协助。

在这一阶段，VirtualWare执行了一个基于计算机的短期培训课程，使现有的和新招聘的开发人员可以保持他们在应用程序安全方面的技能。另外，还要求所有与应用程序相关的人员在每年参加一个为期一天的培训课程。这就完全保证所教授给开发人员的技能不被忘记，而且新的开发人员在在职期间能拥有所必需的技能。

在VirtualWare内实施的最终任务之一是完成一个“AS IS”的差距评估和审核，并确定过去12个月的活动是否有效。在这个短期活动中，问卷调查被发送给所有团队成员以参与对于SAMM的基准审核。在审核过程中发现的弱点和优势被记录在组织的最终策略路线图中，并为VirtualWare设置未来十二个月的策略计划。

执行成本

在项目的这个阶段投入了大量的内部资源和成本。这一阶段的相关成本有两种不同类型。

内部资源需求

内部资源使用在该阶段应用程序安全初始活动的内容创建、研讨会组织和审核之中。资源使用以每个角色总天数的方式展现。

开发人员	4天	业务拥有者	6天
架构师	7天	QA测试员	1天
经理	9天	安全审计员	11天

外包资源

由于VirtualWare内部知识的匮乏，外部资源被用来协助内容创建，以及为开发人员创建/提供培训计划。

安全顾问	22天

下一阶段（第12个月以后）

在过去十二个中，VirtualWare 已实施了一系列的培训和教育计划，以制定内部的指导和政策。在保证计划实施的最后阶段，VirtualWare 开始对外发布，并与他们的客户一起工作，以提高他们客户应用程序平台的安全性。

VirtualWare 的管理层设置了一个原始指令，以确保在公司内部开发的软件是安全的，并确保市场意识到了已采取的安全措施，以协助客户保护他们的应用程序平台。

为了实现这些管理目标，VirtualWare 在开始的十二个月中建立了一个有效策略，并最终开始帮助客户以安全的形式保护他们的应用程序环境。通过不断的发展，VirtualWare 已在组织内设置了多项举措，以确保该公司不会再出现以前的错误习惯。这些计划包括：

✦业务拥有者和团队领导者意识到了与他们的应用程序相关的风险，并需要在应用程序发布前解决这些风险；

✦团队领导人现在要求所有的应用程序正式通过安全处理流程，并由开发人员在每周进行代码审核；

✦为所有的项目工作人员在每年提供培训和教育计划（包括基于计算机的培训）；另外，所有的开发人员都要求在每年至少参加一次课程；

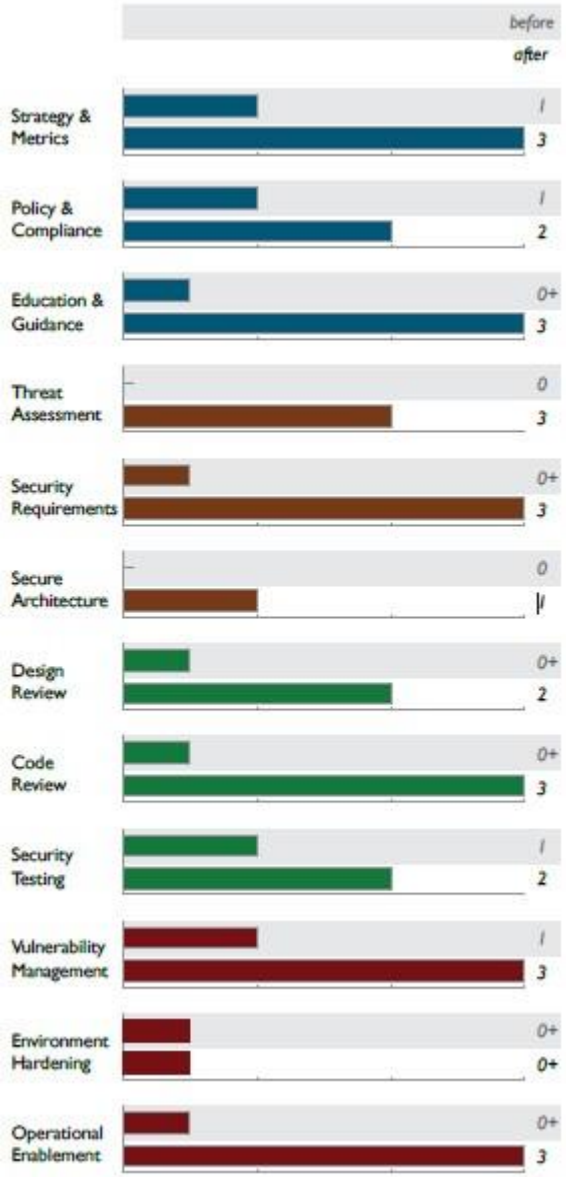
✦为应用程序安全成立一个专门的安全领导人，负责客户沟通、客户技术文档和指导。

随着发展，VirtualWare现在已有了一个针对软件安全性的企业文化，已作为其软件开发生命周期的一部分，从而确保为客户开发和提供的应用程序是安全可靠的。一个有效的处理流程已经部署，以报告漏洞并在组织需要时进行处理。

在实施的最后一个阶段，一个项目的差距评估被执行，以确定在实施过程中发现的任何弱点。特别是由于员工的高流动性，VirtualWare需要不断地对组织新招聘的开发人员进行培训。并为开发人员专门开展一个介绍计划，设定为解决这个问题的一个关键目标，以让他们在组织内开始工作时就接受正式的安全培训。这也将有助于在组织和开发团队中建立关软件于安全性的重要思想。

成熟度记分卡

成熟度记分卡在VirtualWare软件保证计划实施的自我评估阶段完成。最后的计分卡（如右图所示）代表VirtualWare在这个为期四个阶段的项目执行开始前和结束后的状态。



For the latest version and additional info, please see the project web site at

http://www.opensamm.org

Author & Project Lead

Pravir Chandra

Contributors/Reviewers

Fabio Arciniegas
Matt Bartoldus
Sebastien Deleersnyder
Jonathan Carter
Darren Challey
Brian Chess
Dinis Cruz
Justin Derry
Bart De Win
James McGovern
Matteo Meucci
Jeff Payne
Gunnar Peterson
Jeff Piper
Andy Steingruebl
John Steven
Chad Thunberg
Colin Watson
Jeff Williams

Sponsors

Thanks to the following organizations that have made significant contributions to the SAMM Project.

Supporters

Thanks to the following organizations for helping review and support the SAMM Project.

Note: OWASP and the SAMM Project do not endorse any commercial products or services

License