Microsoft®
安全开发生命周期

Microsoft SDL 的简化实施

有关最新信息，请访问 http://www.microsoft.com/sdl。

本文档"按原样"提供。本文档中的信息和观点（包括 URL 和其他 Internet 网站参考）如有更改，恕不另行通知。您自行承担使用本文档的风险。

本文档未授予您任何与 Microsoft 产品知识产权有关的法律权利。您可以出于内部参考目的，复制和使用本文档。

© 2010 Microsoft Corporation。 保留所有权利。

目录

介绍

本文将介绍 Microsoft 安全开发生命周期 (SDL) 的核心概念，并讨论要遵循 SDL 过程所应执行的各种安全活动。

本文内容：

- Microsoft SDL 的简要概述。
- Microsoft SDL 优化模型概述，重点介绍优化模型中 Microsoft SDL 的适用情况。
- 讨论各种 Microsoft 安全开发实践，包括：
  - 应用程序开发过程中相关人员的角色和职责。
  - 必需的安全活动。
  - 可选的安全活动。
  - 应用程序安全验证过程。

本文所述的过程为 SDL 遵从性设置了*最低点*阈值。这就是说，组织各不相同，开发团队应以适合于可用的人才和资源的方式实施 SDL，但是不能危害组织的安全目标。

需要注意的是，本文只重点介绍用于构建软件应用程序和联机服务以进行外部或内部发布的软件开发安全方法。组织中还有其他专门应对“运营”安全威胁的方法（如 IT 安全实践）；这些过程的管理小组所受的技术和法规环境约束与软件开发小组所受的约束类似，但是管理小组通常不开发用在有重大安全风险的环境中的应用程序软件。

本文将尽可能提供所参考的公开信息来源。本文包括指向有关过程、工具和其他辅助信息的特定讨论的 Web 链接。

关于 Microsoft 安全开发生命周期

安全开发生命周期 (SDL) 是侧重于软件开发的安全保证过程。在 Microsoft，自 2004 起，SDL 作为全公司的计划和强制政策，在将安全和隐私植入软件和企业文化方面发挥了重要作用。通过将整体和实践方法相结合，SDL 致力于减少软件中漏洞的数量和严重性。SDL 在开发过程的所有阶段中均引入了安全和隐私。

Microsoft SDL 基于三个核心概念 – *教育*、*持续过程改善和责任*。对软件开发小组中的技术工作角色进行持续不断的教育，这一点至关重要。对知识传授进行适当的投资可帮助组织正确应对技术和威胁态势的变化。因为安全风险不是静止不变的，所以 SDL 非常重视了解安全漏洞的原因和后果，*要求* 定期评估 SDL 过程并随着新技术的发展和新威胁引入应对措施。收集数据以评估培训效果，使用过程内指标确认过程遵从性，使用发布后指标帮助指导进行进一步的更改。最后，SDL 要求对在危机关头维护应用程序所需的所有数据进行存档。如果配合使用详细的安全响应和沟通计划，组织可以向相关各方提供简明扼要、令人信服的指导。

Microsoft 的安全开发

本文重点介绍 Microsoft SDL 在由 Microsoft 外部的小组普遍使用的开发实践中的应用情况。本文内容与已发布的、在 Microsoft 产品和服务开发中应用的过程密切相关。不过，应注意的是，Microsoft 扩展了本文所讨论的核心概念，将 SDL 应用为反映组织的业务和技术环境的过程。Microsoft 实践的 SDL 从本文所讨论的核心概念发展而来，已应用于成百上千的 Microsoft 产品，这些产品在多种操作系统和硬件平台上运行。全世界 100 多个国家/地区的 Microsoft 团队都使用这些核心概念，以应对如此广泛的技术组合所带来的独特安全和隐私挑战。

本文所述的 Microsoft SDL 过程重点讨论在软件开发生命周期中不同时间点应用经过证明的安全实践。此过程与技术无关，不限于大型企业，可以很方便地应用在任何规模的组织

Microsoft 强烈主张安全和隐私过程透明性。用户在反馈中表示，希望进一步详细了解有关 Microsoft 如何应用 SDL 以帮助保护其产品和服务安全的信息，为此，www.microsoft.com/sdl 上发布了 Microsoft SDL 过程的详细介绍。

Microsoft SDL 优化模型

将安全开发概念整合到现有开发过程时，如果方式不当，可能会造成不利的局面且成本高昂。成功还是失败通常取决于组织规模、资源（时间、人才和预算）以及高层支持等因素。理解良好安全开发实践的要素，根据开发团队的成熟度水平确定实施优先级，可以控制这些无形因素的影响。Microsoft 创建了 SDL 优化模型，以帮助解决这些问题。

SDL 优化模型围绕五个功能领域构建，这些领域大致与软件开发生命周期的各个阶段相对应：

- 培训、政策和组织功能
- 要求和设计
- 实施
- 验证
- 发布和响应

此外，针对这些领域中的实践和功能，SDL 优化模型还定义了四个成熟度水平 – 基本、标准化、高级和动态。

SDL 优化模型从基本成熟度水平（几乎或完全没有任何过程、培训和工具）开始，发展到动态水平（特点是整个组织完全遵循 SDL）。完全遵循 SDL 包括高效的过程、训练有素的人员、专用工具以及组织内部和外部各方的强烈责任感。

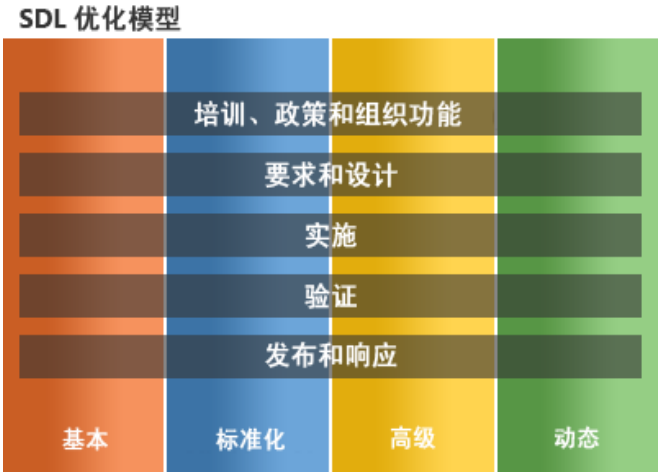


图 1. SDL 具有功能和成熟度水平的 SDL 优化模型

本文重点讨论达到“高级”成熟度水平所需的任务和过程。也就是说，只要组织在前述五个功能领域都具备实力，就完全可以进行 Microsoft SDL 实践。

与其他软件成熟度模型相比，Microsoft SDL 优化模型严格侧重于开发过程改进。它提供可操作的说明性指南，说明如何从较低水平的过程成熟度发展为较高水平，而不采用其他优化模型的“列表的列表”方法。

SDL 适用性

对于将实施 Microsoft SDL 控制的项目类型，为组织设定明确的预期值是非常重要的。通过实践经验可知，具备以下一个或多个特征的应用程序应实施 SDL 控制：

- 在业务或企业环境中部署
- 处理个人可识别信息 (PII) 或其他敏感信息
- 定期通过 Internet 或其他网络进行通信

计算技术无处不在，威胁环境不断变化，因此，可能更方便的方式是确定*不需要*实施安全控制（如 SDL 安全控制）的应用程序开发项目。

安全人员的角色、职责和资格

SDL 包括安全和隐私角色的一般条件和工作说明。这些角色是在 SDL 过程的要求阶段确定的。这些角色实际上是咨询性的，要确定软件开发项目中存在的安全和隐私问题并进行分类和缓解，需要一个由这些角色构成的组织结构。这些角色包括：

- **评析者/顾问角色。**这些角色的任务是对项目安全和隐私进行监督，有权接受或拒绝项目团队的安全和隐私计划。
    - *安全顾问/隐私顾问。*这些角色由项目团队外部的主题专家 (SME) 担任。该角色可以由组织中专门进行此类评析的独立集中小组中的合格成员担任，也可以由组织外部的专家担任。为此任务选择的人员必须担任两个子角色：
        - 审计官。此角色必须监控软件开发过程的每个阶段，并证明每个安全要求的成功实现。审计官必须能够自主证明过程是否符合安全和隐私要求方面的要求，而不受项目团队的干扰。
        - 专家。为顾问角色选择的人员必须在安全方面拥有可靠的相关专业知识。

    最好是集中的内部顾问小组；内部顾问小组具备组织环境和过程知识，这是外部专家无法做到的。
    - *顾问角色组合。*如果可以确认某人具有合适的技能和经验，则安全顾问的角色可以与隐私顾问的角色合二为一。
- **团队负责人。**团队负责人角色应由项目团队的主题专家担任。这些角色负责协商、接受和跟踪最低安全和隐私要求，并在软件开发项目过程中与顾问和决策者保持通畅的沟通渠道。
    - *安全负责人/隐私负责人。*此角色（一人或多人）不仅负责确保软件发布解决了所有安全问题，还负责协调和跟踪项目的安全问题。此角色还负责向安全顾问和项目团队的其他相关方（例如，开发和测试负责人）报告情况。
    - *角色组合。*与安全和隐私顾问角色一样，如果可以确认某人具有合适的技能和经验，则可以由一人承担负责人角色的职责。

简化的 SDL 安全活动

简言之，Microsoft SDL 是一组必需的安全活动，这些活动的执行顺序与其显示顺序相同，按传统软件开发生命周期 (SDLC) 的阶段分组。讨论的许多活动在独立实施时，都具有*某种*程度的安全优势。不过，Microsoft 的实践经验表明，将安全活动作为软件开发过程的一部分来执行，其安全效益大于零散或临时实施的活动。

应重点关注每个阶段的输出准确性。包含不完整或不准确信息的花哨报告只会浪费时间和资源。

为了实现所需安全和隐私目标，项目团队或安全顾问可以自行决定添加可选的安全活动。篇幅所限，不再详细讨论每项安全活动。

图 2：Microsoft 安全开发生命周期 – 简化

需要注意的一个基本概念是，组织应注重每个阶段产生的输出的质量和完整性。以 SDL 优化模型的高级和动态水平运营的组织应具备一定程度的安全过程复杂性。尽管如此，这并不影响威胁模型的产生方式，例如，通过与开发团队进行白板会议产生威胁模型，在 Microsoft Word 文档中以叙述形式描述威胁模型，或者使用专用工具（如 SDL 威胁建模工具）生成威胁模型。投资于高效的工具和自动化，的确会使 SDL 过程受益，但其实际价值在于可以获得全面而准确的结果。

为便于评析，附录 A 提供了演示 SDL 过程的详细关系图。该图以直观的方式说明一个假设项目中使用的安全活动（从培训员工到应用程序发布）。该示例包括必需和可选任务。

必需的安全活动

如果确定对某个软件开发项目实施 SDL 控制（请参见 SDL 适用性一节），则开发团队必须成功完成十六项必需的安全活动，才符合 Microsoft SDL 过程的要求。这些必需活动已由安全和隐私专家确认有效，并且会作为严格的年度评估过程的一部分，不断进行有效性评析。如前所述，开发团队应保持灵活性，以便根据需要指定其他安全活动，但是“必须完成”实践列表应始终包括本文所述的十六项活动。

SDL 实施前要求：安全培训

SDL 实践 1：培训要求

软件开发团队的所有成员都必须接受适当的培训，了解安全基础知识以及安全和隐私方面的最新趋势。直接参与软件程序开发的技术角色人员（开发人员、测试人员和程序经理）每年必须参加至少一门特有的安全培训课程。

基本软件安全培训应涵盖的基础概念包括：

- 安全设计，包括以下主题：
  - 减小攻击面
  - 深度防御
  - 最小权限原则
  - 安全默认设置
- 威胁建模，包括以下主题：
  - 威胁建模概述
  - 威胁模型的设计意义
  - 基于威胁模型的编码约束
- 安全编码，包括以下主题：
  - 缓冲区溢出（对于使用 C 和 C++ 的应用程序）
  - 整数算法错误（对于使用 C 和 C++ 的应用程序）
  - 跨站点脚本（对于托管代码和 Web 应用程序）
  - SQL 注入（对于托管代码和 Web 应用程序）
- 弱加密安全测试，包括以下主题：
  - 安全测试与功能测试之间的区别
  - 风险评估
  - 安全测试方法
- 隐私，包括以下主题：
  - 隐私敏感数据的类型
  - 隐私设计最佳实践
  - 风险评估
  - 隐私开发最佳实践
  - 隐私测试最佳实践

之前的培训为技术人员提供了足够的知识基础。在时间和资源允许的情况下，可能需要进行高级概念方面的培训。示例包括但不限于以下方面：

- 高级安全设计和体系结构
- 可信用户界面设计
- 安全漏洞细节
- 实施自定义威胁缓解

第一阶段：要求

SDL 实践 2：安全要求

“预先”考虑安全和隐私是开发安全系统过程的基础环节。为软件项目定义信任度要求的最佳时间是在初始计划阶段。尽早定义要求有助于开发团队确定关键里程碑和交付成果，并使集成安全和隐私的过程尽量不影响到计划和安排。对安全和隐私要求的分析在项目初期执行，所做工作涉及为设计在计划运行环境中运行的应用程序确定最低安全要求，并确立和部署安全漏洞/工作项跟踪系统。

SDL 实践 3：质量门/Bug 栏

质量门和 *Bug 栏* 用于确立安全和隐私质量的最低可接受级别。在项目开始时定义这些标准可加强对安全问题相关风险的理解，并有助于团队在开发过程中发现和修复安全 Bug。项目团队必须协商确定每个开发阶段的质量门（例如，必须在签入代码之前会审并修复所有编译器警告），随后将质量门交由安全顾问审批。安全顾问可以根据需要添加特定于项目的说明以及更加严格的安全要求。另外，项目团队须阐明其对商定安全门的遵从性，以便完成最终安全评析 (FSR)。

Bug 栏是应用于整个软件开发项目的质量门。它用于定义安全漏洞的严重性阈值：例如，应用程序在发布时不得包含具有“关键”或“重要”评级的已知漏洞。Bug 栏一经设定，便绝不能放松。动态 Bug 栏是一种不断变化的目标，可能不便开发组织的理解。

SDL 实践 4：安全和隐私风险评估

安全风险评估 (SRA) 和隐私风险评估 (PRA) 是必需的过程，用于确定软件中需要深入评析的功能环节。这些评估必须包括以下信息：

1. （安全）项目的哪些部分在发布前需要威胁模型？
2. （安全）项目的哪些部分在发布前需要进行安全设计评析？
3. （安全）项目的哪些部分（如果有）需要由不属于项目团队且双方认可的小组进行渗透测试？
4. （安全）是否存在安全顾问认为有必要增加的测试或分析要求以缓解安全风险？
5. （安全）模糊测试要求的具体范围是什么？
6. （隐私）隐私影响评级如何？应基于以下准则回答此问题：

    - P1 高隐私风险。功能、产品或服务将存储或传输 PII，更改设置或文件类型关联，或是安装软件。
    - P2 中等隐私风险。功能、产品或服务中影响隐私的唯一行为是用户启动的一次性匿名数据传输（例如，软件在用户单击链接后转到外部网站）。
    - P3 低隐私风险。功能、产品或服务中不存在影响隐私的行为。不传输匿名或个人数据，不在计算机上存储 PII，不代表用户更改设置，并且不安装软件。

第二阶段：设计

SDL 实践 5：设计要求

影响项目设计信任度的最佳时间是在项目生命周期的早期。在设计阶段应仔细考虑安全和隐私问题，这一点至关重要。如果在项目生命周期的开始阶段执行缓解措施，则缓解安全和隐私问题的成本会低得多。项目团队应避免到项目开发将近结束时再“插入”安全和隐私功能及缓解措施。

此外，项目团队还必须理解“安全的功能”与“安全功能”之间的区别。实现的安全功能实际上很可能是不安全的。“安全的功能”定义为在安全方面进行了完善设计的功能，比如在处理之前对所有数据进行严格验证或是通过加密方式可靠地实现加密服务库。“安全功能”描述具有安全影响的程序功能，如 Kerberos 身份验证或防火墙。

应考虑将正式的异常或 Bug 递延方法纳入所有软件开发过程。许多应用程序基于旧的设计和代码而构建，因此由于技术方面的约束，可能需要推迟实施某些安全或隐

设计要求活动包含一些必需行动。示例包括创建安全和隐私设计规范、规范评析以及最低加密设计要求规范。设计规范应描述用户会直接接触的安全或隐私功能，如需要用户身份验证才能访问特定数据或在使用高风险隐私功能前需要用户同意的那些功能。此外，所有设计规范都应描述如何安全地实现给定特性或功能所提供的全部功能。针对应用程序的功能规范验证设计规范是个好做法。功能规范应：

- 准确完整地描述特性或功能的预期用途。
- 描述如何以安全的方式部署特性或功能。

SDL 实践 6：减小攻击面

减小攻击面与威胁建模紧密相关，不过它解决安全问题的角度稍有不同。减小攻击面通过减少攻击者利用潜在弱点或漏洞的机会来降低风险。减小攻击面包括关闭或限制对系统服务的访问、应用最小权限原则以及尽可能进行分层防御。

SDL 实践 7：威胁建模

威胁建模用于存在重大安全风险的环境之中。这一实践使开发团队可以在其计划的运行环境的背景下，以结构化方式考虑、记录并讨论设计的安全影响。通过威胁建模还可以考虑组件或应用程序级别的安全问题。威胁建模是一项团队活动（涉及项目经理、开发人员和测试人员），并且是软件开发设计阶段中执行的主要安全分析任务。

首选的威胁建模方法是使用 SDL 威胁建模工具。SDL 威胁建模工具基于 STRIDE 威胁等级分类法。

第三阶段：实施

SDL 实践 8：使用批准的工具

所有开发团队都应定义并发布获准工具及其关联安全检查的列表，如编译器/链接器选项和警告。此列表应由项目团队的安全顾问进行批准。一般而言，开发团队应尽量使用最新版本的获准工具，以利用新的安全分析功能和保护措施。

SDL 实践 9：弃用不安全的函数

许多常用函数和 API 在当前威胁环境下并不安全。项目团队应分析将与软件开发项目结合使用的所有函数和 API，并禁用确定为不安全的函数和 API。确定禁用列表之后，项目团队应使用头文件（如 banned.h 和 strsafe.h）、较新的编译器或代码扫描工具来检查代码（在适当情况下还包括旧代码）中是否存在禁用函数，并使用更安全的备选函数替代这些禁用函数。

SDL 实践 10：静态分析

项目团队应对源代码执行静态分析。源代码静态分析为安全代码评析提供了伸缩性，可以帮助确保对安全代码策略的遵守。静态代码分析本身通常不足以替代人工代码评析。安全团队和安全顾问应了解静态分析工具的优点和缺点，并准备好根据需要为静态分析工具辅以其他工具或人工评析。

一般而言，开发团队应确定执行静态分析的最佳频率，从而在工作效率与足够的安全覆盖率之间取得平衡。

第四阶段：验证

SDL 实践 11：动态程序分析

为确保程序功能按照设计方式工作，有必要对软件程序进行运行时验证。此验证任务应指定一些工具，用以监控应用程序行为是否存在内存损坏、用户权限问题以及其他重要安全问题。SDL 过程使用运行时工具（如 AppVerifier）以及其他方法（如模糊测试）来实现所需级别的安全测试覆盖率。

SDL 实践 12：模糊测试

模糊测试是一种专门形式的动态分析，它通过故意向应用程序引入不良格式或随机数据诱发程序故障。模糊测试策略的制定以应用程序的预期用途以及应用程序的功能和设计规范为基础。安全顾问可能要求进行额外的模糊测试或扩大模糊测试的范围和增加持续时间。

SDL 实践 13：威胁模型和攻击面评析

应用程序经常会严重偏离在软件开发项目要求和设计阶段所制定的功能和设计规范。因此，在给定应用程序完成编码后重新评析其威胁模型和攻击面度量是非常重要的。此评析可确保考虑到对系统设计或实现方面所做的全部更改，并确保因这些更改而形成的所有新攻击平台得以评析和缓解。

第五阶段：发布

SDL 实践 14：事件响应计划

受 SDL 要求约束的每个软件发布都必须包含事件响应计划。即使在发布时不包含任何已知漏洞的程序也可能面临日后新出现的威胁。事件响应计划应包括：

- 单独指定的可持续工程 (SE) 团队；或者，如果团队太小以至于无法拥有 SE 资源，则应制定紧急响应计划 (ERP)，在该计划中确定相应的工程、市场营销、通信和管理人员充当发生安全紧急事件时的首要联系点。
- 与决策机构的电话联系（7 x 24 随时可用）。
- 针对从组织中其他小组继承的代码的安全维护计划。
- 针对获得许可的第三方代码的安全维护计划，包括文件名、版本、源代码、第三方联系信息以及要更改的合同许可（如果适用）。

SDL 实践 15：最终安全评析

最终安全评析 (FSR) 是在发布之前仔细检查对软件应用程序执行的所有安全活动。FSR 由安全顾问在普通开发人员以及安全和隐私团队负责人的协助下执行。FSR 不是“渗透和修补”活动，也不是用于执行以前忽略或忘记的安全活动的时机。FSR 通常要根据以前确定的质量门或 Bug 栏检查威胁模型、异常请求、工具输出和性能。通过 FSR 将得出以下三种不同结果：

- **通过 FSR。**在 FSR 过程中确定的所有安全和隐私问题都已得到修复或缓解。
- **通过 FSR 但有异常。**在 FSR 过程中确定的所有安全和隐私问题都已得到修复或缓解，并且/或者所有异常都已得到圆满解决。无法解决的问题（例如，由以往的“设计水平”问题导致的漏洞）将记录下来，在下次发布时更正。
- **需上报问题的 FSR。**如果团队未满足所有 SDL 要求，并且安全顾问和产品团队无法达成可接受的折衷，则安全顾问不能批准项目，项目不能发布。团队必须在发布之前解决所有可以解决的 SDL 要求问题，或是上报给高级管理层进行抉择。

SDL 实践 16：发布/存档

发布软件的生产版本 (RTM) 还是 Web 版本 (RTW) 取决于 SDL 过程完成时的条件。指派负责发布事宜的安全顾问必须证明（使用 FSR 和其他数据）项目团队已满足安全要求。同样，对于至少有一个组件具有 P1 隐私影响评级的所有产品，项目的隐私顾问必须先证明项目团队满足隐私要求，然后才能交付软件。

此外，必须对所有相关信息和数据进行存档，以便可以对软件进行发布后维护。这些信息和数据包括所有规范、源代码、二进制文件、专用符号、威胁模型、文档、紧急响应计划、任何第三方软件的许可证和服务条款以及执行发布后维护任务所需的任何其他数据。

可选的安全活动

可选的安全活动通常在软件应用程序可能用于重要环境或方案时执行。这些活动通常由安全顾问在附加商定要求集中指定，以确保对某些软件组件进行更高级别的安全分析。本节中的实践可作为可选安全任务的示例，但不应将其视为详尽列表。

人工代码评析

人工代码评析是 SDL 中的可选任务，通常由应用程序安全团队中具备高技能的人员或由安全顾问执行。尽管分析工具可以进行很多查找和标记漏洞的工作，但这些工具并不完美。因此，人工代码评析通常侧重于应用程序的“关键”组件。这种评析最常用在处理或存储敏感信息（如个人身份信息 (PII)）的组件中。另外，此活动也用于检查其他关键功能，如加密实现。

渗透测试

渗透测试是对软件系统进行的白盒安全分析，由高技能安全专业人员通过模拟黑客操作执行。渗透测试的目的是发现由于编码错误、系统配置错误或其他运行部署弱点导致的潜在漏洞。渗透测试通常与自动及人工代码评析一起执行，以提供比平常更高级别的分析。

在渗透测试中确定的所有问题都必须先得到解决，之后才能批准项目发布。

相似应用程序的漏洞分析

在 Internet 上可以找到许多声誉良好的有关软件漏洞的信息来源。在某些情况下，通过对在类似软件应用程序中发现的漏洞进行分析，可以为发现所开发软件中的潜在设计或实现问题获得一些启迪。

其他过程要求

根本原因分析

根本原因分析虽然在传统上不是软件开发过程的一部分，但它对于确保软件安全有着重要的作用。在发现以前未知的漏洞时，应进行调查以确切找出安全过程失败的位置。这些漏洞可能归咎于各种原因，包括人为错误、工具失败和策略错误。根本原因分析的目标在于了解失败的确切本质。此信息有助于确保在将来修订过的 SDL 中不发生同类错误。

过程定期更新

软件威胁不是一成不变的。因此，用于保护软件安全的过程也不能一成不变。组织应从各种实践（如根本原因分析、策略更改以及技术和自动化改进）中汲取经验教训，并将其定期应用于 SDL。一般而言，更新按年度进行应该足矣。发现以前未知的新漏洞类型属于此规则的例外情况。如果出现此现象，则须立即对 SDL 进行非常规修订，以确保在继续进行时已应用缓解措施。

应用程序安全验证过程

开发安全软件的组织自然会需要一种途径来验证是否遵循了 Microsoft SDL 所规定的过程。访问集中的开发和测试数据有助于在许多重要方案（如最终安全评析、SDL 要求异常处理和安全审核）中进行决策。验证应用程序安全性的过程包括以下各种过程和行动人员：

- 应使用专门指定的应用程序跟踪对 SDL 的遵从性。此应用程序用作所有 SDL 过程项的中央存储库，包括（但不限于）设计和实现说明、威胁模型、工具日志上载以及其他过程证明。与任何关键应用程序一样，它应通过访问控制机制确保：
    - 该应用程序只由授权人员使用。
    - 明确区分角色。例如，开发人员应可以使用该应用程序并上载数据，但应禁止他们访问为安全和隐私顾问、安全团队负责人以及测试人员保留的功能。
- 安全和隐私团队负责人负责确保对进行客观判断所需的数据进行正确分类并输入到跟踪应用程序中。
- 输入到跟踪应用程序中的信息由安全和隐私顾问使用，以便为最终安全评析提供分析框架。
- 安全和隐私顾问负责评析输入到跟踪应用程序中的数据（包括 FSR 结果以及顾问分派的其他附加安全任务），并负责保证所有要求都得到满足且/或所有异常都得到满意解决。

此文档重点讨论 SDL 优化模型的“高级”水平，在此水平下，在大多数情况下使用的基本跟踪过程不足以满足任务要求。然而，具有较为简单的过程或较小资源池的组织（适用 SDL 优化模型“基本”或“标准化”的组织）使用较为简单的跟踪过程即可能满足要求。

跟踪和验证过程准确捕获以下内容是十分重要的：

- 组织的安全和隐私要求（例如，在发布时不存在已知关键漏洞）。
- 所开发应用程序的功能和技术要求。
- 应用程序的运行环境。

例如，如果开发团队创建一个在关键环境中运行的过程控制应用程序，则必须投入合适的时间和资源来创建和维护跟踪过程，从而使组织的安全和隐私负责人、高层领导以及相关第三方（如遵从性审核官或评估人）可以进行客观的分析。换句话说，跟踪过程的疏漏不可避免会导致以后出现问题（通常会是紧急问题）。因而务必存在可靠的系统，从而在关键时刻能够解决关键问题。

结束语

本文旨在提供一种*简单*框架，以实用的方式介绍软件开发过程中的安全实践。文中概括了一系列分散的非专有安全开发活动，这些活动与有效的过程自动化和稳固的策略指导相结合时，代表了组织在客观上遵从 Microsoft SDL（按照 SDL 优化模型中的“高级”水平所定义）所需的步骤。

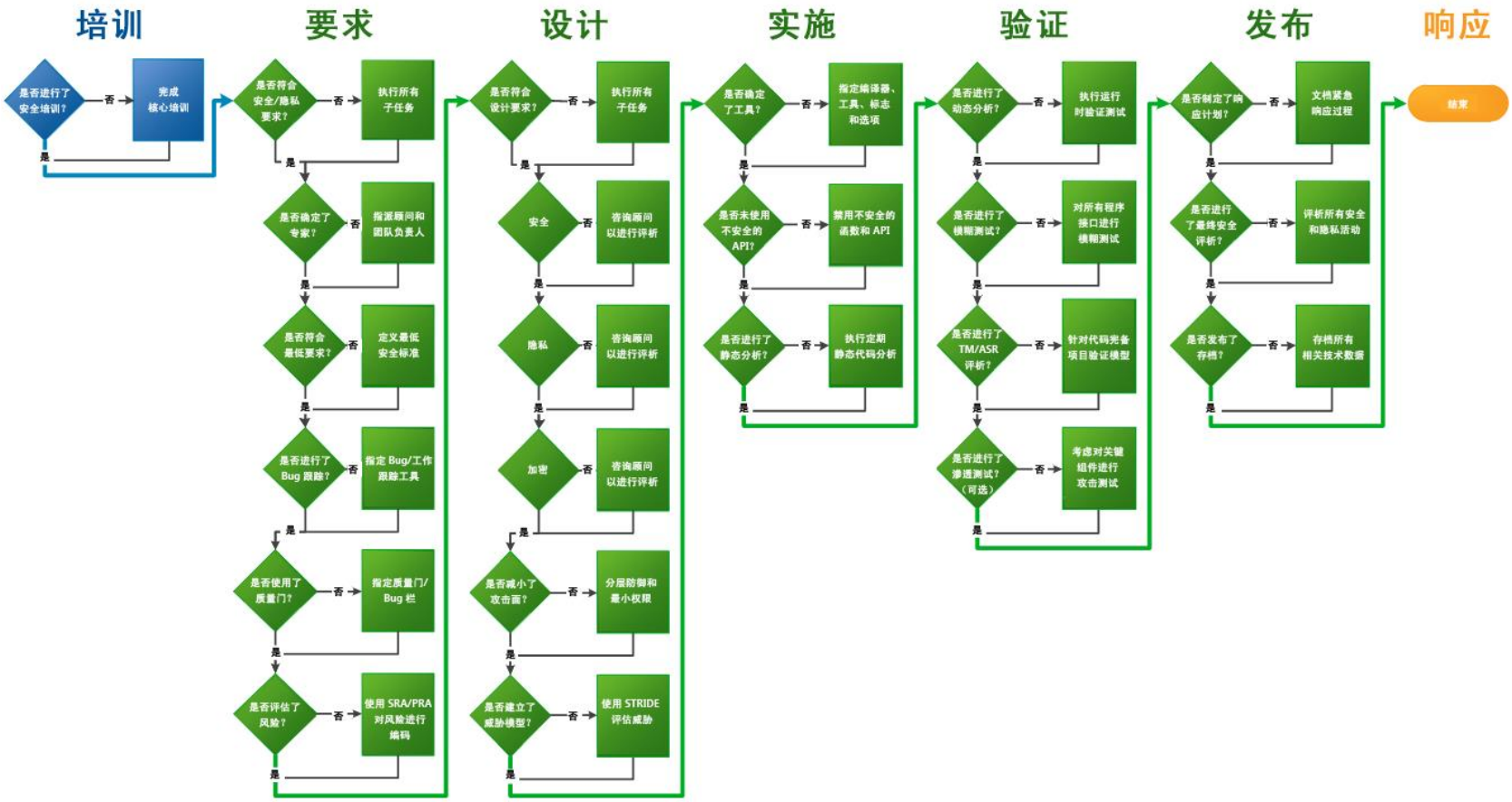
尽管上述过程为 SDL 遵从性设置了*最低点*阈值，但 SDL 并不能以不变应万变。开发团队应以本文为*指导*，实施 SDL 的时候结合考虑组织的时间、资源和业务运营方式。

所讨论的工程概念被普遍视为有效的安全实践，而不是特定于 Microsoft 或 Windows 平台。这些概念适用于不同的操作系统、硬件平台和开发方法。甚至只零散使用其中几种方法，也可能会对所开发应用程序的安全和隐私产生有益影响。

也许有人会说，即使只对安全过程进行简单的综合，也可以在整体上改进安全和隐私，这些改进会超越独立执行的任务的价值。然而，当前计算环境中的威胁已从脚本小子以前对自我满足感的追求演进为大范围有组织的金融犯罪，最近甚至演化为国家战争的新战场。威胁所经历的这种演进强烈要求采用某种比典型零散方法可靠得多的方法实现软件安全。

Microsoft SDL 是用于改进软件安全和隐私的免费提供的过程。该过程已应用于成百上千的软件程序以及数亿行生产代码。SDL 不仅包括遵从传统软件开发过程的必需行动，还十分灵活，允许添加其他策略和方法，从而创建组织所特有的软件开发方法体系。过程、培训和工具的组合具有明显的优势，如可增强可预测性、技术能力及软件安全性，这些优势直接转化为组织及软件用户风险的降低。

附录 A：SDL 过程图示


培训
是否进行了安全培训?
否
完成核心培训
是
要求
是否符合安全/隐私要求?
否
执行所有子任务
是
是否确定了专家?
否
指派顾问和团队负责人
是
是否符合最低要求?
否
定义最低安全标准
是
是否进行了 Bug 跟踪?
否
指定 Bug/工作跟踪工具
是
是否使用了质量门?
否
指定质量门/Bug 栏
是
是否评估了风险?
否
使用 SRA/PRA 对风险进行缓码
是
设计
是否符合设计要求?
否
执行所有子任务
是
安全
否
咨询顾问以进行评析
是
隐私
否
咨询顾问以进行评析
是
加密
否
咨询顾问以进行评析
是
是否减小了攻击面?
否
分层防御和最小权限
是
是否建立了威胁模型?
否
使用 STRIDE 评估威胁
是
实施
是否确定了工具?
否
指定编译器、工具、标志和选项
是
是否未使用不安全的 API?
否
禁用不安全的函数和 API
是
是否进行了静态分析?
否
执行定期静态代码分析
是
验证
是否进行了动态分析?
否
执行运行时验证测试
是
是否进行了模糊测试?
否
对所有程序接口进行模糊测试
是
是否进行了 TM/ASR 评析?
否
针对代码完备项目验证模型
是
是否进行了渗透测试?(可选)
否
考虑对关键组件进行攻击测试
是
发布
是否制定了响应计划?
否
文档紧急响应过程
是
是否进行了最终安全评析?
否
评析所有安全和隐私活动
是
是否发布了存档?
否
存档所有相关技术数据
是
响应
结束