墨菲安全 | OSCS

开源软件供应链安全治理

最佳实践 V1.0

发版时间：2024年04月

写在前面

随着开源软件在企业的应用比例越来越高，各行业使用开源软件的占比已超过90%，开源软件供应链所带来的安全威胁已成为企业面临的主要威胁之一，上个月底刚爆发的“核弹级”开源软件供应链“XZ-Utils”后门事件在全球范围内引起了轰动，而全球最著名的加密勒索组织Lockbit最惯用的攻击手法也是来自开源软件供应链0day/Nday漏洞。在此背景下，近两年国内主要行业（包括互联网、金融、运营商、能源、智能制造、政务数字化等）的头部企业均已开展开源软件供应链安全治理工作，一些头部企业在这项工作开展中已经初见成效。我们希望能够把这些行业先行者和创新者的实践总结出来分享给更多的企业，提升全行业企业开展开源软件供应链治理的效率，最终推动行业共同进步。

这篇最佳实践的内容取材于互联网、金融、运营商、智能制造、能源等领域数十家头部企业在过去近三年时间的实践经验总结，在编写这篇最佳实践的2个多月过程中，我又收到将近20家企业的应用安全专家及安全负责人的高质量建议，这其中包括来自小米的piaca、月胜、涂鸦智能的刘龙威、快手的廖师傅及非零解、百度的长林、知乎的维祥及夜明、传化物流的国超、某支付的刘奇及山人、某科技公司的我宝哥及杨哥、京东的斯诺、蚂蚁的小哥等一众大佬的支持，当然还有来自墨菲安全的一众专业和可爱小伙伴们两个多月的付出。还有很多在这个过程中通过微信和群聊给我反馈的匿名大佬们的支持（匿名的

写在前面

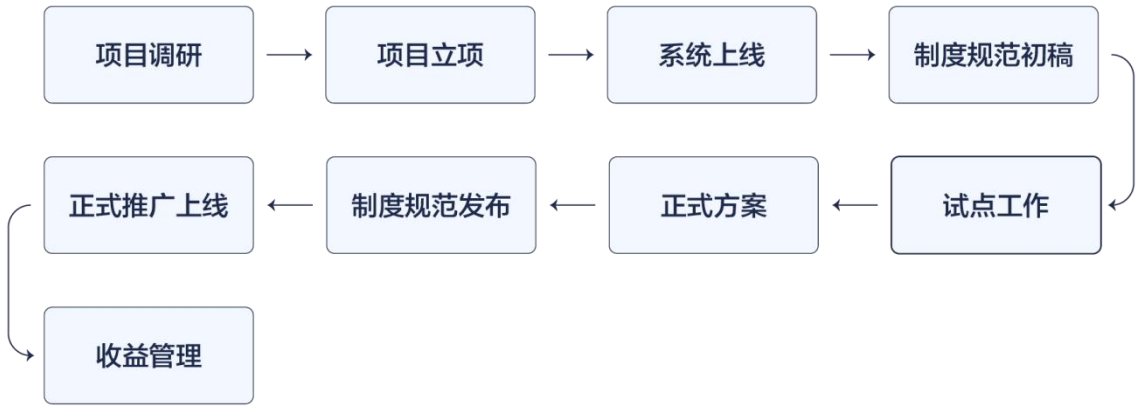
ID，我暂时没有对上号，后续被我发现了之后再致谢吧），如果有遗漏大佬们提醒我一下。过程中我还采访了美团、平安银行、shein、某国有银行、某能源央企、某头部运营商等等的一众大佬，感谢所有为这个最佳实践贡献经验的每一个小伙伴，感恩。

墨菲安全是一家专注于软件供应链安全领域的科技创新企业，创始团队来自百度、华为、乌云等公司，拥有超过十年的企业安全建设经验和全球领先的软件安全攻防专业能力，目前企业已签约服务了蚂蚁、腾讯、阿里、美团、中国银行、移动、电信、国家电网等近百家头部客户，公司成立之初就获得红杉资本的数千万融资。

如果您正在或者正准备开展开源安全治理工作，建议您认真阅读此文，本文会从企业开展开源安全治理工作的早期需求调研到方案制定、试点、正式推广运营等各个环节展开，详细说明每个阶段的具体工作要点及操作方法，希望会给你的工作带来帮助。

本文目标人群	企业应用安全负责人/开源安全治理项目负责人/企业安全负责人
解决什么问题	为目标人群在开展开源安全治理相关的工作时提供详尽的指引和参考，提升项目成功率及效率，少踩坑。
包含什么内容	围绕开源安全治理生命周期的九大核心环节的实施思路、实用方法及实际参考资料

我们把企业的开源安全治理工作拆解为核心的九大环节，本文会对这九大环节分别展开说明，包括每个环节的目标、关键要点及实际操作方法。



一、项目调研

1.1 目的

通过调研，明确企业开展此项工作的必要性和可行性，其中必要性是从企业的核心痛点需求出发，从企业的数据安全、业务保障及监管合规的角度说清楚企业开源安全方面存在的风险及其严重性。关于可行性主要是判断企业当前开展此项工作是否具备条件，并明确投入产出比。基于以上两点的分析，让公司管理层清晰的理解并认同应该立即开展此项工作。

1.2 需求及必要性分析

1.2.1 分析思路

首先，因为此项工作是企业安全治理工作，所以需求及必要性分析必须从公司管理者的视角来展开，必须是公司核心管理者最关注的公司运营层面的风险，并便于公司的核心管理者理解。基于此，建议从企业的数据安全、业务保障及监管合规保障三个角度展开需求的分析，因为通常这三个维度是企业管理者比较关注的。那么我们尝试以这三个维度的风险为目标，来拆解出开源软件安全威胁所带来的影响，并且将风险尽可能量化。

1.2.2 分析方法

分析方法的核心思路是由点及面，立体的呈现当前开源软件安全威胁在数据安全、业务保障、监管合规方面存在的风险。点是指公司面临的最严重的此类安全事件，重要是体现单个问题的严重性，面是指公司到底面临多少此类安全事件，重点体现的是严重性的规模。

关于点，首先需要找出企业历史上因为开源软件安全漏洞或许可合规风险导致的所有安全事件或潜在安全事件，然后对这些事件按照最终影响进行分类，把

它们都归类到数据安全、业务保障、监管合规三个方面。最后，只需要挑出其中每一类中最严重的 1～2 个安全事件案例。如果某一类型在企业过去历史上没有出现过此类风险，那么我们可以从同行对标企业过去产生过的安全事件中找出 1～2 个典型的安全事件。这里的同行对标最好是公司管理层最喜欢对标的同行企业，这样才具备说服力。

关于面，需要对以上三类风险导致的原因进行剖析：

比如是企业软件开发过程中引入开源组件的漏洞导致的数据泄露事件？

比如是企业外部采购的商业软件中引入的开源组件漏洞导致的漏洞攻击，进而导致被勒索，影响业务连续性？

诸如此类，剖析导致这些典型事件的主要原因。

然后，基于以上的原因剖析，那么从规模的角度来统计，公司有多少自研的软件，用了多少开源组件，存在多少同类型可能导致数据泄露的安全漏洞？以此来从“面”的角度分析出事情的严重程度及其规模。

注意，以上从点到面的分析方法，需要分别从数据安全、业务保障、监管合规的角度分别展开一轮分析，最后总结成完整的调研报告。

1.2.3 预期结论

公司历史上发生了 xx 起安全事件，其中跟开源安全威胁相关的事件 xx 起，占比超过 xx%；

其中导致数据安全相关的事件 xx 起，占比 xx%，最严重的两起事件分别是 xxx，导致 xxx 万用户数据的泄露风险，背后的原因主要是我们 xx 系统开发过程中用了 xx 开源组件，该开源组件存在高危漏洞被攻击。

其中导致业务连续性相关的事件 xx 起，占比 xx%，最严重的两起事件分别是 xxx，这两起事件导致 xx 核心业务停摆 xx 分钟，导致企业损失 xx 元。背后

的原因主要是我们采购的 xx 系统用了 xx 开源组件，该开源组件存在高危漏洞被攻击。

其中导致企业面临监管合规风险的事件 xx 起，占比 xx%，最典型的两起事件分别是 xx，这两起事件导致公司收到 xx 监管部门的 xx 影响。背后的原因主要是我们 xx 系统开发过程中用了 xx 开源组件，该开源组件是 GPL 开源许可证，存在知识产权合规风险。

通过分析，引起这些事件的原因均是来自我们的业务系统使用了大量的开源组件，且这些开源组件存在严重的安全漏洞及开源许可证合规问题导致安全事件。通过我们对你企业使用的 GitLab 代码仓库摸底，发现我们仓库中的 xxx 个代码仓库中使用了 xxx 个开源组件，其中能导致此类安全事件的安全漏洞 xxx 个，严重的漏洞 xxx 个，而且每天会新引入 xx 个开源组件，带来 xxx 个漏洞威胁。其中 xx 重点业务受影响的漏洞 xx 个，xx 重点业务系统受影响的漏洞 xx 个。

基于以上，企业多个核心业务系统当前面临非常严重的开源软件安全威胁，应该尽快开展治理工作。

附录 13.2 中含各行业开源安全相关事件参考。

1.3 方案及可行性分析

1.3.1 分析思路

通过需求及其必要性分析，公司管理层已经非常明确此项工作的价值及其必要性，接下来公司管理者最核心思考的问题是如何开展这项工作的治理，如果需要开展治理大概投入是如何的？以及该项工作治理的成功率如何？管理者必须考虑开展此项工作的性价比，以此来最终决定当下是否需要开展此项工作。

1.3.2 分析方法

关于方案及可行性的分析我们通常可以对行业进行调研，通过调研同行业的其他企业面临的风险和举措，起到查漏补缺和对照的作用，为后续方案选型做基础。

行业调研关键要点

- 选取至少 3-5 家同行业、规模和业务形态相近的企业进行调研；
- 同行业是否开展了这项工作、投入的资源情况、牵头负责部门是哪些；
- 同行业识别到了哪些软件供应链/开源组件安全风险，开展项目的动机，怎么解决的（采用方案、解决到了什么程度和周期情况）。

1.3.3 方案调研

通过调研识别有哪些解决方案，调研的范围包括：

- 业界有哪些解决方案，比如有哪些开源解决方案、商业产品、技术框架；
- 同行业的解决方案，比如 A 企业通过自研解决、B 企业采购了某商业产品
- 企业内部有没有类似的解决方案，比如基础架构团队实现了软件资产的统一管理；
- 企业内部研发流程现状调研，包括当前 CI/CD 流程的成熟度，这对于选择什么样的方案也很关键。

常见的调研维度包括：

- 功能调研，该方案具备哪些功能，包括：支持的语言、软件形态、支持分析的风险类型、修复和集成能力等；
- 性能调研，方案的检测准确率、覆盖率、检测速度和系统配置要求等；
- 成本调研，包括实施成本、运营成本、采购成本、时间成本；
- 现状调研，包括企业 CI/CD 流程现状及都使用了什么工具，接入覆盖率及流程成熟度等，这对于选择什么样的产品方案和实施成本的预估会更准

确；

- 同行使用情况，有哪些同行使用该方案。

行业调研报告参考：

附录 13.2 中含各行业完整的调研报告参考，可获取。

1.3.4 方案评估

将这些方案从功能、性能、成本和风险等方面来评估，根据可投入的资源情况，筛选得到初步可行（ROI 较高）的方案。

1.3.4.1 评估表

逐个维度具体评估不同方案，形成如下的方案对比评估表：

一级评估维度	二级评估维度	完全自研	部分自研	使用开源工具 dependency-check	使用商业产品-A	使用商业产品-B
功能评估	语言及包管理工具支持度	高（可根据需求定制）	高	低（只支持部分语言）	高	高
	文件类型支持度	中（二进制支持难）	中	低（不支持二进制）	中	高
	漏洞优先级判断效果	中	中	低（仅参考 cvss 评分）	中	高（漏洞评级优化）
	漏洞可达性分析效果	低	低	低	中（支持覆盖度低）	高（支持覆盖度高）
	漏洞修复能力效果	中	中	低	中	高

	检测结果输出效果	高	中	低	中	高
	系统集成能力	高	高	低	中	高
	API 完善度	高（可根据需求定制）	高	低（不支持）	中	高
	CI/CD 支持度	高（可根据需求定制）	高	低	中	高
	易用性	中（依赖产品能力）	中	低	中	高
性能评估	检测效果（准确率、覆盖率）	中	中	低（改造存在不确定性）	高	高
	知识库更新时效	低（跟进难度极大）	中	低（无保障）	较高	高
	稳定性	高	中	低	中	高
	并发度（同时可检测任务数）	高	中	低	中	高
	检测速度（单任务）	高	中	低	高	高
成本评估	实施成本	极高（需投入 5-10 人研发）	中（投入 1-2 人研发）	低（投入 1 人研发改造）	低（无需研发投入）	低（无需研发投入）
	使用成本	低	中	高	低	低
	维护成本（知识数据维护和功能迭代）	极高	高	高	中	低

	采购成本	无	中	无	低	中
	时间成本	高（半年以上研发周期）	中（三个月研发周期）	高（四个月研发改造）	中（部分功能不具备）	低
风险		高（需要持续长期投入，技术门槛高）	对三方工具核心能力存在依赖导致性能瓶颈	开源工具无服务保障	缺少本地化团队支持	不支持定制
同行使用情况		A	B	C	E	D、F、G、H

1.3.4.2 评估结果初步汇报

基于以上评估表结果，向上级进行总结汇报，以此确定下一步工作方向，主要是确定自研还是结合商业方案：

1）总结每个方案主要的成本投入和收益情况，根据关注的成本和风险，对方案进行综合排序；

2）沟通汇报初步调研的方案结果，决策下一步需要 POC 测试验证的方案。

1.3.4.3 POC 验证方案

在初步筛选了方案后，通常需要通过 PoC（Proof of Concept）测试对方案的可行性、实际效果进行验证。

1.3.4.3.1 明确 PoC 测试方案

方案的确定需要考虑以下方面，因地制宜，避免一味追求检测结果数量的多少而忽略实际应用效果：

- 需求，从要解决的问题出发，判断方案的要解决的重点问题；
- 技术栈，根据内部主要使用的语言、开发工具，决定 SCA 工具需要支持的范围；
- 研发流水线，根据代码仓库、制品库、CICD 等使用情况，决定 SCA 工具需要集成的范围；

- 业务形态，根据业务形态确定需要分析的软件形态，如源代码、固件等，决定 SCA 工具需要分析的范围；
- 现有安全能力，根据已有安全能力情况，考虑 SCA 工具可能需要具备的扩展性、联动能力。

例如：

- **功能验证**
  - 由于部分检测对象无法提供源代码，SCA 工具还需要能够对制品文件进行分析；
  - 由于配置中依赖的第三方组件，在代码中可能没有调用，导致漏洞不能被利用，因此需要通过可达性分析判断漏洞在应用中的可利用性。
- **性能验证**
  - 由于 SCA 工具需要集成进流水线检测，需要考虑检测的速度和并发检测能力。

附录 13.2 中，提供了详细的行业企业 POC 测试方案及测试用例参考，可获取。

1.3.4.3.2 基于 PoC 测试方案进行验证，需要注意：

- 测试过程覆盖每一个测试项；
- 每项的测试用例应尽可能涵盖未来的所有使用场景；
- 选择具有区分度的测试样本作为用例；
- 准确率与覆盖率可参考 SCA 基准测试项目。

1.3.4.4 确定技术方案

根据方案初步评估情况和 POC 验证结果，根据预计可投入的资源情况，汇总得出当前最优的方案及其预期效果（即可解决哪些问题、解决的程度如何），此处注意不一定需要一下子解决所有问题，应该列出分阶段投入多少资源把这项问题解决到何种程度，因为本身也没有绝对的安全。站在企业的管理层角度关注的是投入产出比，同时将安全风险控制在一个可接受的范围。

1.3.3 预期结论

同行调研结果：

- 同行业同规模的 xx、xx、xx 等几家公司均已开展开源安全治理工作，xx、xx 等几家公司今年正在开展开源安全治理工作；
- 同行业均是有 xx 部门牵头，xx、xx 部门配合开展此项工作；
- xx 公司（与我们公司最相似的公司）治理到 xx 程度了，投入了 xx 人力，使用的是 xx 商业产品，投入预算 xx 万/年；目前取得的效果是 xxx（描述一下做到什么程度了，什么时候开始建设的）；
- 附行业调研报告。

我公司建设方案：

- 基于同行业分析，结合我们公司的风险现状和基础设施现状，建议采用成熟商业产品工具作为底层平台，将商业工具与公司研发流水线对接、结合安全运营平台落地；
- 预计需投入预算 xx 万，人力 xx 个，需要协调 xx 部门配合 xx 工作；
- 预计一年内达成 xx 目标，2-3 年内达成 xx 目标；
- 附技术方案调研报告及商业产品对比报告。

二、项目立项

立项的目的是获得公司管理层、及项目实施相关方的认可和资源的支持，推动相关方协同配合好项目工作，保障项目能够推进落地。

2.1 初步制定整体运营方案

在有了上一步的技术方案之后，通常还需要制定运营方案（即如何通过工具解决项目问题，达成预期效果），两者作为立项汇报材料的主要内容。

2.1.1 项目背景及需求

此项内容就是上面第四项的项目调研的结论，从点到面，呈现公司当前由开源软件安全威胁引起的数据安全、业务保障及监管合规相关的风险。

2.1.2 项目目标

项目的目标肯定是与项目需求和痛点直接相关的，通常站在公司管理者的角度上考虑，我们确定目标的时候需要存在两个维度的目标：A 面目标（业务价值）和 B 面目标（能力沉淀）；一般也称为业务指标和技术指标。业务指标，关于安全质量和合规质量的提升，包括事件数量减少，漏洞数量减少，研发漏洞修复效率，运营效率提升等，而技术指标关注相关技术维度的能力落实进度，包括各种平台对接，新技术应用，以及整体能力对业务的覆盖面等等。一个项目兼顾创造业务价值和能力沉淀，对于公司的价值才是长远和可持续的。

以下给出关于目标的一个参考建议示例：

A 面：

- 对于有外网暴露面的业务，开源软件造成的增量安全风险收敛 100%；
- 100%收敛强互惠性许可开源代码使用，避免开源许可合规事件；
- 对于有外网暴露面的业务，存量风险 1 年内高危及以上风险 100%收敛；
- 2 年内，有外网暴露面的业务中低危风险收敛 80%以上，无外网暴露面的业务高危及以上风险 100%收敛。

B 面：

- 2 年内，建立开源软件资产管理平台，100%覆盖公司自研及外部引入的应用系统；
- 建立一套完善的开源软件风险全生命周期控制体系，覆盖率 100%。

关于制定目标的一些要点：

1. 目标可包含 A 面（业务价值）和 B 面目标（能力沉淀，保障目标的持续达成）；
2. 目标设定可从存量和增量两方面风险治理展开，通常包括风险的治理率、处置时间等；
3. 目标应尽可能向上承接、对齐，如对齐部门的目标、承接公司整体的安全保障目标；
4. 根据收益程度定目标，避免过泛的范围导致收益不清晰，建议挑选能够完全解决的重点业务风险实现 100%治理，而不是全量范围的非 100%治理。

2.1.3 技术方案

此项内容就是上面 4.3 节内容的总结，需结合公司软件研发及采购流程来说明清楚要实施的技术方案，并给出方案需要哪些团队配合做哪些工作。

2.1.4 运营方案及实施计划

在明确目标及技术方案后，需要规划项目的整体实施计划，通常包括：系统上线、发布制度规范、试点验证、形成正式方案、正式上线推广运行、例行运营等。

在规划的过程中需要注意不同环节之间的前后依赖，明确每一环节的负责人，同时需要预估每一环节的开始时间和结束时间。

2.1.5 各环节工作计划和目标

在有了大的方向之后，进一步细化各个环节的工作内容和目标、涉及到的相关方，并和协同方达成共识，例如：

系统上线的目标是具备一个可用的开源安全治理系统，用于后续的运营落地，工作计划包括：

- 系统发布：申请资源，提交发布流程。预估时间：1.11-1.13 号 协同方：运维部门；

- 功能验证：发布后验证各项功能是否符合预期。预估时间：1.12 号 协同方：测试部门；
- 发布操作手册：根据发布后的系统情况，形成针对性的操作手册。预估时间：1.13 号。

2.1.6 项目风险及处置预案

在项目运营实施过程中，可能会面临哪些阻碍项目推进的风险，对应的处置方案是什么。

例如：

风险：今年业务迁移上云环境，上云导致研发模式可能有较大变动，当前上云具体计划未确定；

处置方案：流水线迁移计划明确后提前验证和对接云上研发流水线，保障安全卡位能力连续性；

评估的维度：

- 资源（预算、人力）；
- 相关方配合情况（业务线配合情况、采购进度、定制开发导致的供应商交付风险）；
- 如果自研关键工具平台，涉及研发失败等风险；
- 供应商方案不及预期等。

常见的关键点包括：

1. 项目收益应 > (风险出现的概率 * 风险解决的成本)，否则项目不具备可行性；
2. 每项风险应有对应的处置方案，否则立项可能失败。

2.2 立项汇报

汇报的最主要的目的是为了达成各方的共识，包括该项目落地过程中所涉及的所有关键相关方，这些相关方都必须明确该项目的价值及各自在该项目中需要承担的工作。

2.2.1 明确汇报对象

判断需要有哪些相关方重点支持，则需要通过汇报与这些重要的相关方领导达成共识。如果多个相关方具有一个统一的领导，则需要与该领导汇报达成共识。请注意，一定要覆盖所有项目落地环节中涉及的重要相关方的决策人。

2.2.2 关键汇报对象预沟通

在明确了汇报对象（参会人员）后，除了与上级沟通外，还需要提前与关键人物沟通，避免在汇报过程中由于相关方工作安排冲突、讲解原因导致没有获得关键人物的支持。

关键人物通常是在项目立项会议中具备较高话语权、在项目中投入最大或受益最高的人员，根据各个企业的不同，可能会是：

- CTO/CIO；
- 研发部门总监；
- 运维部门总监；
- IT 部门总监；
- 财务总监；
- CEO 助理。

预沟通建议采用面对面沟通的方式进行，向对方介绍项目的整体情况，希望对方协助的事项和收益等。应该提前了解相关方的疑问、诉求等，并尽量进行解答和满足。如果预沟通阶段发现有相关方无法配合和认可的情况，一方面应该寻求其领导的支持，另外一方面在确定无法支持后，应该给出其他备选方案，此方案不对此相关方形成关键依赖。

2.2.3 正式汇报

汇报阶段通常有以下注意事项：

1. 参会人员可能事先不完全了解项目，因此需要按照听众能理解的方式介绍项目背景；
2. 在介绍方案和计划中体现有哪些需要他人协同的工作、资源需求情况；
3. 篇幅不宜过长、语速不宜过快；
4. 汇报会前应该确认关键相关方的决策人到场；

2.2.4 正式发布汇报结论

汇报结束后，立项成功、评审纪要等关键性结论一定要通过书面（邮件）的方式知会参会方和项目相关方，起到盖棺定论的效果，同时也是对项目组的初步认可，有利于后续工作开展。

2.2.5 预期效果

项目立项通过，得到所有关键相关方的认可。

三、系统上线

系统上线是开源安全治理的重要一环，通过将系统投入实际使用，为后续试点及最终开源风险治理工作提供能力支撑，验证项目运营方案在实际实施过程中的可行性，通过系统使用过程中发现的问题及时对运营方案进行调整，保证运营方案能够有效的实施和运行。

3.1 平台部署上线

3.1.1 系统发布

为了让用户能够正常访问并使用系统，需要将系统发布在企业内部指定环境，在实际上实施过程中应注意以下事项：

a. 确认系统所需运行环境及相关资源已准备就绪；

b. 确保系统运行所需各项配置正确，如数据库连接、网络配置等；

c. 确保系统能够正常运行。

在实际部署发布过程中由于企业内部实施流程差异，可能需要运维部门支持系统的发布，因此为了系统能够顺利发布，需要与相关人员做好沟通及配合。

3.1.2 功能验证

功能验证旨在确保系统上线后各项功能能够正常运行，过程中需要对系统的功能完整性以及系统可靠性进行验证，确保用户能够按照预期进行操作，系统能够在实际使用过程中稳定运行。

由于系统需要与研发流程中所使用工具进行集成，进而实现安全能力的嵌入，因此在测试过程中应重点关注系统与内部所用研发工具是否能够正常集成接入，以及接入后各项操作功能是否正常。

因为系统需要面向所有工程师（包括研发及安全工程师）开放，所以产品的操作文档及知识库的可读性、完整性其实也是挺重要的，需要找几个目标用户对系统配套的文档及操作手册、知识库进行测试验证。

除此之外，对部署的平台进行一轮安全性测试也是非常必要的，以确保安全自身的系统平台的安全性，防止出现安全问题，使得安全团队的专业性遭到质疑。

3.2 配套的文档及资源

3.2.1 产品安装部署文档

产品安装部署文档的目的是为用户提供系统的详细安装和部署步骤，确保能够顺利将系统部署到指定的环境中，因此在制作产品安装使用文档时应注意以下几点：

a. 文档应该覆盖系统的所有部署环境，包括物理服务器、虚拟机、云服务等，以满足不同用户的部署需求。

b. 需要明确列出系统部署所需的硬件和软件环境，包括操作系统版本、数据库版本、依赖库等，确保用户的环境能够满足系统的运行要求。

c. 按照步骤详细描述系统的安装和配置过程，包括下载安装包、解压文件、配置参数等，确保用户能够按照指导正确完成部署。

d. 需要提供系统的初始化和启动步骤，包括数据库初始化、服务启动等，确保系统能够顺利启动并进入正常运行状态。

3.2.2 操作手册

操作手册旨在为用户提供详细的系统操作指引，保证用户了解及正确使用系统的各项功能，因此提供使用手册时应注意以下几点：

a. 操作手册需要与系统的实际情况保持一致，确保操作手册中的内容与系统的实际操作一致，避免出现误导用户的情况。

b. 操作手册的内容应该简明扼要，清晰易懂，避免使用过多的专业术语，确保用户能够快速理解和掌握操作方法。

c. 操作手册应该按照操作流程和模块划分，对每个步骤进行详细说明，包括文字描述、截图示例等，确保每一步操作清晰明了。

3.2.3 产品常见问题解决及故障排除指南

产品常见问题解决及故障排除指南的目的是帮助用户在使用产品时遇到问题时能够快速定位并解决，确保系统的稳定运行，因此在制作产品常见问题解决及故障排除指南应该注意以下几点：

a. 针对常见问题和故障，提供清晰的解决方法和排查步骤，包括文字说明、截图示例等，让用户能够快速理解和操作。

b. 根据问题的严重程度和影响范围，将问题分类，区分紧急问题和一般问题，提供相应的解决方案和应对措施。

c. 需要提供常见问题的预防措施和注意事项，帮助用户避免常见问题的发生，提高系统的稳定性和可靠性。

d. 在编写指南时，需要考虑用户的技术水平和使用经验，提供简单易懂的解

决方案，确保用户能够正确理解和操作。

3.2.4 服务支持及质量保障

除了配套的文档及故障解决指南之外，需要提供服务支持群或提供 oncall 机制作为兜底方案，一方面方便用户反馈一些使用过程中存在的紧急问题，另外一方面也可以从这个群收集用户的使用反馈。建立一个和用户高效沟通的渠道。

既然配备了相应的服务支持群及 oncall 机制，那么就应该对内对外明确相应的服务质量标准，包括每个问题的响应时效性，问题跟进闭环的保障机制等，同时需要建立对运营人员的考核机制。

此外，需要考虑这一套安全系统出现故障时的降级处置方案，保障不能对业务的正常上线及发布流程造成影响。

四、制度规范初稿

为后续开展开源安全治理工作建立标准和指导依据，同时为后续的治理工作的推进提供公司级的背书。

4.1 制度规范编写准备

- 确定内部制度编写的发起和审批流程；
- 调研国内/国外关于开源安全治理相关的国家标准/行业及团体标准，形成内容参考；
- 调研同行业同业务类型公司的开源安全相关制度制定情况，如果有成熟的方案可进行参考；
- 调研公司内部是否有《软件开发安全》/《漏洞管理条例》等现行管理制度，并查看其制度内容中是否有相关描述，记录相关内容和制度修订部门，考虑相关的制度规范的合并工作，尽量避免出现过多的同类型制度规范；

- 根据立项中确定的治理目标及治理方案，明确制度规范应该约束的核心内容，并与公司内部各相关团队沟通，了解各部门/各角色在软件供应链安全方面的诉求，沟通其应当承担的义务，防止制度编写过程中存在阻力；

4.2 制度规范编写

4.2.1 明确参编人员范围

根据该治理项目涉及的所有相关方，比如研发团队、运维团队、法务团队、安全团队等，向各团队征集代表人员参与编写，该代表人员最好由各部门的负责人指派和自由报名两种方式选出，这样既保障未来可以得到各关键部门的负责人审批，同时又得到相关人员的广泛参与和认可。

通常至少包含以下人员：

- 安全团队：安全能力的提供方，同时做定期的安全检查和安全漏洞的识别和下发；
- 开发团队：安全漏洞的直接责任人，负责安全漏洞的修复，同时防止安全漏洞的再次发生；
- 运维团队：配合安全团队建立安全能力；
- 内审团队：安全的第三道防线，负责审计安全措施是否执行到位，出现违规事件后进行相应处罚；
- 法务团队：负责许可证合规方面的法律解释和制度建立。

明确人员范围的最终目的是为了通过制度文件说清楚各方的职责及权利、义务，这个后续可以在编写制度规范过程中各团队协商明确并最终落在正式的文件中。

4.2.1 核心内容及编写

4.2.1.1 设定政策和流程

编写具体的政策和流程可参照以下关键点进行编写，具体内容以实际公司

情况为准。

a. 开源组件准入

主要内容描述开源组件的准入流程，根据实际落地情况为准，确保引入的组件满足集团、公司的安全、合规以及安全管理要求。

b. 版本管理

对开源软件实施有效的版本管理与控制，包括对版本一致性、并发性、安全性等进行管控，可确保高质量、稳定的版本应用到生产环境，同时可及时跟踪新版本存在的问题，进行及时修复。

c. 黑名单开源软件管理

主要描述黑名单组件的来源以及黑名单组件的管理办法。

d. SCA 运营流程和政策：

规范日常 SCA 产品的运营流程，明确每个角色在日常运营中的责任和义务，以及处置时效性。

e. 开源软件突发事件应急处置：

规范开源软件突发事件的应急处置流程，明确每个角色在日常运营中的责任和义务，以及处置时效性。

4.2.1.2 实施和监督

a. 描述如何执行这些政策和流程，包括使用的工具、技术和资源。

b. 设立监督机制，确保制度的有效实施，如定期审查、报告制度等。

4.3 征求意见

- 组织内部审阅：邀请项目相关人员和管理层审阅制度规范草案，收集反馈。
- 外部专家咨询：可选，寻求外部安全专家或顾问的意见，以增强制度规范的全面性和实用性。

4.4 形成初稿

- 根据收集到的反馈和建议，修订和完善制度规范文档；
- 确定制度规范的最终版本，准备提交给公司负责流程制度规范的管理层审批。

五、试点工作

试点工作的主要目的：

1. **验证方案有效性**：确认运营方案的具体措施在实际操作中的有效性，并且这个过程也能够检验上线的系统和工具在真实环境下的适用性。
2. **提前发现和解决问题**：通过圈定有限的试点范围并进行实施，及时发现和识别实施过程中可能遇到的技术和管理问题，并解决问题或者进行运营策略的调整，避免问题遗留到项目的全面推广环节。
3. **优化制度规范和流程**：基于试点工作中接收到的反馈和对试点结果的分析，对制度规范初稿、项目运营流程进行优化和调整，提高制度规范的可操作性。
4. **评估和调整项目指标**：根据试点的结果来评估项目目标的合理性，并调整和设定更加准确和实际的项目成功评估指标和阶段性目标，便于正式运营后对项目整体效果的评估。
5. **避免破坏用户信息**：通过小范围的样本验证，保障未来运营时意外的故障和问题不会对线上大量的研发和业务人员产生影响，从而导致用户的信心丢失。

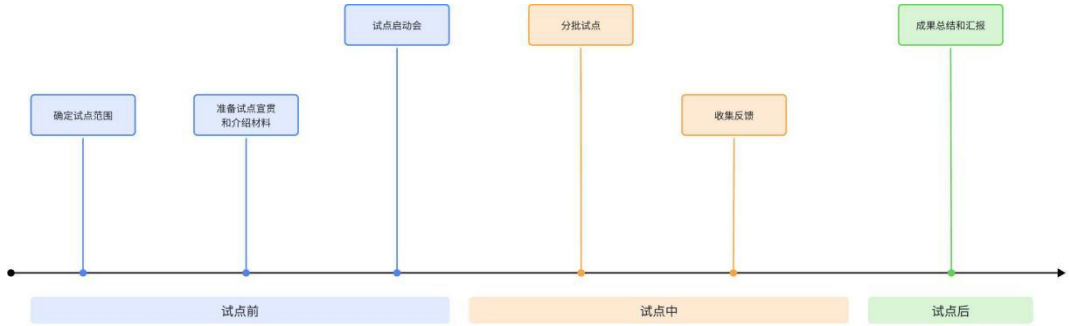
5.1 试点工作计划

我们把整个试点环节分为三部分，分别是试点前、试点中和试点后：

- 试点前：**确定试点范围**对于整个试点工作起到至关重要的作用，根据业务的关键级别和业务部门配合程度等多个维度划出试点范围。为了试点工作的顺利进行，应该确保参与试点的相关方都能明确自己的工作内容和职责，因此**准备试点宣贯和介绍材料**，并召开**试点启动会**进行宣贯是必不可

少的环节。

- 试点中：在实际试点的过程中，使用**分批试点**的方式能够逐步验证和调整项目策略，避免因为过程中产生的问题直接影响整体试点工作和最后的效果。同时设立有效的机制来**收集反馈**，以便及时发现和解决问题。
- 试点后：**成果总结和汇报**环节是对试点项目进行回顾和评估的重要步骤，通过对试点结果进行分析和效果数据整理，准备汇报材料，并召集试点相关方和项目决策层进行汇报，提升项目决策层对未来项目正式推广运营的信心。



5.2 确定试点范围

试点范围一般是在公司中选择几个部门或团队进行，这些参与试点的部门/团队一定要具有代表性。这样做的好处是一方面可以充分验证 SCA 运营方案在不同场景下的实施效果，确保试点结果的普遍适用性；另一方面是试点结束并取得阶段性的成果后，这些代表性的部门/团队作为参与方可以帮助我们提升整个项目在公司内的影响力，也会提高未来项目正式运营后其他部门/团队的配合程度。

5.2.1 关键点

在具体选择参与试点的部门/团队时，我们可以根据一些维度进行筛选。通常重点会考虑的是部门/团队的业务重要性以及配合程度，除此之外还有其他维度可以考虑。

评估维度	评估说明	评估权重
业务重要性	评估该部门/团队是否存在公司的重要业务，是否属于公司的重要事业群组。通常情况下，重要性越高的业务对于安全越重视，对于代码中出现的安全风险会花费更多的精力去处理。因此存在公司重要业务的部门/团队更加适合参与试点。	30%
配合程度	评估该部门/团队对安全工作的配合程度，可以结合历史安全问题处置的态度和积极程度进行评估。配合度较高的部门/团队，在试点过程中对出现各种各样的问题接受度也会较高。因此对安全工作配合度高的部门/团队更加适合参与试点。	30%
业务迭代速度	业务迭代速度一般或较快的部门/团队，对于代码安全问题处理带来的历史负担较轻，因此配合度相对较高。对于迭代速度较慢或基本处于维护状态的业务，修复安全问题带来的稳定性风险较大，并且修复过程中遇到的问题也更加复杂。因此迭代速度较快的部门/团队可能更加适合参与试点。	25%
技术栈复杂度	技术栈过于复杂，会导致试点过程中遇到的问题较多，而在试点阶段项目可能不会有足够多的人力去解决这些问题。技术栈复杂度可以通过对代码仓库的语言类型进行评估，如果该部门/团队下的大部分代码仓库都是属于公司主流的开发语言，那可能更加适合参与试点。	15%

5.3 试点宣贯准备

确定了试点范围，在正式开始试点之前我们要召集参与试点的相关方，通过启动会的形式对项目进行介绍以及对项目试点工作进行宣贯。目的是确保所有试点参与者都能充分理解试点项目的目标、流程、预期成果以及他们的角色和责任。

为了试点启动会能达到令人满意的效果，因此我们需要提前准备好详尽的宣贯材料。

5.3.1 关键点

在准备试点宣贯材料时，至少应该要包含以下内容。下面表格是材料中包含的内容、内容的说明以及内容应该着重关注的关键点。

序号	内容	说明	关键点
1	试点项目背景	讲清楚试点项目的背景信息，解释为什么会启动这个项目，以及项目的总体目标是什么	讲述项目背景时要站在更高的角度，让大家明确项目的重要性和预期，对公司和业务的长远影响
2	试点项目的具体目标	列出试点项目的具体目标，描述试点预期达到的成果，以及这些成果对试点参与方的收益	讲清楚对试点参与方的收益，让参与方明确
3	参与方的角色和责任	列出有哪些参与者、角色，以及他们的责任	确保每个人都清楚自己在项目中的作用和期望贡献
4	详细的操作说明	整理详细的操作流程和步骤，包括如何使用相关工具	包括常见问题的解答，帮助参与者解决可能遇到的问题
5	问题反馈	明确试点过程中遇到问题的反馈渠道	确保参与试点的员工出现问题时能够及时响应

5.4 试点启动会

准备好宣贯材料之后，就开始组织试点参与方召开启动会。通过这次启动会要让所有参与试点的角色都能清晰自己的角色和责任，了解在试点过程中具体要做哪些事情，会议后正式启动试点工作。

5.4.1 关键点

启动会开始之前，首先要确定所有需要参会的人员，包括参与试点部门/团队的管理层、技术专家、安全部门的管理者以及参与运营的相关同事，其次要确保所有参与者都被及时通知会议时间和地点，最后是准备详细的会议议程并按照议程完成启动会的召开。

5.4.2 会议议程

- 项目介绍；
- 目标和预期成果；
- 角色和责任分配；
- 项目流程和时间表；
- 答疑环节。

5.5 分批试点

5.5.1 分批设计

应该优先选择关系最好，对安全工作配合度最高的部门作为第一批试点，第一批试点的范围应该尽可能小，这样保证及时试点初期出现一些严重的流程和工具能力上的缺陷时，对用户造成的影响是可控的。

接下来，应该明确后续每一批的试点范围及试点时间。

5.5.2 试点工作计划

5.5.2.1 存量风险梳理

5.5.2.1.1 存量全量风险检测

试点对公司代码仓库中的全量风险进行检测：

- **治理范围**：全量的 GitLab 仓库及容器镜像仓库 harbor
- **接入检测**：
  - GitLab：通过 SCA 检测平台对接 GitLab API，配置一个有全局权限的 token 来实现
  - harbor：通过 SCA 检测平台对接 HarBor API（不同版本的 Harbor

的 API 略有差异。），配置一个具有全局权限的账号和密码来实现。

5.5.2.1.2 高优先级治理风险筛选

我们需要通过相对科学的风险分级策略，筛选出来高风险及高处置优先级的安全问题。

- **筛选策略**

  原则：如果被攻击成功导致系统中断、数据泄露等，造成损失越大的，越优先处理。

- **综合考虑如下点：**

(1) 漏洞危害程度：RCE、反序列化、注入、投毒等能直接拿到权限和数据的优先修复、SSRF 等次之。

(2) 利用复杂度和利用环境：优先处理互联网能攻击、利用成本低和漏洞可达的。其次是纯内网环境（如 dubbo）等攻击成本为中等的或更高的。

(3) 升级兼容性：无兼容性风险的可以优先升级，存在兼容性风险的需要全量回归测试，存在一定的风险。

(4) 业务重要程度：根据业务和应用重要程度进行排序，越重要越优先处理。

(5) 系统业务属性：分为对互联网暴露的和纯内网访问的。对互联网暴露的优先修复。

举例来说，struts、fastjson、log4j2、ruoyi 等重点风险组件需要尽快修复。

5.5.2.1.3 分批处置策略

根据筛选出来的需要高优先级处理的风险，我们需要分批分阶段的进行处置，避免一次性给业务带来太大的压力和负担，通常进行分批推进的时候需要重点关注以下几个方面：

- 优先策略：优先选择配合度高、业务重要性高、外网业务等；
- 确定责任人：必须要对每一批处置范围明确责任人；
- 确定处置时限：与责任人确定明确的处置时限及对应的处置措施（如纳入考核、强制下线、限制发布等）；

- 目标对齐：以上相关信息必须与责任人及支持运营的团队对齐，保障各方有效协同落地。

5.5.2.2 高危风险治理

5.5.2.2.1 相关责任人

在进行风险修复处置时，应明确该风险关联负责人，在进行项目负责人查询时，根据企业内部管理形式采用对应查询方式，例如：

- 通过被检查代码分支关联提交人定位相关项目负责人；
- 通过内部应用管理平台查询相关项目负责人。

5.5.2.2.2 处置流程

1. 明确风险处置的必要性（可根据高优先级治理风险筛选策略研判）；
2. 提供风险处置建议；
3. 提交漏洞修复工单并指派项目负责人；
4. 确认风险处置结果并验证；
5. 关闭工单并归档。

5.5.2.3 增量风险卡位

卡位的一个非常重要的价值是对不遵守安全策略的行为具备强行管控的能力，这样安全策略的实施落地才具备强有力的抓手。但是卡位的顺利落地需要慎重考虑对于现有业务流程效率的最小影响，通常需要考虑检测速度、检测准确率及例外情况的豁免措施等等，关于豁免措施，可以包括：漏洞细粒度白名单，用户自主误报反馈，以及紧急上线功能审批等，给与使用者一定的主动权，但是保留安全的审批权。

其中，检测速度和检测准确率需要和相关流水线的负责人进行协商确定，但是最终的准则都是对业务发布流程的体验不要有明显的损失，更不要阻塞现有的发布流程。

5.5.2.3.1 优劣势分析

- 优势

  a. 通过集成可以确保每一次代码构建都能够自动进行风险检测，及时发现项目存在的风险。

  b. 可以根据公司的管理制度规范对存在风险未修复的项目上线进行卡位，倒逼研发在上线前修复安全问题。

- 劣势

  a. 由于是在构建过程中进行安全检测，可能会增加项目整体构建时间。

  b. 项目构建环节风险发现相对后置，如项目存在必须处置安全风险，可能会导致项目上线推迟，如果漏洞修复成本较高会导致项目延期，同时容易引起研发反弹情绪。

5.5.2.3.2 卡位点选择

1. Jenkins

在 Jenkins 构建环节引入安全产品的安全检测能力对待构建项目进行安全检测，对于安全检测能力的调用一般可以通过以下方式：

a. 客户端工具集成（适用于安全产品有封装好安全检测能力的客户端工具的情况下）：

   (1) 安装客户端工具到 jenkins 构建服务器。

   (2) 在项目构建流水线中调用客户端工具对待上线项目进行检测。

   (3) 获取项目检测结果，结合安全基线进行风险卡位。

b. Open API 集成（适用于安全产品具备功能完善的 Open API，该方式对项目构建时长影响较大）

   (1) 在项目构建流水线中通过调用安全产品 Open API 上传待上线项目包进行安全检测。

   (2) 通过对应结果查询接口获取检测结果，结合安全基线进行风险卡位。

2. GitLab CI/CD

在 GitLab CI/CD 的流水线配置中，可以使用自定义脚本来调用第三方代码安全检测工具。

- 可以通过在.gitlab-ci.yml 文件中定义特定的阶段（stage）和任务（job），在其中执行安全检测工具的命令。
- 可以通过修改 GitLab Runner 配置的方式，在代码进入 CI 阶段前，进行自动开源风险扫描，并可选择是否设置卡点(此方案无需项目做任何变更即可完成接入)。
- 这种卡点方式需要用户自己配置.gitlab-ci.yml 文件，研发可以对这个配置进行删改。如果无法真正做到制度强管控，很难实现 100%卡点，所以如果采用这种卡位机制需要考虑一方面从制度管理上强管控，另外一方面也可以使用技术手段对.gitlab-ci.yml 文件是否存在及是否合理配置进行审计抽查。

注： 小米有此类的最佳实践，可参考：
https://mp.weixin.qq.com/s/bSFkkGSxCBHZ22VAh3XgPA

5.5.2.3.3 实施步骤

a. **增量风险记录**

(1) 根据 CI/CD 集成检测，对上线项目增量风险进行记录；

(2) 项目增量高风险组件推修（推修可参考高优先级治理风险筛选策略）；

(3) 分析项目风险情况，制定卡位处置策略，具体卡位的策略可根据企业的实际情况进行设计，这里也推荐一下涂鸦智能之前采取过的策略，就是当期发现的问题，下期发布的时候卡点，不卡当期，这个适合一开始推进研发效率影响比较大的时候。

a) **灰度试点**

(1) 根据试点项目重要程度分批上线卡位处置策略。

b) **全量卡位上线**

(1) 根据灰度试点情况调整卡位规则，全量项目卡位上线。

5.5.2.4 准入管控

5.5.2.4.1 准入管控方案设计

- **目的**

在组件引入阶段进行风险组件卡位，防止高危开源组件及投毒组件引入，同时避免内部自研二方组件风险扩散，降低风险后置处置成本，提升项目安全性。

- **优劣势分析**

a. **优势：**

(1) 安全左移至组件引入源头，降低风险后置处置成本。

(2) 突发 0day&投毒及时拦截，降低应急成本。

(3) 降低内部自研组件风险扩散。

b. **劣势：**

在项目构建拉取组件时会增加整体项目构建时间，但是基本上可控制在 5%以内。

- **方案设计**

a. 通过在内部私服仓库集成插件实现与检测服务端通信；

b. 在服务端配置安全准入规则，及处置策略；

c. 在研发本地进行组件拉取及项目构建拉取组件时进行安全验证；

d. 基于命中准入规则对应的处置策略对风险组件进行处置（审计记录/拦截并记录）；

e. 为临时紧急的业务需求设计豁免通道。

5.5.2.4.2 常见的准入管控实现方式

1. Nexus

a. 集成方式

(1) 通过 Nexus 网关插件实现网关能力集成；

(2) 基于内部安全基线配置准入规则；

(3) 基于准入规则对拉取组件进行安全验证并处置。

2. Jfrog

   a. 集成方式

      (1) 通过 Jfrog 网关插件实现网关能力集成；

      (2) 基于内部安全基线配置准入规则；

      (3) 基于准入规则对拉取组件进行安全验证并处置。

5.5.2.4.3 实施方案

1. 准入管控策略设计：明确范围

   a. 将内部现有禁止使用的组件版本纳入管理范围，加入黑名单；

   b. 通过代码仓集成对现有业务代码进行检测，建立资产台账及风险组件清单，与业务部门沟通评估需纳入管控的组件版本，并加入准入规则；

   c. 根据组件安全情况更新准入规则；

   d. 企业内部开发的通用组件（二方组件），也应该纳入管控范围，存在高危风险的二方组件应该限制在内部发布，包括应该维护内部组件的历史漏洞信息知识库。

2. 审计观察

   a. 针对纳入管控的组件版本可以先通过审计处置策略，记录规则命中情况及影响项目范围。

3. 通知整改

   a. 根据审计记录定位命中规则的项目，并通知相关项目负责人进行整改。

4. 豁免通道设计

   a. 在卡位正式上线前，还是要设计豁免的通道，包括申请豁免的流程及快速响应机制；

   b. 针对豁免的流程设计需要考虑让申请方承担责任，同时事后有监督整改的机制尽快缩小风险窗口。

5. 卡位上线

   a. 提前通知可能受影响的人群。

   b. 调整准入规则处置策略为拦截模式，并设定拦截提醒。

6. 异常反馈
   a. 建立异常反馈渠道，收集异常信息。
   b. 根据异常反馈信息调整策略。

5.5.2.5 应急响应

模拟当出现新 0day 时，基于 SCA 工具收集的 SBOM 信息进行排查

a. 0day 曝出；
b. 风险排查；
c. 应急处置：
   (1) 止损；
   (2) 问题修复；
   (3) 安全防护能力提升：针对此漏洞，添加 WAF 规则、xAST 检测规则、xDS 检测策略等。

5.6 收集反馈

- 设立反馈机制
  - 用户主动反馈：
    - 有奖反馈。
    - 关键人员访谈奖励。
  - 数据观测：
    - 漏洞修复率。
    - 漏洞修复时效。
    - CI/CD 接入后影响。
    - 准入拦截率。
    - 漏洞修复成本。

- 研发效率的影响：这个需要联合工程效率团队来统计。

5.7 成果总结和汇报

试点结束后，对试点过程中发现的问题和验证的成果进行总结，充分证明该运营方案的有效性。

- 准备总结报告和汇报材料
  - 技术方案的可行性：
    - 各试点任务的验证情况。
  - 预期的效果达成情况（结果和成本）：
    - 部门支持及配合情况；
    - 安全问题发现情况；
    - 安全问题处置情况；
    - 满意度。
  - 问题及改进：
    - 过程中发现的问题；
    - 改进的流程；
    - 待解决问题。
- 组织汇报会议：
  - 汇报对象；
  - 汇报内容；
  - 预期效果；
  - 下一步工作计划。

六、正式方案

根据第五章节中制定的初步方案，加上试点过程中发现的问题并对初步方案进行调整优化，形成正式方案。并在原始的初步方案中加入试点效果，以证明优化后方案的可行性，最终邀请项目相关方及公司相关管理层进行一次正式的项

目汇报。

七、制度规范发布

根据第七章节制定的初步制度规范，通过试点过程中发现的问题进行优化完善，形成最终的制度规范文件，并按照公司的制度发布流程进行正式发布。

7.1 制度规范正式审批发布

1. 提前与重要的评审相关方沟通，如运维负责人、研发负责人等，根据意见修改内容；
2. 召开制度评审会，再次同步被修改的部分制度内容给制度相关方，在评审会中获得相关方的公开认可；
3. 评审会结论应形成书面纪要，并通过邮件方式抄送全体参会人员；
4. 按照内部制度发布流程进行审批、公示和最终发布；

7.2 内部宣贯

1. 公告：通过内部公众号、消息推送、企业动态等平台，推送相关制度规范说明和解读文章；
2. 会议培训：联系业务技术负责人、研发人员、运维人员、安全工程师等人召开制度宣贯会议，针对其中容易被误解的点进行更正并介绍具体做法，重点强调要求和职责；
3. 组织线上考试：10 分钟内阅读完制度、考核对应人员的职责和核心条款。如在多少天内要修复中高危漏洞，如何联系反馈等。从而让对应人员都确认了解到条款，并且保障每一个人是知晓内部制度规范要求的。

八、正式上线推广

根据正式运营方案及制度规范要求落实开源安全治理工作，将开源安全治理工作全面推广至企业内部，确保开源安全治理的全面实施和有效执行，最终保障

项目取得优异的成果。

8.1 启动会准备

启动会主要是面向项目所有的参与方及目标用户人群，目标是为了拉齐大家对于项目价值的认知并明确各方的职责，为保证启动会正常召开，在召开启动会前，应做以下准备：

1. 准备会议材料：准备启动会所需各项材料；
2. 制定会议议程：会议主题、内容安排、时间安排、会议地点等；
3. 确定参会人员：明确参加启动会的人员范围，包括各部门负责人、关键利益相关方以及项目团队成员等；
4. 会前预沟通：与相关参会人员提前沟通，确保相关方能够提前了解会议沟通信息，并达成初步一致。

8.2 召开启动会

召开启动会旨在向相关部门及利益相关方介绍开源安全治理工作背景、目标、实施计划及实施方式，确保所有参与者对开源安全治理工作有清晰的认知和理解，明确各方参与者在整个开源安全治理过程中的责任及分工，保证开源安全治理工作能够在企业内部顺利展开并运营。

因此启动会应包含以下讨论内容：

a. **开源安全治理背景介绍：**可以以立项材料中阐述的内容为主，包括项目需求背景、目标等方面的内容。

b. **项目方案介绍：**介绍整体的项目方案，目的是让参会人员理解项目具体是要做什么、怎么做、预计达成什么样的目标、项目落地后达到什么样的效果，同时可分享一下试点阶段的成果，增强大家的信心。

c. **制度规范和流程解释：**针对某个特定流程，应讲述清楚具体流程的相关制度及流程中各角色需要承担的义务以及责任。

d. **实施计划：**明确开源治理工作最终目标和具体实施计划，明确实施过程

中各个环节参与方的责任与分工，让所参与者明确自己的工作内容及责任。

e. **项目里程碑**：依据实施计划，设定项目阶段里程碑，目的是让每个团队了解项目节奏，并具有时间的紧迫感。

f. **各团队需要支持的内容**：最后通过一个表格的形式例举出每个角色在项目中需要支持的内容。

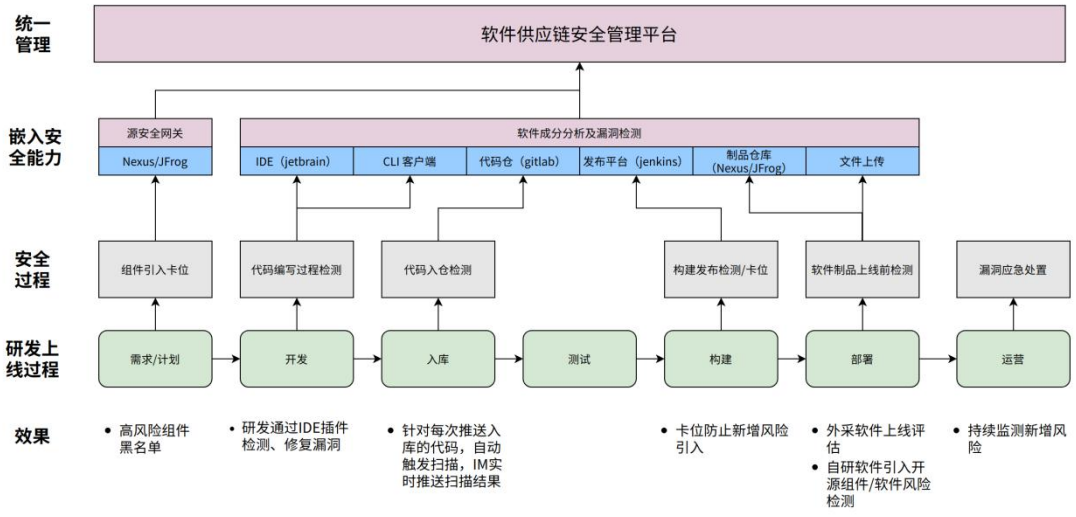
g. **关于成果及表彰**：介绍该项目运营过程中，对于积极参与配合项目开展的小伙伴会以月度、季度、年度的方式开展奖励计划。

h. **讨论与答疑**：如参会人员有疑问，需要进行充分讨论和答疑，保证会议结束后参会人员有明确的工作内容和工作目标。

8.3 正式推广治理

将开源软件安全治理相关的能力嵌入至软件开发及管理流程，逐步开展治理工作。

下图是关于治理能力与软件研发流程的一个结合示意图



下图是关于分阶段开展开源软件安全风险治理的一个逻辑流程示意图：

8.3.1 存量风险梳理

试点对公司代码仓库中的全量风险进行检测：

- **治理范围：**全量的 gitlab 仓库及容器镜像仓库 harbor。
- **接入检测：**
  - GitLab：通过 SCA 检测平台对接 GitLab API，配置一个有全局权限的 token 来实现；
  - harbor：通过 SCA 检测平台对接 HarBor API（不同版本的 Harbor 的 API 略有差异），配置一个具有全局权限的账号和密码来实现。

优先级分级策略：

- 筛选策略：

  原则：如果被攻击成功导致系统中断、数据泄露等，造成损失越大的，越优先处理。

  综合考虑如下点：

  (1) 漏洞危害程度：RCE、反序列化、注入、投毒等能直接拿到权限和数据的优先修复、SSRF 等次之。

  (2) 利用复杂度和利用环境：优先处理互联网能攻击、利用成本低和漏洞可达的。其次是纯内网环境（如 dubbo）等攻击成本为中等的或更高的。

  (3) 升级兼容性：无兼容性风险的可以优先升级，存在兼容性风险的需要全量回归测试，存在一定的风险。

  (4) 业务重要程度：根据业务和应用重要程度进行排序，越重要越优先处理。

举例来说，struts、fastjson、log4j2、ruoyi 等重点风险组件需要尽快修复。

8.3.2 高危风险治理

8.3.2.1 高危风险策略

关于企业高危风险的定级策略应该考虑业务重要性、是否具备外网暴露面、漏洞等级、许可证合规风险等级来综合评定。

关于安全漏洞的处置优先级，只考虑业务重要性、是否具备外网暴露面、漏洞等级：

- 业务重要性：将业务按照数据敏感程度及业务重要程度结合起来分为三级；
- 是否存在外网暴露面：是/否；
- 漏洞风险等级：
  - 强烈建议修复：

    CVSS>=7（高危以上）、有 POC 、存在可直接导致数据泄漏的漏洞类型 、利用条件为低。
  - 建议修复：

    (1) CVSS>=7（高危以上）、有 POC 的、存在可直接导致数据泄漏的漏洞类型 、利用条件为高。

    (2) CVSS>=7（高危以上）、有 POC 的、存在可直接导致数据泄漏的漏洞类型 、利用条件为中。

    (3) CVSS>=7（高危以上）、有 POC 的、不存在可直接导致数据泄漏的漏洞类型 、利用条件为低。

    (4) CVSS>=7（高危以上）、有 POC 的、不存在可直接导致数据泄漏的漏洞类型 、利用条件为中。

    (5) CVSS>=7（高危以上）、有 POC 的、不存在可直接导致数据泄漏的漏洞类型 、利用条件为高。
  - 可选修复：

    未命中以上条件的其他情况。

建议，最优先处理最高级别的业务 + 外网有暴露面 + 强烈建议修复的安全问题。

8.3.2.2 分批策略

	业务重要性	外网暴露面	安全漏洞处置等级
第一批次	A	是	强烈建议修复

第二批次	B	是	强烈建议修复
第三批次	C	是	强烈建议修复
第四批次	A/B/C	是	建议修复
第五批次	A/B/C	是	可选修复
第六批次	A	否	强烈建议修复
以此类推	1.优先考虑是否有外网暴露面 2.其次考虑是否强烈建议修复 3.再考虑业务等级		

8.3.2.3 分批推进方案

分批推进的方案，原则上是以代码仓库为基准单位，存量的代码项目风险治理一批，然后对这一批代码项目进行增量的风险卡位。

所谓的风险卡位就是在 CICD 流程中实现门禁，当代码项目进行编译构建发布时，检测是否存在强烈建议修复的开源组件漏洞，如果存在则编译失败，要求开发者修复完这些安全问题后再提交编译通过。以此方式倒逼软件开发者在软件开发环节中将这些严重的安全风险提前修复。

每一批次的存量风险，应该要求相关负责人在规定的时间内，主动整改已知的安全风险，如果不按期完成整改风险，将直接上线安全门禁。不过建议在规定时间到期时，应通过宣贯、沟通等方式，保障至少有超过 70%的安全问题被处理，剩下的 30%可以强制执行。

8.3.2.1 相关责任人

在进行风险修复处置时，应明确该风险关联负责人，在进行项目负责人查询时，根据企业内部管理形式采用对应查询方式，例如：

- 通过被检查代码分支关联提交人定位相关项目负责人。
- 通过内部应用管理平台查询相关项目负责人。

8.3.2.2 处置流程

1. 明确风险处置的必要性（可根据高优先级治理风险筛选策略研判）；
2. 提供风险处置建议；
3. 提交漏洞修复工单并指派项目负责人；
4. 确认风险处置结果并验证；
5. 关闭工单并归档。

8.3.3 增量风险卡位

8.3.3.1 确定范围

卡位的范围应该考虑从业务分级和风险分级两个维度来选定，同时和 11.3.2.3 中的分批推进策略保持一致。

8.3.3.2 确定卡位策略

8.3.3.2.1 卡位策略配置

卡位策略包括阻断和审计两种模式，首先准备好需要配置进行审计或者阻断的项目列表及漏洞列表范围，然后进行配置上线。

8.3.3.2.2 例外流程设计

通常需要设计例外流程，当某些业务被阻断编译构建，但是业务又需要紧急上线发布时，相关人员可以通过申请例外流程来临时放行，通过这样的方式可以避免重要紧急的业务需求被阻断。

8.3.3.3 卡位实现

因为卡位通常可能会对现有的业务流程造成影响，所以需要分四步走，保障整个卡位过程的丝滑：

8.3.3.3.1 确定卡位策略

8.3.3.2 章节已经明确了卡位策略，这里只需要复用即可。

8.3.3.3.2 实现方案

- 基于 Jenkins CI/CD 的实现方案：

在 jenkins 构建环节引入安全产品的安全检测能力对待构建项目进行安全检测，对于安全检测能力的调用一般可以通过以下方式：

a. 客户端工具集成（适用于安全产品有封装好安全检测能力的客户端工具的情况下）：

(1) 安装客户端工具到 jenkins 构建服务器；

(2) 在项目构建流水线中调用客户端工具对待上线项目进行检测；

(3) 获取项目检测结果，结合安全基线进行风险卡位。

b. Open API 集成（适用于安全产品具备功能完善的 Open API，该方式对项目构建时长影响较大）：

(1) 在项目构建流水线中通过调用安全产品 Open API 上传待上线项目包进行安全检测；

(2) 通过对应结果查询接口获取检测结果，结合安全基线进行风险卡位。

- 基于 GitLab CI/CD 的实现方案：

a. 在 GitLab CI/CD 的流水线配置中，可以使用自定义脚本来调用第三方代码安全检测工具。

b. 可以通过在`.gitlab-ci.yml`文件中定义特定的阶段（stage）和任务（job），在其中执行安全检测工具的命令。

c. 获取项目检测结果，结合安全基线进行风险卡位

8.3.3.3.3 开启审计模式

在明确卡位策略和范围后，可以先将此批范围的代码项目和漏洞级别配置审计模式，通过审计模式来观察当前还有多少项目存在的即将被阻断风险，并输出每日的审计记录清单，以此来预估未来卡位时会带来多大的影响。

8.3.3.3.4 限期整改

对这一批卡位可能影响的所有用户人群范围进行邮件及 IM 消息等各种方式的通告，一定要明确告知用户卡位的具体策略是什么？以确保大家知晓可能面临的卡位和应对措施。

- **通告方式：**邮件&IM&电话等，正式卡位前至少预留一个月的时间给大家提前整改，在启动时通知和临近卡位上线前 7 天、1 天再次通知；
- **通告内容：**明确且清晰的卡位策略，针对哪些项目范围及卡位的具体条件，比如漏洞缺陷等级等。卡位后的应对措施等，包括但不限于例外申请、问题咨询、自查整改等；
- **限期整改：对**前期审计过程中发现的可能导致阻断的所有用户行为，统计后发给对应的用户，要求其限期整改，并明确告知如果不整改可能导致的影响；
- **应对措施：**应考虑用户在被卡位后，可能出现的影响并给出应对措施，包括告知用户如何提前自查整改项目问题，例外申请等；
- **惩罚措施：**考虑到会出现少部分人不按期整改的情况，需要和业务方负责人对逾期不完成任务的情况进行处罚，首先参考公司相关制度规范、此外还包括公开通道、纳入绩效考核评分等。

8.3.3.3.5 上线卡位

当限期整改期满，且整改整体进度超过 70～80%时，可考虑直接上线卡位。如果整改期满，且整改整体进度低于 70%，那么应该分析是哪个环节出了问题，并积极调整后，促使整改进度超过 70%后，推进卡位的上线。

8.3.4 私有源准入准出管控

基于 CI/CD 的卡位流程是更偏事后的，此时代码已经开发完了准备发布上线，这时发现的安全漏洞修复成本非常高，所以很多企业考虑在研发过程中，甚至代码开发之前就能识别并管控风险。那么对企业使用的内部私有源（nexus、jfrog）进行管控就成了最有效的方式。

8.3.4.1 确定范围

卡位范围可参考 11.3.2.2，卡位范围可以分为哪些需要拦截、哪些需要审计的两类风险，需要拦截的通常至少是强烈建议修复范围的组件及版本范围，而需要审计的通常是危害严重但是又无法直接拦截的部分。

8.3.4.2 确定卡位策略

8.3.4.2.1 卡位策略配置

卡位策略包括阻断和审计两种模式，首先准备好需要配置进行审计或者阻断的组件及版本列表，然后进行配置上线，所有待阻断的行为都先发布审计。

8.3.4.2.2 例外流程设计

通常需要设计例外流程，当某些业务被阻断编译构建，但是业务又需要紧急上线发布时，相关人员可以通过申请例外流程来临时放行，通过这样的方式可以避免重要紧急的业务需求被阻断。

8.3.4.3 卡位实现

8.3.4.3.1 确定卡位策略

8.3.4.2 章节已经明确了卡位策略，这里就不赘述了。

8.3.4.3.2 实现方案

- 基于 nexus 的实现方案：
  a. nexus 集成具备与平台通信及阻断组件下载能力的插件；

b. 平台测通过调用 nexus API 实现与 Nexus 集成；

c. 平台测基于内部安全基线设置准入规则及处置策略；

d. 通过 Nexus 进行组件拉取时与平台交互验证准入策略命中情况，并进行处置。

- 基于 jfrog 的实现方案：

a. JFrog 集成具备与平台通信及阻断组件下载能力的插件；

b. 平台测通过调用 JFrog API 实现与 JFrog 集成；

c. 平台测基于内部安全基线设置准入规则及处置策略；

d. 通过 JFrog 进行组件拉取时与平台交互验证准入策略命中情况，并进行处置。

8.3.4.3.3 开启审计模式

在明确卡位策略和范围后，可以先将此批范围的组件及版本配置审计模式，通过审计模式来观察当前还有多少项目存在的即将被阻断风险，并输出每日的审计记录清单，以此来预估未来卡位时会带来多大的影响。

8.3.4.3.4 限期整改

对于 11.3.4.3.3 中开启审计模式后产生的日志进行统计分析，可以根据被审计的 IP 溯源分析到具体的员工和时间点，以此为依据来关联哪些员工的哪些项目将受到阻断之后带来的影响，

对这一批卡位可能影响的所有用户人群范围进行邮件及 IM 消息等各种方式的通告，一定要明确告知用户卡位的具体策略是什么？以确保大家知晓可能面临的卡位和应对措施。

- **通告方式：**邮件&IM&电话等，正式卡位前至少预留一个月的时间给大家提前整改，在启动时通知和临近卡位上线前 7 天、1 天再次通知；
- **通告内容：**明确且清晰的卡位策略，针对哪些组件版本范围及卡位的具体条件。卡位后的应对措施等，包括但不限于例外申请、问题咨询、自查整改等；
- **限期整改：**对前期审计过程中发现的可能导致阻断的所有用户行为，统计后发给对应的用户，要求其限期整改，并明确告知如果不整改可能导致的

影响；

- **应对措施：**应考虑用户在被卡位后，可能出现的影响并给出应对措施，包括告知用户如何提前自查整改项目问题，例外申请等。

8.3.4.3.5 上线卡位

当限期整改期满，且整改整体进度超过 70～80%时，可考虑直接上线卡位。如果整改期满，且整改整体进度低于 70%，那么应该分析是哪个环节出了问题，并积极调整后，促使整改进度超过 70%后，推进卡位的上线。

8.3.5 应急响应

模拟当出现新 0day 时，基于 SCA 工具收集的 SBOM 信息进行排查：

a. 0day 曝出；

b. 风险排查；

c. 应急处置：

(1) 止损；

(2) 问题修复；

(3) 安全防护能力提升：针对此漏洞，添加 WAF 规则、xAST 检测规则、xDS 检测策略等。

8.4 例行运营质量管理

例行运营质量管理的核心逻辑是：

1. 让所有参与该项工作的人及时知晓自己的工作成果及价值，并保持积极性；
2. 让所有关心这个项目成果的人及时知晓进度和是否存在项目失败风险，保障最终项目成功；
3. 让所有参与该项目且做的不好的人及时知道自己的问题并改进；

4. 通过数据运营及时观测和发现项目过程中存在的问题，及时调整并保障项目成功。

8.4.1 关键指标设计

8.4.1.1 业务价值指标

1. 线上业务增量已知风险处置率：可以按风险级别统计一下上线前整改情况（不同级别的整改率及时效）；
2. 线上业务增量未知风险处置时效：主要是 0day 漏洞应急处置时效性；
3. 线上业务存量风险整改率。

8.4.1.2 能力建设指标

1. 开源安全治理能力覆盖率：覆盖的线上项目/总线上项目；
2. 开源安全漏洞处置效率：（各级别，各环节）单漏洞平均处置时长。

8.4.2 过程运营质量监控

8.4.2.1 能力覆盖度监控

- GitLab 代码仓库覆盖率：线上有效代码库；
- 研发 IDE 插件覆盖率：出现过漏洞的研发；
- 私有源的网关覆盖率。

8.4.2.2 过程效果监控

- 员工满意度；
- 漏洞处置时效性；
- 漏洞修复率；

- 网关阻断率；
- 上线卡位阻断率；
- 新项目漏洞检出量；
- 系统稳定性评估；
- 研发效率。

8.4.3 过程运营

8.4.3.1 周报&月报

8.4.3.1.1 面向项目管理者

每月/周向项目管理者，包括前期项目立项过程中的主要决策者及项目组实际管理人员，汇报项目进展：

- 项目当前进展及是否符合预期：这里需要结合阶段性目标，来展示当前关联的进度，最好量化呈现；
- 重大成果展示：不定期展示该项目治理过程中产生的重大成果；
- 项目下一阶段计划：下一阶段的工作计划及目标也需要量化；
- 项目当前运行潜在的风险及应对方案；
- 当前需要的支持和解决不了的问题。

8.4.3.1.2 面向项目参与者

- 每周/月向与项目相关的全员发布项目进展及成果，并鼓励和感谢大家的参与
- 对及时处理相关问题的员工及时表示肯定和鼓励
- 对于处理问题不及时的员工及时提醒问题并明确告知后果

8.4.3.2 奖惩运营

- 月度/季度对做的好的员工及团队进行表彰，包括全员通告成绩、颁奖等方式。
- 月度/季度对做的不好的员工及团队进行处罚，包括一定范围内的成绩晾晒、通知对应上级领导等；严重到违反公司相关制度的应该根据公司的制度规范进行处罚。

九、收益管理

9.1 收益评价指标

开源安全治理项目的收益可以从以下几个维度来定指标，按照价值排序：

- 合规满足；
- 降低被攻击风险；
- 提效；
- 提升企业影响力。

9.1.1 合规满足

合规满足分两类，一类是直接的合规满足，主要是指明确的合规要求指向开源安全治理，包括开源软件漏洞治理或者开源许可证合规，比较典型的是金融行业四部委发的开源治理要求的文以及银行所属的监管单位明确发的一些关于开源治理的要求，除此之外，一些涉及出海业务的公司的甲方企业有非常明确和强制的开源许可证合规审查要求；另外一类是间接的合规满足，比如 hvv 的防御需求、数据安全合规满足等，hvv 近年来主要的攻击手段就包括开源组件的漏洞攻击，而针对数据窃取和数据加密勒索的攻击大多数攻击手段都是来自开源组件安全漏洞。

- 直接合规满足：
  - 行业监管单位关于开源安全/软件供应链安全的检查及整改要求的满足情况；

- 企业服务的甲方及关联合作方对交付物的开源许可证合规满足审查；
- 公司有对外开源软件或者对外发布软件的场景，需特别关注发布软件的开源许可证合规；
- 间接合规满足；
- hvv 防止因为开源组件/软件漏洞攻破风险，降低 xx%风险；
- 以《网络安全法》为背景的合规满足，其中主要是涉及事件处置、漏洞管理及数据安全保护相关的条款满足；
- 找出企业过去或者未来涉及到的信息安全相关监管要求，从中梳理出与开源安全治理间接关联的事项。

关于合规满足的评价指标的制定，一方面有明确直接的监管要求的，直接将指标定为满足或者不满足即可，另外一方面如果是间接监管满足的，建议从合规风险的量化角度来评估，比如不合规潜在风险从 80%降低到 10%，具体的数据量化可以从过去的抽样风险事件数/漏洞数到未来治理后的此类同口径数据的变化。

9.1.2 降低安全风险

建议将过去一定时间范围内的（比如三年）可带来或者已经带来明确影响的安全事件进行统计分析，梳理出其中由开源组件/软件相关安全及合规问题引起的数据。然后计算经过阶段性治理后，此类的事件由多少可以在上线前规避，有多少是上线后可以通过有效机制控制风险的。最终的效果就是根据过去三年的安全事件统计参考：

- 核心指标一：通过开源安全治理，上线前可解决的有损安全事件比例下降 60%（示例），其中 xx 核心业务、xx 核心业务的线上数据泄露风险下降 80%；
- 核心指标二：通过开源安全治理，线上业务出现的安全事件可主动有效控制风险的占比提升 70%（示例），线上业务风险敞口时间从 3 天下降至 1 天；

9.1.3 提升效率

统计做这个项目之前和之后，某核心业务单个应用系统的安全评估时长从多少下降至多少，保障核心业务安全性的同时，降低了安全评估的耗时，提升了项目发版效率。

- 核心指标一：在保障线上安全的情况下，核心业务的项目发版用时下降90%（含安全评估及安全问题修复用时）；
- 核心指标二：核心业务上线前安全保障流程覆盖率从 10%提升至 100%。

9.1.4 提升企业影响力

影响力可以从用户安全体验、技术品牌、专利、监管影响力等方面展开：

- 用户安全体验：如果公司有对外开源软件可以用这个，提升了开源项目安全性，解决了多少安全问题；
- 技术品牌：技术社区评奖、顶级技术会议投稿等；
- 专利：开源安全治理相关技术的专利申请数；
- 监管影响力：监管支撑工作、监管部门优秀案例评选等。

9.2 收益成果运营

9.2.1 公司内评优

阶段性根据项目成果应该及时申请公司内部的评优活动，这是对于项目组成员很好的一种激励和反馈。

9.2.2 业界分享

阶段性成果及经验可在业界交流论坛及技术大会上进行分享，提升个人及公司的行业影响力，并与同行交流各自经验，不断优化项目流程，提升治理效果。

9.2.3 行业优秀案例评选

参与信通院、公安部、各行业组织协会的优秀案例评选，提升企业及个人影响力。便于后续持续开展工作。

9.2.4 专利申请

将项目中的最佳实践及优秀创新在公司内部申请专利，提升企业影响力及行业竞争力。

十、附录

附录中主要包含各企业在开源安全治理实践中会用到的一些具体资料和模板，包括制度文档、立项 PPT、漏洞分类分级标准等。这些资料包通常具有企业属性，部分内容涉及敏感，所以采取定向发送的方式。

10.1 参考资料获取方式

联系时注意备注公司名-姓名，说明开源安全治理最佳实践资料需求，以下方式半个小时内可获取资料。

方式一：微信添加运营人员获取

方式二：发送邮件获取

hi@murphysec.com

方式三：电话咨询获取

400 180 9568

10.2 项目调研参考资料

资料名称	说明及用途	获取方式
近几年行业开源安全事件列表	调研阶段，论证企业在开源安全上存在的潜在风险	非公开，参考 10.1 的联系方式快速获取
各行业同行开源治理方案调研报告	含金融、互联网、运营商、智能制造、能源等八大行业一些典型公司调研报告，用于调研及立项阶段，向公司管理层说明同行做法	非公开，参考 10.1 的联系方式快速获取
POC 测试方案及测试样本	含各开源治理需求场景的 POC 测试方案及靶场典型样本，样本经典且权威具有代表性	非公开，参考 10.1 的联系方式快速获取

10.3 项目立项参考资料

资料名称	用途	获取方式
立项汇报 PPT 模板	立项汇报时，用来捋清楚汇报思路	非公开，参考 10.1 的联系方式快速获取

10.4 系统上线参考资料

资料名称	用途	获取方式
部署文档	用于一键脚本式快速完成服务的部署	非公开，参考 10.1 的联系方式快速获取

10.5 制度规范参考资料

资料名称	用途	获取方式

企业开源安全治理管理制度	含金融、互联网、运营商、智能制造、能源等行业内部的开源安全治理相关制度	非公开，参考 10.1 的联系方式快速获取

10.6 试点工作参考资料

资料名称	用途	获取方式
试点宣导 PPT 模板	启动试点工作时，用来做内部宣导的	非公开，参考 10.1 的联系方式快速获取

关于墨菲安全

墨菲安全是一家为您提供专业的软件供应链安全管理的科技公司，围绕着开源组件安全风险治理、资产管理及漏洞投毒应急响应、高危风险组件准入管控、开源组件许可证合规风险治理、商业软件安全风险治理等场景，向企业中研发工程师、安全工程师、法务人员等角色提供简单易用的安全产品。

产品形态包括：软件成分分析(SCA)、资产管理及漏洞预警、源安全网关、静态代码扫描(SAST)，丰富的安全工具助您打造完备的软件供应链安全能力。

旗下的安全研究团队墨菲安全实验室，专注于软件供应链安全相关领域的技术研究，关注的方向包括：开源软件安全、程序分析、威胁情报分析、企业安全治理等。公司核心团队来自百度、华为等企业，拥有超过十年的企业安全建设、安全产品研发及安全攻防经验。

版权声明