BLACKDUCK®

保护软件供应链安全的五个考虑因素

Black Duck 在年度《开源安全与风险分析》(OSSRA) 报告中指出，开源和第三方软件的使用量持续快速增长。这意味着在软件开发生命周期中，您的开发团队无疑会以某种方式并在某种程度上使用开源软件。

如果您是软件供应商，那么，您就需要依赖开源的第三方和商业组件来构建应用，进而成为供应链中的一环。不过，需要注意的是，这条供应链并不仅限于您的团队，它还涉及到软件开发和交付活动之外的所有相关方，一直延伸到应用的最终用户。

您的供应链涵盖了与应用直接相关或在其组装、开发和部署中扮演某种角色的所有各方，还包括您的开发团队自己编写的专有代码和组件，以及用于构建和向最终用户交付该软件所使用的基础架构。

我们可以借助传统的供应链示例来更清晰地了解软件供应链。我们以汽车制造业为例。该行业的供应链从原材料供应商开始，他们为零部件制造商提供金属、棉花、皮革和木材等原材料。这些零部件制造商将原材料加工成螺丝、织物、螺母和螺栓等零部件，然后再将这些零部件运送到汽车部件和系统制造商那里，由他们制造出组装汽车所需的部件（如发动机和变速器）。最后，这些部件将被运送到原始设备制造商那里，由他们在装配线上组装汽车，并卖给消费者。

如果您能意识到这个制造过程涉及到无数环节，便会知道很多地方都可能存在风险。制造这辆车所用的零部件质量如何?汽车制造商组装这些零部件的工艺如何?这辆汽车的实际驾驶性能如何?它如何应对突发事故和恶劣驾驶条件等情况?

对于供应链，可以肯定的一点是，这个过程中涉及的所有组件、人员、活动、材料和程序都会对最终产品及其用户产生影响。任何环节中的薄弱点都会带来风险，而降低这种风险的唯一方法就是完全了解整条供应链。软件供应链也是如此。

简言之，如果不能完全了解供应链，企业就无法准确或全面地管理风险。面对如此众多的活动部件，只要漏掉一个未识别的薄弱点或设计缺陷，就会为攻击者创造可乘之机。

对于供应链而言，整个过程所涉及的每个组件、人员、活动、材料和程序都会对最终产品及其用户产生影响。

软件供应链现状

Capterra是Gartner旗下的在线市场调研公司。Capterra报告指出，过去一年中，有超过五分之三 (61%) 的美国企业受到了软件供应链威胁的直接影响。该报告指出，开源软件是供应链风险的一个主要来源，94%的美国公司以某种形式使用了开源软件。

显然，不安全的开源软件对软件开发行业内外的企业都有影响。例如，广泛使用的 Apache Log4J 实用程序中的0-day漏洞允许攻击者在易受攻击的服务器上执行任意代码。

但是，开源软件并不是供应链完整性的唯一威胁。SolarWinds的入侵是由一个泄露的密码引起的，黑客利用该密码进入系统，并通过例行更新来访问数千名客户的敏感数据。在CodeCov的供应链入侵中，黑客利用他们在Docker镜像中发现的秘密安装了一个后门，窃取了客户数据。

为了应对安全漏洞的增加，美国颁布了一项行政命令 (EO 14028)，针对与联邦政府有业务关系的企业如何保护其软件安全提出了具体方针。这项旨在增强美国网络安全状况的命令促使全国各地的公司重新审视其安全实践，而不限于命令所指定的企业。

这项命令还引发了对软件安全实践的集体审查。EO 14028 的一项主要内容是建议企业改进软件供应链。该命令还对现有软件安全实践添加了另一层考虑因素，要求企业在制定安全措施时，要从供应链安全的视角审视现有活动。其结果是在规划安全措施时必须满足更多的要求。

软件供应链安全的主要考虑因素

从根本上说，保护供应链的任何努力都必须考虑如何保护应用不受上游风险的影响，以及如何防止产生下游风险。

为了帮助您通过更简洁的方式开始处理供应链安全问题，Black Duck 确定了有关供应链安全活动的五个考虑因素。通过回答这五个问题，您可以深入了解自己在供应链安全方面的成功和疏漏，并为您的安全活动提供信息。

问题 1

您使用的开源软件是否安全?

了解开源风险

对于确保供应链安全性而言，开源软件的普遍存在是需要考虑的一个基本问题。虽然开源的风险与专有代码差不多，但如果不能充分保护它，会给您的企业整体安全性带来巨大的风险。您的开发人员很可能在其创建的几乎所有应用中都使用了开源软件，这意味着您的应用中有很大一部分不是开发团队自己编写的代码。年度OSSRA报告证实了这一点：该报告调查的几乎所有代码库中都包含开源代码，并且开源代码几乎占到这些代码库中全部代码的80%。

顾名思义，开源代码来自不受控制的地方：企业外部的开发人员。保护供应链意味着要始终保持对应用组件的可视性，尤其当它来自外部时。

需要注意的是，您的开发人员在其工作中可能有意或无意地使用了开源代码，您的供应商也是如此，他们的开发人员可能在不知不觉间将开源代码纳入到了项目中。虽然这本身并不是一件坏事，但它确实为上游风险进入应用提供了途径。您有责任跟踪供应链中的开源组件、许可证和漏洞及其相关风险。但考虑到这项工作的规模，手动跟踪是不够的，而且太耗费人力。

了解法律风险

因许可证违规和冲突而引发的法律问题，很容易转化成并购、供应商纠纷和分销问题，只不过相对安全风险而言，这些法律风险并不是那么广为人知。对于软件供应商和分销商来说，由于供应链漏洞引起的法律风险会对公司的声誉和财务安全构成威胁。

开源软件的存在导致知识产权的识别和监控变得相当模糊，因此，对开源软件管理不善是引发法律风险的常见原因。Cisco因收购了含有违反开源软件许可协议的固件而与自由软件基金会之间那场轰动一时的诉讼案就是一个例子。因为继承许可证冲突，Cisco受到了本可以避免的法律和声誉影响。这再次凸显了供应链中的任何事物都有可能影响到整个运营。将软件风险视为业务风险，并相应地进行处理是至关重要的。

您有责任跟踪供应链中的开源组件、许可证和漏洞及其相关风险。

了解运维风险和已知漏洞之外的风险

当团队使用过时的组件、近期未更新的组件，或者无足够庞大的开发者社区积极维护的组件时，就会引入运维风险。除了带来代码质量、可靠性和可维护性问题外，运维风险也是引发安全风险的途径。如果没有开发人员去发现和修复项目中的bug，就没有开发人员去发现、披露和修复安全缺陷，尤其是在使用了声誉和口碑不佳或不明确的组件开发软件时。此类项目很容易被攻击者利用。

风险还可能以恶意包 (malicious package) 的形式出现，这是一个伪装成合法软件的恶意软件包。一旦被激活，攻击者便可以利用它们通过开源代码库或第三方组件渗入软件供应链，执行有害操作。恶意包会对应用的完整性和安全性造成严重的威胁。一旦恶意包中的恶意软件感染了系统，它就有可能窃取敏感数据、禁用安全软件、修改或删除文件，甚至接管整个系统或网络，传播到其他设备，进一步增加破坏力。

问题 2

您是否需要提供软件物料清单 (SBOM)?

软件可以通过多种途径进入应用。例如，软件可通过以下途径有意或无意（即传递依赖）添加到应用中：

- 包管理器
- 复制/粘贴和AI生成的片段
- 缺乏包管理器的语言，如C/C++
- 容器镜像、动态链接库和其他的第三方二进制文件或库

所有这些都会带来巨大的风险面和模糊不清的视图。如果没有一个完整、动态的视图来显示应用的组成，那么，您自己、您的供应商或消费者都将不能自信地确定您所面临的风险。维护软件物料清单 (SBOM) 是一种最佳实践，也是成功的供应链安全计划的基石。SBOM有时也是强制性的监管合规要求。SBOM本质上是代码库的“成分表”，对您了解软件组成至关重要，但为了保持准确性和完整性，您需要在适当时间对其进行编译。

仅通过分析包管理器来梳理依赖关系是不够的，仅依赖几天、几周或几个月前编译的SBOM也是不够的。要使SBOM有效，您必须通过隐藏在代码中的依赖项来不断地填充它，与语言、依赖类型或版本无关。而且SBOM中不应只包括开源代码，定制和商业代码也应包含在内并予以跟踪。

以Apache Log4J 0-day漏洞为例，拥有最新SBOM的企业能在漏洞披露后几小时内就确定其风险并开始风控活动，无需回过头去扫描整个应用组合。

如果没有一个完整、动态的视图来显示应用程序的组成，那么，您自己、您的供应商或您的客户都将无法自信地确定您所面临的风险。

问题 3

您编写的代码是否安全?

由于大部分的应用代码都是开源代码,因此开源攻击面占到总体攻击面的一大部分,这一事实不足为奇。但是,确保开发人员编写的代码能够有效保护敏感数据和系统免受网络攻击仍然至关重要。

无意中编码到应用中的安全缺陷和漏洞会使诸如缓冲区溢出、SQL注入和跨站脚本等攻击有机可乘。一旦系统被黑客侵入,这些安全缺陷就会将敏感数据暴露在危险之中。恶意攻击者可能会利用这些弱点注入恶意代码,并借此渗透到由运行该软件的组织所维护的系统中。

为了避免这些问题,您通常要进行代码审查,希望通过多人检查源代码来发现潜在的问题。然而,大多数软件开发者并没有接受过安全编码培训,而且许多安全漏洞在手动代码审查中很难被发现和追踪。因此,您应借助能够自动审查代码控制流和数据流并找出安全漏洞的静态分析解决方案,它是确保您可以相信未向下游引入任何风险的必备工具。

即便如此,您还是应该始终专注于持续提高产品的安全性。避免安全弱点的最佳方法就是从一开始就避免它们产生。您应为开发者提供安全编码培训,并培养安全倡导者 (security champions) 来推动建立供应链安全文化。

无意中编码到应用中的安全缺陷和漏洞为诸多攻击打开了大门,例如缓冲区溢出、SQL注入和跨站脚本等。

问题 4

您的开发和交付基础架构是否安全?

随着数据存储需求不断增长,部署期限越来越短,而且快速扩展变得空前重要。为了应对这些挑战,软件行业日益依赖云技术来驱动其应用。这种云原生方式要求您采用支持可扩展性和敏捷性的应用部署方法,这正是容器化和基础架构即代码 (IaC) 的用武之地。

了解哪些软件被打包到容器中

容器可使应用或微服务的部署、修补和扩展变得更加快捷。容器还可以确保应用或微服务在不同的操作系统和硬件平台上提供一致的性能。这使得容器化成为云原生方法的理想选择,但它也给供应链带来了额外的威胁入口。

一般来说,开发人员在容器化应用时首先会创建一个基础镜像(通常是开源的),以此为基础构建容器化应用。首先,开发人员会在这个基础镜像之上添加额外的第三方和自定义代码层,然后在最终镜像被执行时将其合并为单一文件系统,以二进制文件表示。基本的软件组成分析工具假设层中的文件是通过包管理器(如YUM)添加的,分析工具将使用它们来确定软件的组成。

但这并不是完全有效。当使用Dockerfile中的某些命令(如ADD、COPY或RUN)向层中添加文件时,并不涉及包管理器。只有对容器镜像进行真正的二进制分析才能检查组件签名,识别出任何来源的开源组件,以及容器镜像内部的任何敏感数据。

容器化是云原生方法的理想选择,但它也给供应链带来了额外的威胁入口。

使用基础架构即代码进行安全部署

如今，底层基础架构的配置、管理和调配方式与云平台的运行方式密不可分。为了满足虚拟化和容器编排，服务器现在以一种临时的方式运行，根据需要按规范构建。这些规范由工作团队使用Terraform和Ansible等软件来编制，以实现服务器的自动运行。但这些规范也可能引入安全缺陷，并且这些安全缺陷可能会随应用的部署规模而倍增 — 应用的部署次数每周可能从数万次到数十万次不等。

例如，因未经授权的提权或基础架构配置中的网络暴露而造成的无意和有意后门并不是新的风险。新的问题是谁负责构建和组装服务器配置。这曾经是IT运维工程师的工作，他们在安全配置服务器和预测威胁方面接受过培训，经验丰富。这项责任后来转给了开发团队，由他们使用基础架构即代码（IaC）为正在构建的应用轻松指定部署配置。在这方面缺乏经验可能会导致错误和疏忽，使攻击者更容易渗透到基础架构中，从而进一步靠近应用的用户、操作人员和数据。

IaC之所以如此有价值，部分原因在于它是由代码组成的。因此，它的编写和读取方式、保存和版本化方式，以及分析和审查方式，都与代码类似。正如我们应该使用静态分析工具来分析应用源代码中的安全弱点一样，我们也应使用该工具来扫描IaC文件，以发现因开发人员缺乏培训或经验而无法发现的代价高昂的配置错误。

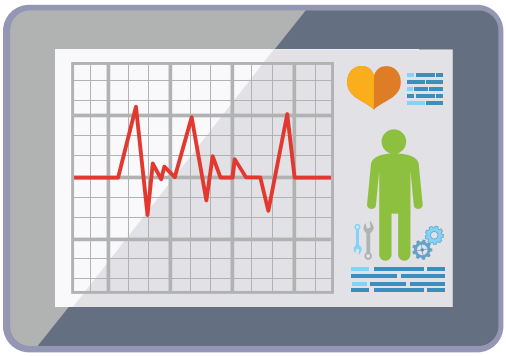
问题 5

您是否在受监管的行业创建或使用软件？

监管机构通过实施新标准来帮助管理软件供应链的复杂性。作为监管合规的重要组成部分，SBOM已经逐渐成为各行各业的强制性要求，以应对如下风险：

- 过时的组件 — SBOM可以突出显示那些已经停止支持或存在已知安全漏洞的组件
- 未经授权的组件 — SBOM可以显示那些未经组织批准、可能会带来许可或安全风险的组件
- 不符合法规要求 — SBOM可以验证软件是否符合相关的安全法规和标准

美国食品药品监督管理局（FDA）规定，从2023年3月起，所有使用软件的医疗设备都必须创建并维护SBOM。开源软件在医疗设备中使用广泛，Linux是医疗设备系统中最常用的开源组件之一。如果您是基于Linux开发软件，那么，您的软件中很可能还包含了其他开源组件，如中间件和前端框架，这些都必须通过SBOM定期向FDA汇报。



美国食品药品监督管理局所有使用软件的医疗设备都必须创建并维护SBOM。

同样，汽车行业的原始设备制造商 (OEM) 也需要详尽的SBOM来了解其供应商的代码。目前已经出台了一些法规和标准（后续还会陆续出台更多的法规和标准）要求OEM监测其车辆的网络安全风险，并在“合理的时间内”消除漏洞。 一项研究显示，汽车软件栈中有50%到70%的软件基于开源代码，所有这些都必须在SBOM中列出。

通过SSDF评估将安全性融入软件开发生命周期中

为了提高联邦政府的网络安全水平，美国国家标准与技术研究院 (NIST) 开发了“安全软件开发框架” (SSDF)。SSDF现已成为《第14028号行政命令》(EO) 中的一项要求。最终，联邦政府获取的所有软件（免费软件或内部开发的软件除外）都将必须遵守SSDF。

简言之，SSDF是一套旨在将安全性融入到软件开发生命周期中的最佳实践和指导方针。它为开发安全的软件提供了一种结构化方法，将安全考虑因素融入到整个开发过程中的每一个阶段。

SSDF侧重于主动的安全措施，倡导“从源头保障安全”（BSIMM项目中提出的一个概念），而不是把安全问题当作亡羊补牢之举。它强调了考虑安全需求、进行威胁建模、执行安全编码实践、执行适当的安全测试以及在整个软件生命周期中维护安全性的重要性。

下一步

结合本文提出的问题来检查您的现有安全活动、策略和程序，将有助于您找出需要注意的薄弱的供应链安全领域。虽然供应链安全性看起来可能非常复杂，但如果能将其分解为清晰简洁的考虑因素，并使用适当的工具和实践来处理已知缺陷，则供应链安全性是可以实现的。

如想了解有关供应链安全保护的更多信息，请
查看Black Duck如何能够帮助您端到端地识别和管理风险。

关于 Black Duck

Black Duck与众不同

Black Duck® 提供业界最全面、最强大、最值得信赖的应用安全解决方案组合。我们拥有无与伦比的专业知识和经验，来帮助世界各地的组织机构快速保护其软件，在其开发环境中高效集成安全性以及使用新技术进行安全创新。作为软件安全领域公认的领导者、专家和创新者，Black Duck拥有您构建可信软件所需的一切。如预了解更多信息，请访问www.blackduck.com。