SYNOPSYS®

[2023]
开源安全和风险分析报告

[目录]

[简介]

关于2023年开源安全和风险分析报告与新思科技网络安全研究中心(CyRC)

欢迎阅读2023年第8版《开源安全和风险分析(OSSRA)报告》。今年的OSSRA提供了新思网络安全研究中心(CyRC)对商业软件中的开源安全性、合规性、许可和代码质量风险当前状态的年度深入研究。我们分享了这些调查研究结果，以帮助安全、法律、风险和开发团队更好地了解安全和许可证风险状况。新思科技网络安全研究中心(CyRC)为本报告提供了数据。该中心的任务是发布安全建议和调研报告，以帮助企业更好地开发和使用安全的高质量软件。

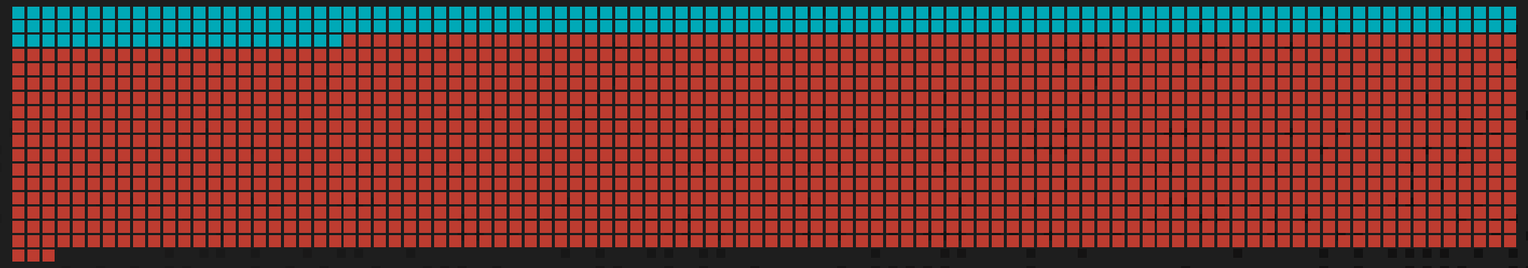
OSSRA年度报告代表CyRC从上一年数据中得出的结论。因此，我们的2023年报告显示的是2022年的数据。在2022年，CyRC对来自17个行业的超1,700个商业代码库的匿名调查结果进行了研究。我们的审计服务团队每年为客户审计数千个代码库，主要目的是识别并购(M&A)交易中一系列的软件风险。尽管2022年经济前景不明朗，科技领域的并购也相应放缓，但审计代码库的数量依然可观。

近20年来，新思科技Black Duck®软件组成分析(SCA)产品团队和CyRC审计服务团队一直在帮助世界各地的安全、开发和法律团队加强其项目的安全性和许可证合规。Black Duck SCA使企业能够识别和跟踪开源代码，并在其现有的开发环境中集成自动化的开源实施策略。Black Duck审计通常在并购交易背景下进行，涵盖软件风险的方方面面。该审计还提供全面的、高度准确的软件物料清单(SBOM)，涵盖企业应用中的开源代码、第三方代码、Web服务和应用编程接口(API)。审计服务团队依靠Black Duck KnowledgeBase™知识库的数据识别潜在的许可证合规与安全风险。该知识库由CyRC创建、管理和多年积累，存储了来自2.8万多个开源代码仓库超610万个开源组件的数据。

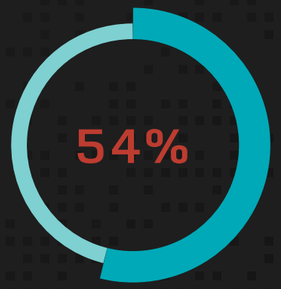
无论您经营什么行业，或者您在企业安全和风险管理方面扮演什么角色，OSSRA持续强调，日益普及的开源软件推动着业务发展，同时也存在无法进行有效管理的困难。我们每年都在强调，开源软件是我们今天所依赖的每个应用程序的基础。因此，有效地识别、跟踪和管理开源代码对于成功的软件安全计划至关重要。本报告提供了关键的建议和洞察，以帮助开源软件的开发者和使用者更好地了解开源生态系统以及如何对其进行负责任的管理。

无论您经营什么行业，OSSRA持续强调，日益普及的开源软件推动着业务发展，同时也存在难以进行有效管理的困难。

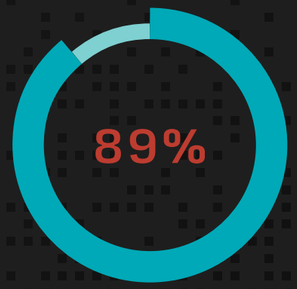
[概述]



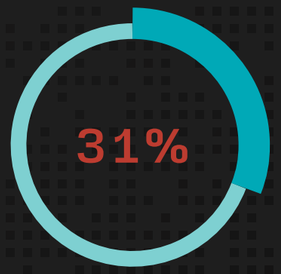
2022年审查的**1,703**个代码库中，**87%**包括安全和运营风险评估。



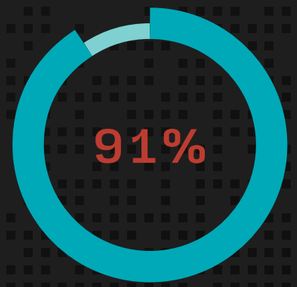
54%的代码库存在许可证冲突



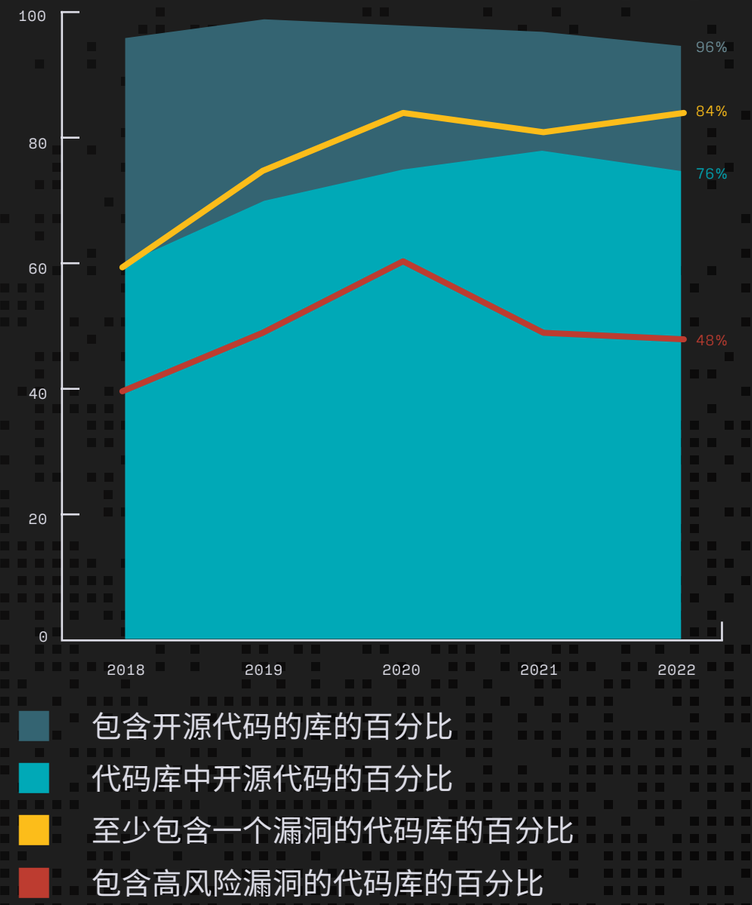
89%的代码库包含至少已过期四年的开源代码



31%的代码库包含没有许可证或使用定制许可证的开源代码



91%的代码库包含两年内未更新的组件



按行业划分的开源代码使用情况

术语

代码库

组成应用程序或服务的代码及相关的库。

二进制分析

静态分析的一种，用于在无法访问源代码时识别应用程序的内容。

Black Duck Security Advisory (BDSA)

由CyRC安全研究团队研究和分析的关于开源漏洞的详细、及时、一致的信息。BDSA为新思科技客户提供了开源漏洞的早期预警通知和升级/修补指导。Black Duck Security Advisory 的漏洞库超越了美国国家漏洞数据库(NVD)，提供增强的数据、更为完整和准确的信息，从而用户能够获得漏洞预警和全面的洞察。BDSA提供当天通知、实际可操作的漏洞消减指导和变通解决方案信息、严重性评分和参考信息等。

软件组件

开发人员可以添加到其软件中的预先写好的代码。软件组件可以是日历函数等实用程序，也可以是支持整个应用程序的综合软件框架。

依赖项

当某个软件组件被其他软件使用时，也就是说当这些软件依赖于该组件时，该软件组件就变成了依赖项。任何给定的应用程序或服务都可能有许多依赖项，而这些依赖项本身也可能依赖于其他组件。

开源许可证

当软件中使用开源组件或开源组件的代码片段时，用于阐述最终用户义务的一组条款和条件，包括如何使用和重新分发该等组件。开源许可证基本上分为两类。

宽松型许可证(Permissive License)

宽松型许可证基本不设任何使用限制。一般来说，此类许可证的主要要求是原始代码的归属权属于其原始开发者。

著作权许可证(Copyleft license)

此类许可证通常涵盖互惠义务，规定代码的修改和扩展版本必须在与原始代码相同的条款和条件下发布，并且有改动的源代码必须按要求提供。商业实体对在其软件中使用著作权许可证的开源代码十分谨慎，因为它的使用可能会带来整个代码库的知识产权(IP)问题。

软件组成分析(SCA)

一种用于自动化开源软件管理流程中的应用程序安全工具。SCA工具可以集成在软件开发生命周期中，用于识别代码库中使用的开源代码，提供风险管理和缓解建议，并执行许可证合规验证。

软件物料清单(SBOM)

代码库中的软件组件和依赖项的全面目录清单，通常由软件组成分析工具生成。正如美国国家电信与信息管理局(NTIA)所说，“SBOM应包括一个机器可读的软件组件和依赖项清单，以提供关于这些组件及其层次关系的信息。”由于SBOM旨在跨公司和社区共享，因此，具有一致的格式(即人可读和机器可读)和一致的内容至关重要。美国政府指南中，目前将两种格式指定为已被批准的标准格式：Software Package Data Exchange (SPDX)和CycloneDX。

第14028号行政命令(EO 14028)

2021年5月，美国总统拜登发布了一项名为“改善国家网络安全状况”(Improving the Nation’s Cybersecurity)的命令，指示各联邦政府机构为与联邦政府开展业务的企业制定软件安全指南。该命令中包括一个时间表，里面列出了截至本报告撰写之际不要求强制履行合同义务的各项活动。然而，尽管不存在硬性要求，但该命令已经促使很多企业重新审查其安全实践，并严格审查其软件安全风险水平。EO 14028大力提倡使用软件物料清单（SBOM），因为这可以促进软件生产者和消费者之间软件供应链信息的交流。

Apache Log4j2漏洞（BDSA-2021-3887和CVE-2021-44228等）

开源组件ApacheLog4J2（通常称为Log4j）在Java社区中用于实现应用程序日志记录。Log4j中已经发现了多个漏洞，包括远程代码执行(RCE)、拒绝服务和LDAP漏洞。

0-day漏洞

不为软件供应商或作者（他们往往对修复感兴趣）所知晓的漏洞，或者虽已知晓却尚无补丁来修复的漏洞。

OpenSSL漏洞

2022年11月，常用的开源命令行工具OpenSSL针对两个严重漏洞（CVE-2022-3602和CVE-2022-3786）发布了官方警告。这两个漏洞的严重程度后被降为“高级”。它们是证书验证过程中存在的缓冲区溢出（Buffer overflow/Buffer overrun）漏洞。利用前一个漏洞可能导致系统崩溃，并有可能导致任意代码执行；利用后一个漏洞则可能带来内存损坏问题。

缓冲区溢出（Buffer overflow/overrun）漏洞

缓冲区是应用程序执行期间的临时内存区域。当写入缓冲区的数据量超过缓冲区的容量时，就会发生缓冲区溢出，这可能会导致系统崩溃、内存问题或其他意外行为。攻击者可以利用该漏洞实施篡改文件和访问敏感信息等行为。

[漏洞与安全性]

开源漏洞与安全性

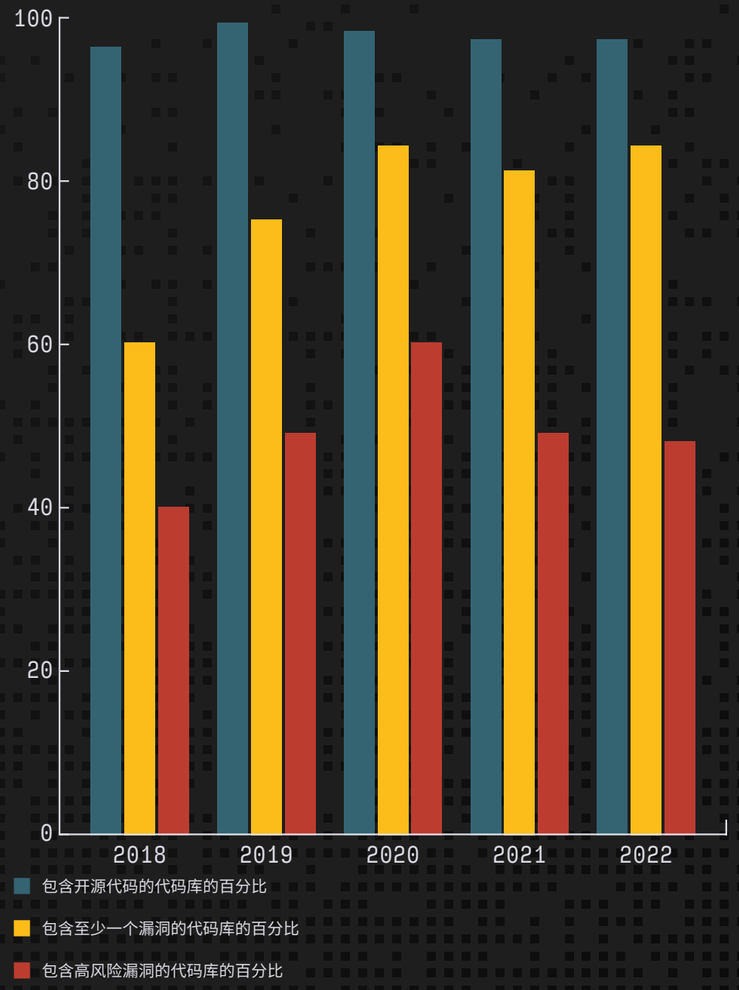
在Black Duck审计服务团队今年分析的1,703个代码库中，有96%包含开源代码。在所有的被测代码库中，76%为开源代码库，这意味着在我们研究的所有代码库中，76%为开源代码库。

今年，给定应用程序中的开源组件的平均数量为595个。如果只有区区几个组件，则手工监控其安全漏洞并对其执行安全维护活动也许是可行的，但在这种规模下，开展此类活动将变得举步维艰，几乎不可能实现，因此需要使用SCA之类的自动化解决方案。

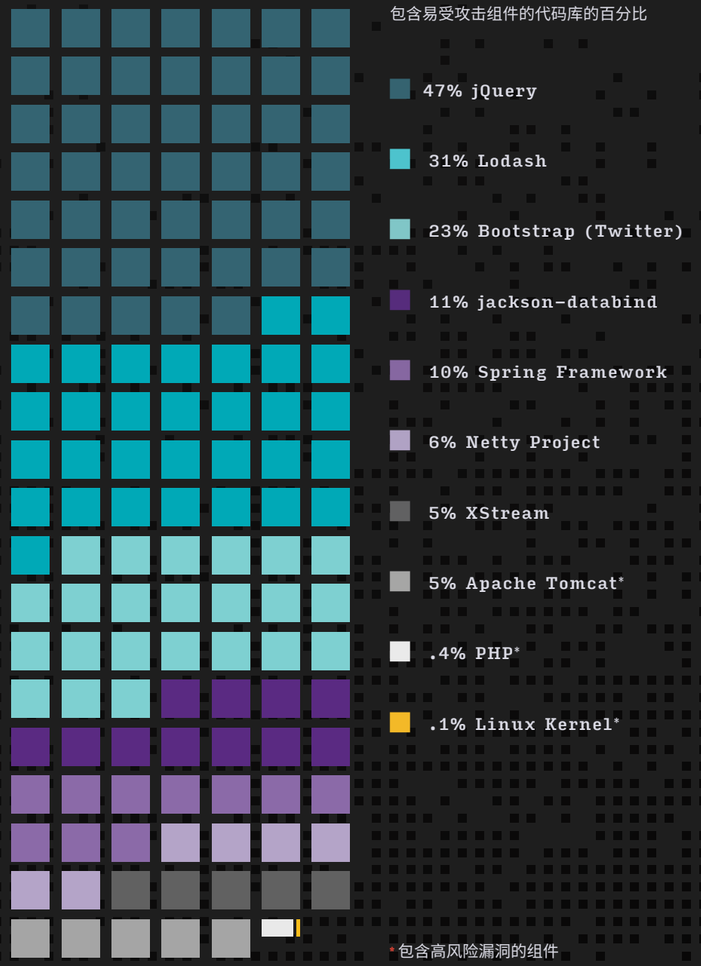
84%的代码库包含至少一个已知开源漏洞，比2022年版的OSSRA报告增加了近4%。我们检查的代码库中有48%包含高风险漏洞，仅比去年减少了2%。高风险漏洞是指已被主动利用、已有POC（证明漏洞存在）记录、或已被归类为远程代码执行的漏洞。

所有的Black Duck审计都会检查开源许可证的合规性，但客户可以自行决定放弃该审计的漏洞/运营风险评估部分。2023年，Black Duck审计服务团队共进行了1,703次审计。在这些审计中，87%的客户（1,481家）选择了安全和运营风险评估。在2023年的OSSRA报告中，“开源漏洞与安全性”以及“开源代码的维护”部分的数据基于包含风险评估的 1,481个代码库，而“许可”部分的数据则基于全部的1,703个代码库。

漏洞和高风险漏洞



包含漏洞的组件



戈尔迪之结：开源软件风险与供应链安全

新思科技与Enterprise Strategy Group共同发起的一份最新研究报告《一往无前：GitOps与安全左移》(Walking the Line: GitOps and Shift Left Security)探讨了市场关注的问题以及企业如何应对当前的安全挑战。73%的受访企业表示，近期的软件供应链攻击促使他们“大大加强了对开源软件、容器镜像和第三方软件组件的安全保护”。令人不安的是，34%的受访者还表示，他们在过去12个月内经历过“利用开源软件已知漏洞发起的攻击”。

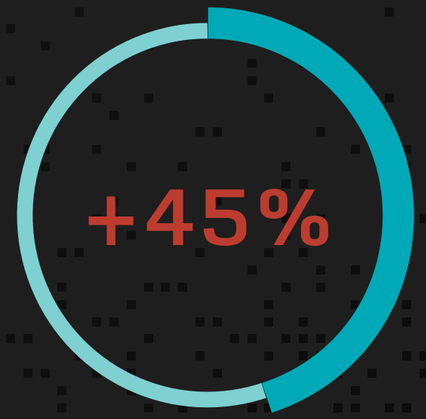
如今，任何稍微涉及软件安全的人都可能会关心软件供应链。软件供应链安全问题频繁出现在新闻报道中，并已触及各行各业的各个组织。但是，距离拜登总统签发第14028号行政命令(EO 14028)已近两年，各组织仍然未能解决供应链基础问题，包括了解其软件供应链的广度、对其所依赖的软件建立可视性、以及满足其分发和销售的软件日益增长的透明度要求。

那么，该如何应对呢?保护软件供应链的第一步是管理应用程序中的开源代码和第三方代码。

如果您不能有效地管理并确保开源软件及第三方软件的安全性，那么，您为了保护供应链而开展的所有其他工作都将徒劳，或者坦白说，根本无济于事。管理这类软件，您需要完全了解贵组织对它们的依赖情况，并能够针对这些组件引入的风险轻松收集相关信息。一旦确定了风险，您将需要借助适当的工具和实践对其进行管理、确定优先级并进行补救。

此外，将已识别的所有风险悉数告知关键利益相关者也很重要，只有这样他们才能了解这些风险，为风险管理活动和计划提供支持。再者，所有这些能力和实践均应构建到现有的开发管道中，并尽可能自动执行。

听起来很复杂?因为事实就是如此。供应链的最终产品及其用户都会受到其创建过程中涉及的每个组件、人员、活动、材料和程序的影响。只有获得对供应链及其众多相关元素的完全可视性，您才能开始保护它。而这种可视性则是从验证您是否真正安全开始的。俄罗斯的古老谚语“要信任，但要验证”用在这里再合适不过了；管理开源软件和第三方软件的前提是您已经验证了其安全性。如果不进行验证，就相当于您毫无根据地信任供应链中最薄弱的环节。



到2025年，全球将有超过45%的组织遇到软件供应链攻击问题

–Gartner

开源软件风险和供应链安全密不可分。

那么，该如何应对呢?保护软件供应链的第一步是管理应用程序中的开源代码和第三方代码。

按行业划分的漏洞情况

我们看到，包含开源代码的代码库占比逐年上升。今年，即便是占比最低的行业（制造业、工业和机器人），也有92%的代码库中包含开源代码。

然而，开源与漏洞的密切相关令我们感到不安，尤其是航空航天、汽车、运输和物流行业。我们在该领域审查的所有代码库中都包含开源代码，且开源代码占到代码总数的73%。63%的代码库中包含高风险漏洞（严重程度为7分或更高）。

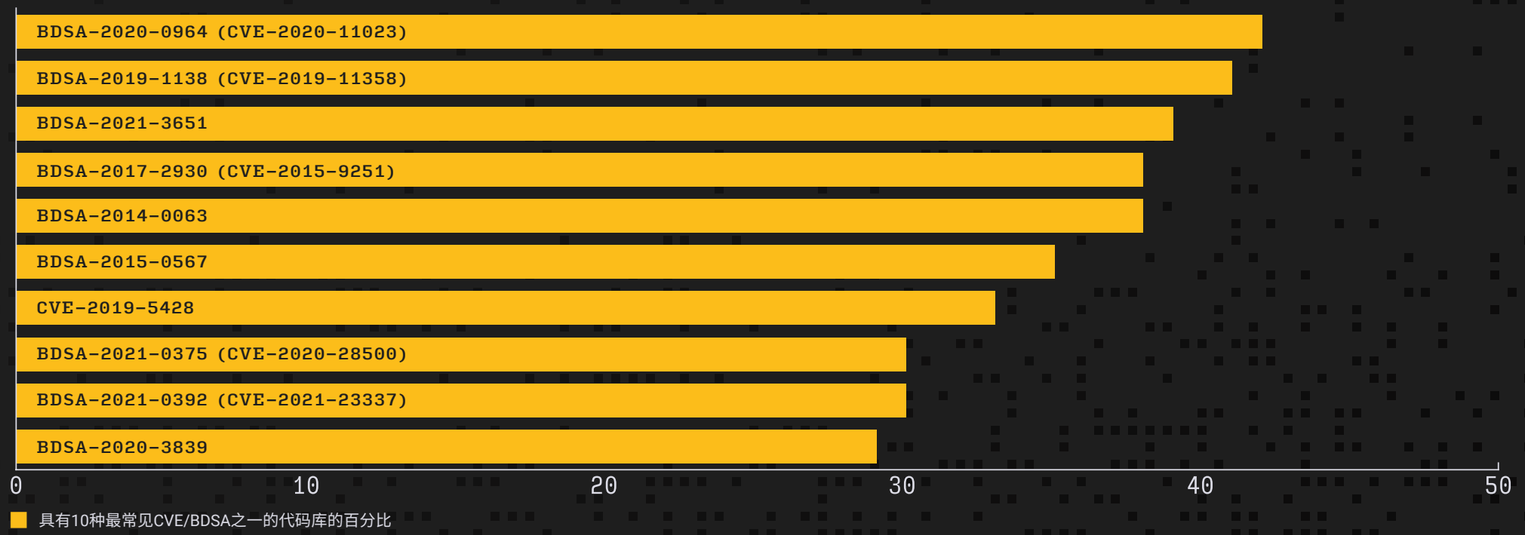
能源与清洁科技领域的情况基本相同，其中78%的代码为开源代码，69%的代码中包含高风险漏洞。该领域中包含开源代码的代码库占比为95%。

其他行业虽然程度较轻，但情况大致相同，这给安全团队敲响了警钟。我们今年审查的所有代码库中几乎都包含开源代码；开源代码几乎出现在各行各业的所有代码库中，并且包含企业未能修补的大量已知漏洞，这些漏洞很容易被利用，着实令人感到不安。请切记，尽管开源本身不会带来任何固有的风险，但如果管理不善，就会带来风险。

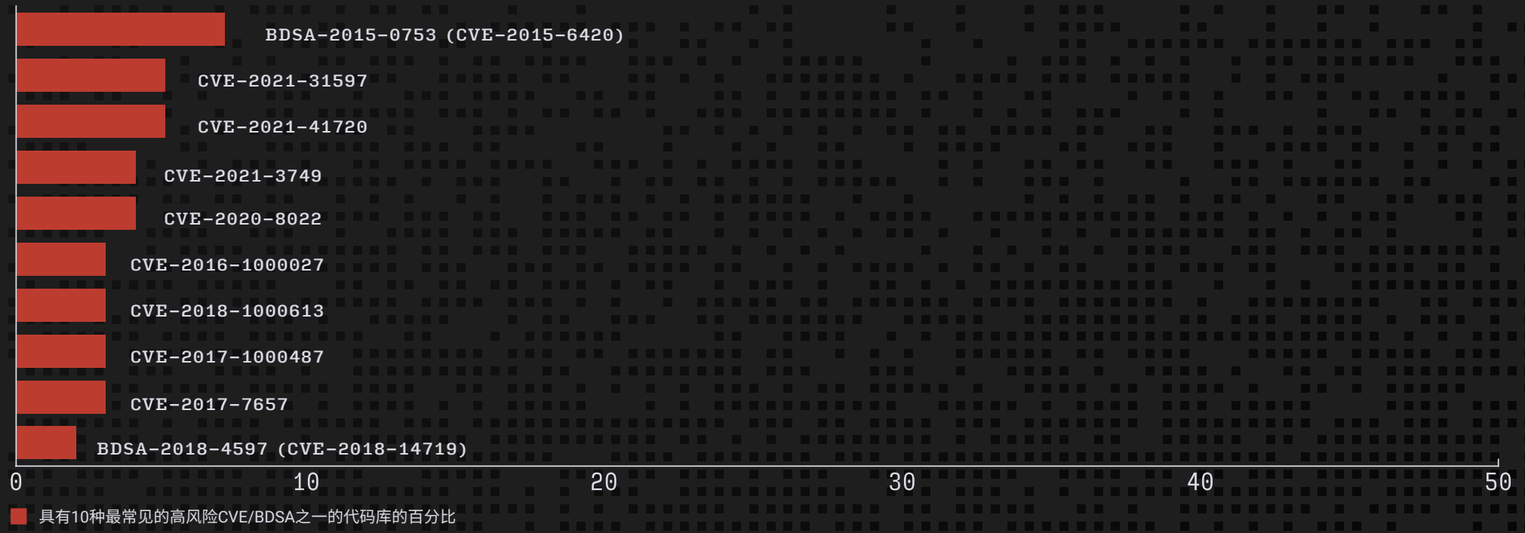
漏洞严重性评分

新思科技漏洞严重性评分系统在确定分数时收集大量的影响评分的变量信息。这些分数是我们Black Duck Security Advisory(BDSA)的一部分。BDSA利用FIRST.org规定的CVSS评分系统，以确保漏洞严重性分数与CVSS相一致，但新思科技的漏洞严重性分数是由CyRC给出的，而不是简单地模仿NVD发布的评分。在评分时，BDSA会考虑可利用性等诸多因素，这有助于确保最精确的CVSS分数。此外，不同于NVD等评分系统，BDSA在评分时还会考虑时间指标。我们的目标是尽可能提供最精细、最准确的评分，以帮助客户合理分配活动的优先级。

CVEs/BDSAs



高风险CVE/BDSA



五年回顾

今年，为了发现显著趋势，我们对OSSRA报告数据进行了五年回顾。我们的调查结果如下。

调查结果1. 开源代码的采用因行业而异

我们每年都会报告开放源代码的持续增长和采用情况，但通过五年回顾，我们发现不同的垂直领域在开源代码的采用方面存在较大差异。正如我们之前提到的那样，今年，所有的垂直领域都有92%以上的代码库中包含开源代码。然而，当我们查看过去五年中开源代码在代码库中的总占比时，发现了巨大的不同。

2018到2022年间，在教育科技领域的审计代码库中，开源代码占比增长了163%；航空航天、汽车、运输和物流领域增长了97%；制造业和机器人领域增长了74%。我们认为，开源代码在教育科技领域的这一爆炸性增长是新冠疫情导致的；随着教育被推到线上，软件成为教育的关键基础，因此这种增长是合理的。由于开源是免费的（尽管它必须得到妥善管理），因此，它对希望在短时间内做出重大改进但预算紧张的行业特别有吸引力。许多教育科技系统都是由志愿者自行开发和维护的，因此，开源很可能成为新兴教育科技技术的基础。

通过五年回顾，我们发现开源代码在航空航天、汽车、运输和物流领域的采用速度相对较慢（但去年突然增长了22%）可能是这些垂直行业受到严格监管的缘故。在过去，由于这些行业中的组织缺乏足够的资源和能力来有效保护开放源代码，因此会更加刻意地避免使用开源代码。

按行业划分的开源和高风险漏洞

新思科技政府与关键基础设施项目前主管Joe Jarzombek对这些数字发表了另一种看法。Jarzombek指出："软件质量不佳所带来的技术债务阻碍了新功能的交付。与其他工业领域和政府部门一样，国防工业基地将大部分时间和精力花费在纠正技术债务上，而不是开展主动的、创造性的或预防性的工作上。"他接着指出，近期的网络安全成熟度模型认证"有助于"许多行业减轻与数据保护相关的安全顾虑"，从而使这些行业提高了创新能力和敏捷性。

也就是说，开源代码的采用速度缓慢——及其所提供的创新、速度和灵活性——导致过时的做法得以继续。但技术债务和繁琐的法规正在慢慢让位，推动开源代码得到更广泛地使用。

调查结果2. 企业未修复高风险漏洞

当我们按行业比较高风险漏洞的发生率时，我们看到所有行业都存在不同程度的高风险漏洞。自2018年以来，高风险漏洞在营销科技领域至少增加了42%；在零售和电子商务领域更是猛增557%，着实令人担忧。

让我们再次回到航空航天、汽车、运输和物流领域（高风险漏洞在这个垂直领域大幅增加了232%），该领域是否根本不需要降低风险呢?漏洞修复的一个关键驱动因素是漏洞利用的可能性和潜在影响。这些行业使用的大部分软件和固件都在封闭系统中运行，因此能够降低漏洞被利用的可能性，从而修补漏洞的紧迫性不高。此外，这些行业的漏洞还可以通过无需更新组件的其他方式进行缓解。

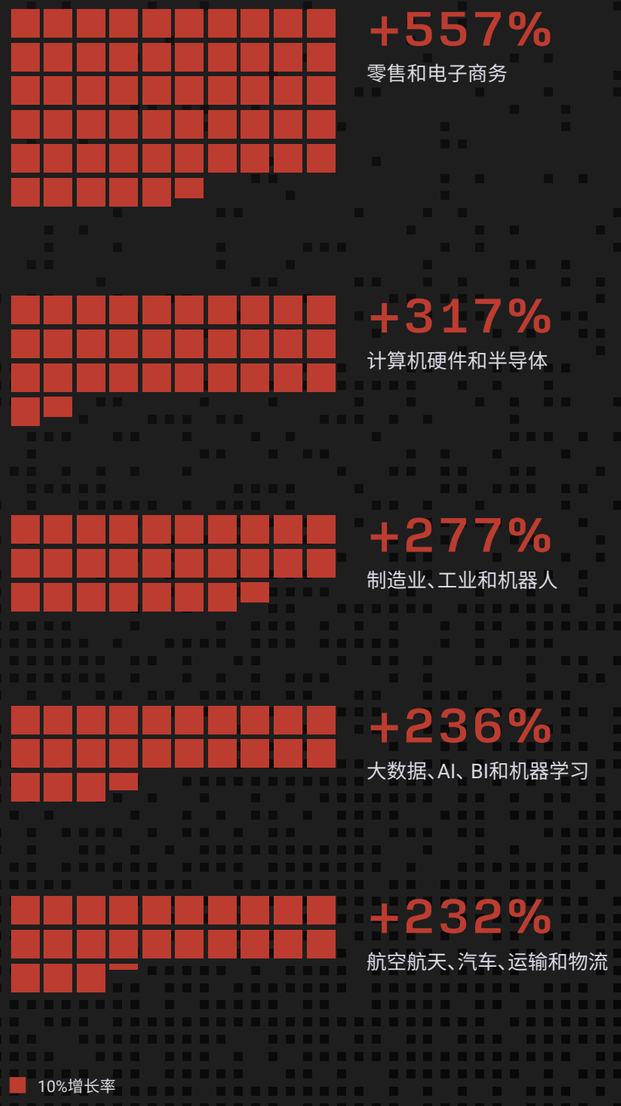
如果我们考虑到软件在该垂直领域的部署、分发和操作方式，则能够为存在如此大量的高风险漏洞找到另一种可能的解释。嵌入式软件和固件通常用于无法访问网络的硬件上，这意味着更新包无法在网络中轻松自动地发布，就像SaaS应用一样。当安装补丁意味着必须下载并安装更新版本，或将U盘插入设备，则补丁的安装频率势必有所减少，从而导致未修补的漏洞有所增加。

此外，这个领域的软件采用了不同的标准与实践，这与独立软件供应商和公司构建企业级软件及SaaS应用的标准不同。这种区别或许是该垂直领域无法解决高风险开源软件漏洞问题的原因之一。

在物联网(IoT)领域，我们看到了不同的情况。自2020年以来，我们扫描的所有代码库中都包含开源代码，且每个代码库中的开源代码总量也在增加；开源代码占比自2018年以来增长了35%，今年的总占比高达89%。IoT是开源优势的绝佳体现；IoT公司面临着快速开发新软件的巨大压力，例如，Ring、Amazon和Nest迫切需要为其提供的智能设备开发新软件。在竞争白热化的行业，开源可以让企业快速行动起来。如果没有开源，他们将不可能跟上飞速发展的步伐。但问题在于开源代码中存在安全漏洞。

自2018年以来，IoT领域的高风险漏洞增加了130%，今年，53%的审计应用中包含高风险漏洞，是我们调查中高风险漏洞占比较高的领域之一。当考虑到IoT设备的功能和作用时，这一点尤其令人担忧；我们将许多日常生活事务交给这些设备，并认为它们天生安全。IoT设备可以根据我们设置好的时间表自动开灯，因此这些设备存储着我们何时在家以及何时外出的数据。可以拍摄家里图像的摄像头、大门上的智能锁以及婴儿监控器等等，都存在安全隐患。

高风险漏洞五年增幅最大的行业



[许可]

开源许可

Black Duck审计服务团队发现，2022年审计的代码库中有54%包含存在许可证冲突的开源代码，比上一年略微增加了2%，但相对2020年的65%仍然大幅减少了17%。

Creative Commons ShareAlike 3.0 (CC BY-SA 3.0)许可证是今年许可冲突的最主要的原因；22%的审计代码库存在与该许可证相关的某种形式的冲突。CC BY-SA 3.0许可证冲突数据揭示出开源许可证方面经常被忽视的一个问题。商业和开源开发人员经常将代码片段、函数、方法和操作代码片段引入到他们的软件中，通常称为“依赖项”（因为整个软件都依赖于该代码）。因此，开源项目通常涉及一些子组件，这些子组件与整个项目在不同的许可证下授予许可，其条款和条件也不同于主许可证中的条款和条件，从而引发冲突。

在Black Duck审计服务团队发现的许可证冲突中，85%涉及来自Stack Overflow的内容 — Stack Overflow是用于查找和共享技术知识的在线问答平台。考虑到Stack Overflow广受欢迎，并且CC BY-SA 3.0许可证在分布式和SaaS部署模式中都是主要的冲突来源，因此，Stack Overflow成为最大的冲突来源也就不足为奇了。

我们看到各行各业的开源许可证冲突均大幅减少。去年，计算机硬件和半导体行业有93%的代码库存在开源许可证冲突。这一数字在今年降至75%。总体而言，我们看到各行各业均取得了类似的改进；今年，开源许可证冲突占比最高的垂直领域是物联网领域，为78%。开源许可证冲突的这一总体改进可能是经济不确定性和人们对供应链风险的普遍焦虑所致。

按行业划分的开源和许可证冲突

了解许可证风险

在美国和许多其他地方，创造性的工作（包括软件）默认受专有版权的保护。未经创作者/作者以授权许可证的形式明确允许，任何人对该等软件的使用、复制、分发或修改均不合法。即使最友好的开源许可证也会规定用户在使用软件时需要承担的义务。

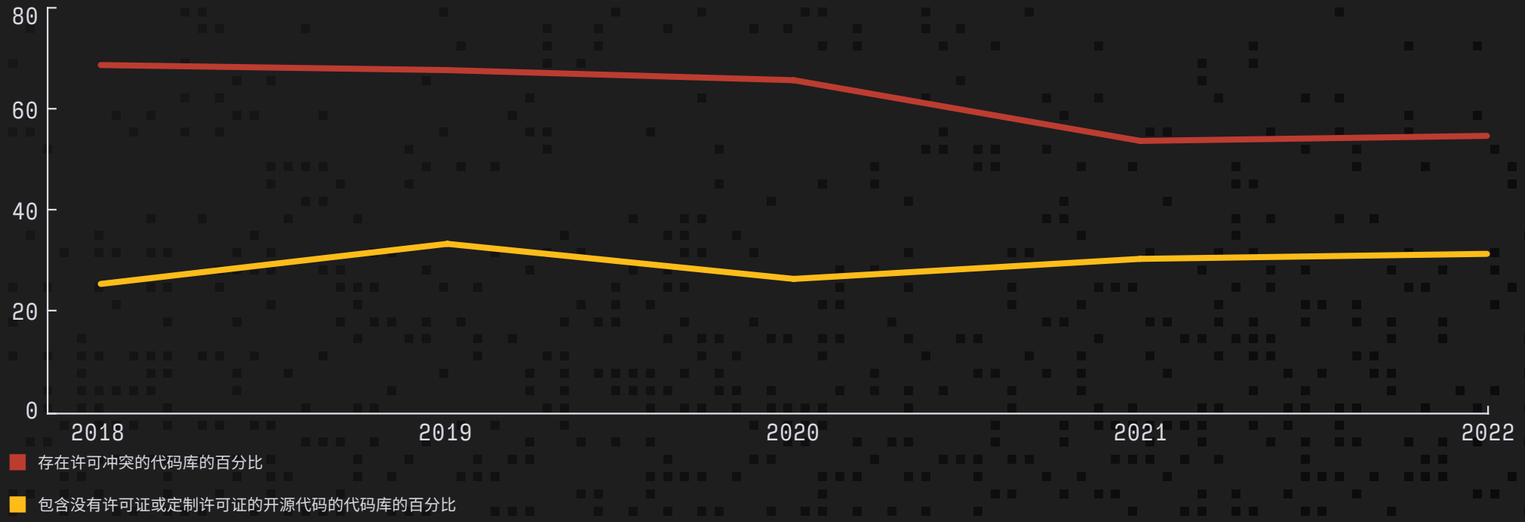
当代码库中包含的开源代码许可证与该代码库的总体许可证存在冲突时，就可能存在潜在的许可证风险。GNU通用公共许可证(GPL)是应用于开源项目的最常见的著作权许可证。如果商用闭源软件中包含在GPL下授予许可的代码，就会产生冲突。

在2022年审计的代码库中，1/3的代码库使用了没有可识别许可证或具有定制许可证的代码，比去年增长了55%。这一结果可能是两个原因导致的。首先，这可能是因为开发人员手动将代码片段或部分组件添加到代码库中。这样做的时候，开发人员通常无法将代码片段的相关许可证一并添加。其次，这可能仅仅是因为项目维护人员制订的许可证和条款偏离了已知标准许可证。未能充分理解许可证的含义虽然看似没有问题，但可能会有风险，应尽量避免。

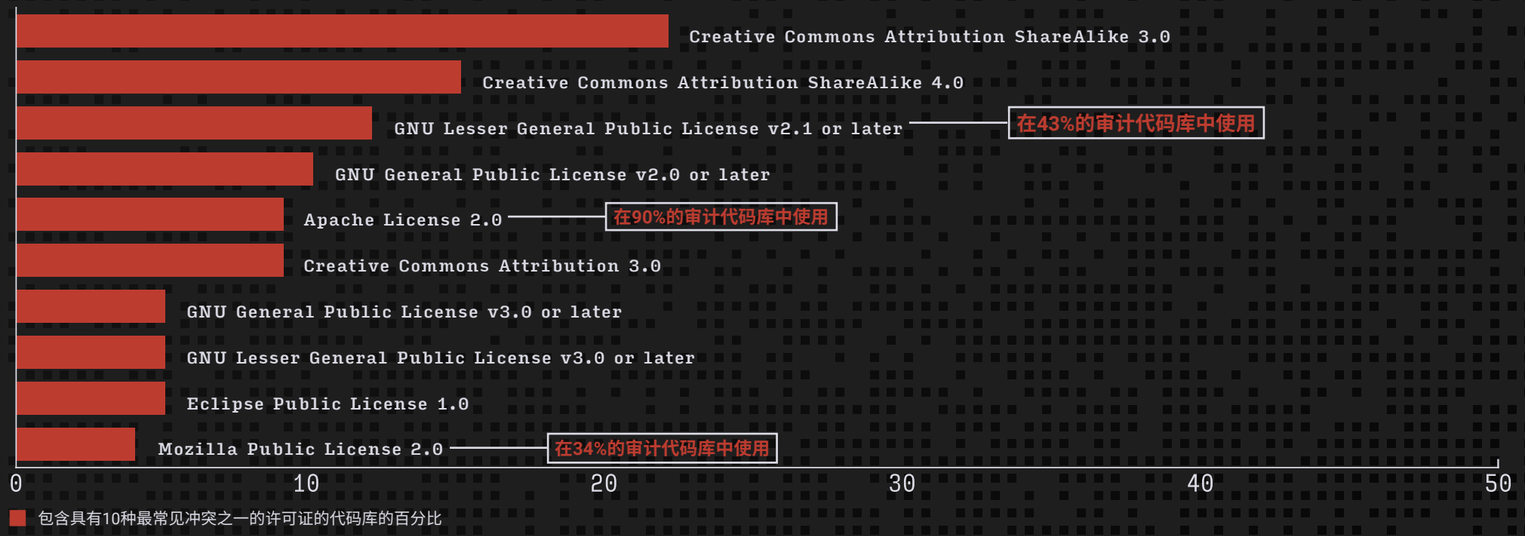
在Github上，可以找到大量无许可证的项目代码，开发人员已经公开了代码，但是无论是在代码中、.txt文件中、还是与项目相关的元数据中，都没有声明许可。因此，GitHub中可能存在未经许可的代码。或者，开发人员可能会从博客或网站复制代码，理所当然地认为这些公开提供的源代码是可以随意使用的。这种想法看似合理，但版权法却不这么认为。

此外，标准开源许可证的变体或定制版本许可证可能会对被许可方提出不必要的要求，并且要求对可能的知识产权问题和其他问题进行法律评估。例如，JSON许可证便是定制许可证的典型示例。JSON许可证基于宽松型MIT许可证，只不过添加了“该款软件严禁用于恶意用途，仅限用于善意用途”的限制。该声明的含糊不清导致“善意用途”和“恶意用途”成为很难界定的东西，许多律师都建议避免使用采用该等许可模式的软件，尤其是在并购的情况下。

许可问题



许可证冲突



[开源代码的维护]

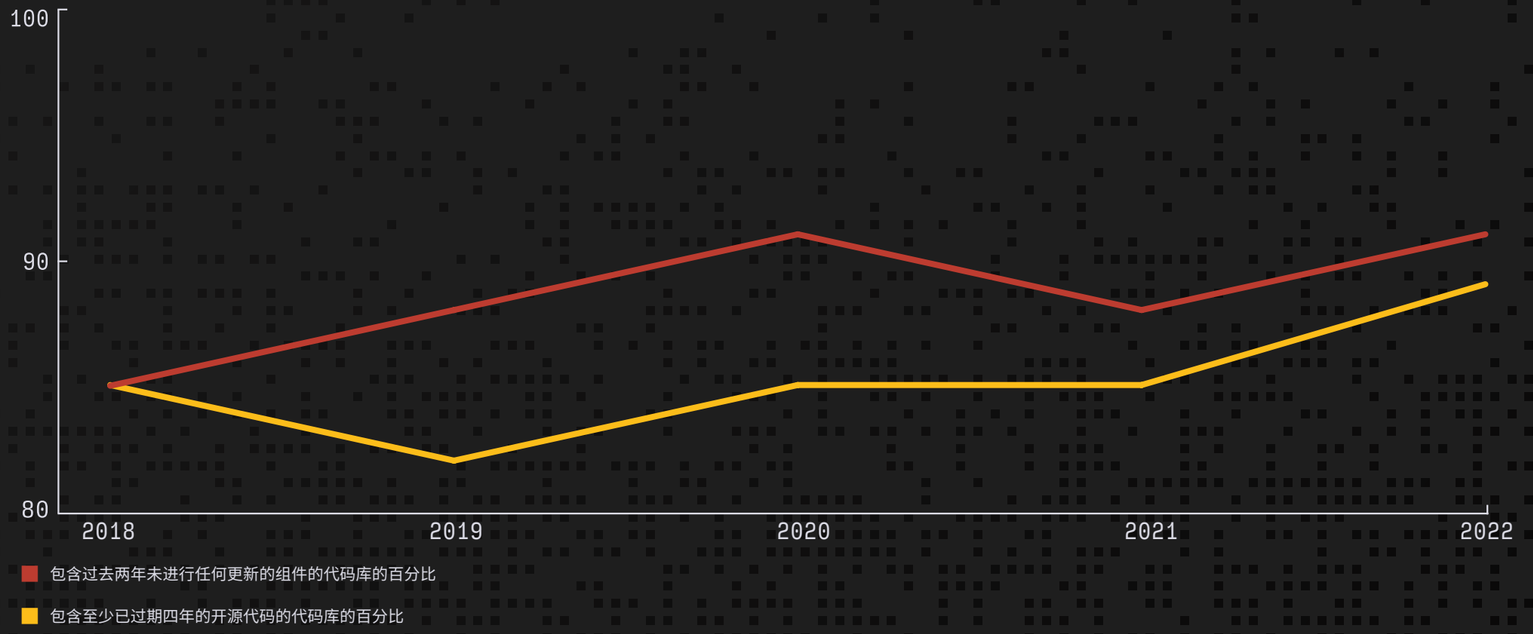
由开源代码的开发者进行维护

按理说，开发人员应选择由强大社区提供支持的开源组件。以Linux为例，来自数百个组织的数千名开发人员每天都在改进Linux。然而，在Black Duck审计服务团队审查的1,481个含风险评估的代码库中 (见第7页)，91%都包含过去两年内未进行任何更新的开源代码 — 在过去24个月中未开展任何功能升级、代码改进和安全问题修复活动。这可能意味着相关方不再对该项目进行维护，尤其是在小型项目的情况下。

顾名思义，开源项目是由不确定数量的贡献者和维护者共同努力的结果。这种结构使开源成为一种协作项目，但挑战在于开展维护活动的贡献者缺乏激励。诸如Kubernetes之类的重要项目通常可以获得良好的支持，但也有很多项目仅由少数几个人进行维护。

近期，有关开源项目的一篇评论性文章指出，“对于软件所严重依赖的重要开源项目，大多数维护人员几乎都没有得到任何认可。”尽管Microsoft、RedHat和Google等部分企业已经制订了适当的奖励计划来激励开源项目的维护和参与，但绝大多数企业都没有相应举措。其结果是为我们每天依赖的软件提供动力的基础性开源项目无人维护，有时一放就是数年。

代码库的可持续性



例如，Twitter因放弃开源项目而在最近几个月引起了巨大轰动。据Twitter前开源负责人Will Norris称，Twitter已经“放弃了开源开发”。他表示：“在Twitter从事开源工作的大多数关键人物都已经离开了。和我一起开发开源软件的所有工程师都不在了。随着新的管理层到位，新的业务优先事项出台，曾经备受重视的开源项目开始降级，取而代之的是管理层新确定的重点项目。”

从诸如Twitter的情况来看，因缺乏维护而带来的后果是开源项目面临的主要挑战。如果项目没人维护，轻则带来技术债务，重则导致安全风险。

已知风险之外的风险

除了被放弃的项目之外，还发现开源项目被恶意破坏的情况。对于供应链安全性，我们经常听说某某组织发现并缓解了已知漏洞，但在供应链中引起关注的新形式的开源软件风险又是什么呢？恶意开源软件包、“抗议软件”(protestware)、依赖混淆和误植域名(typo-squatting)都对开源安全构成了令人担忧的新威胁。

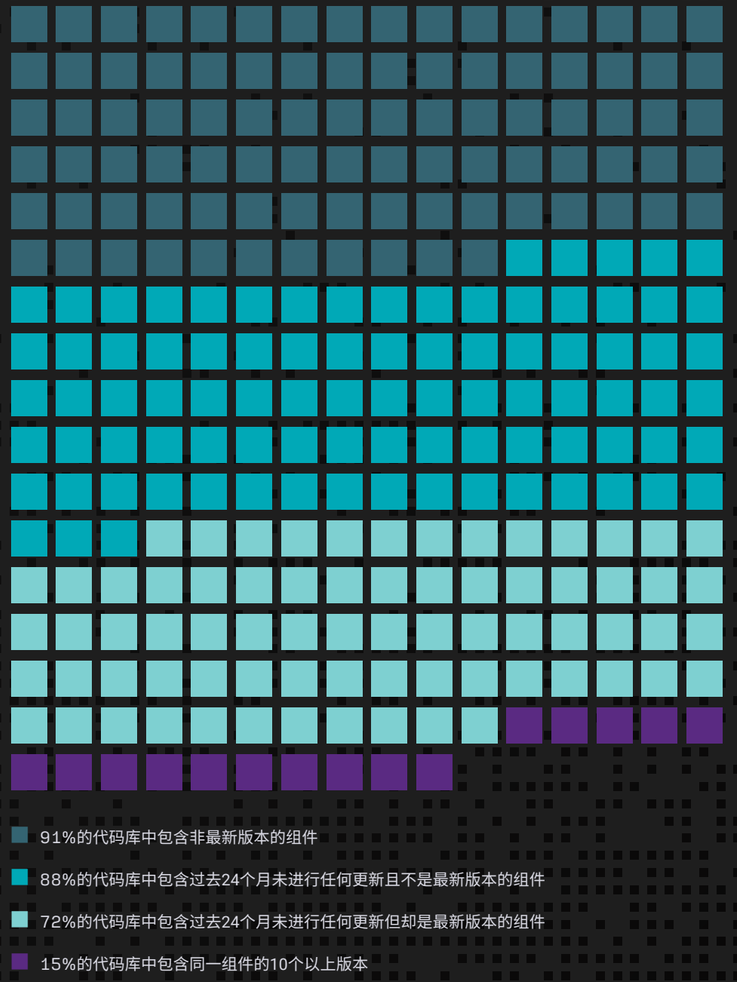
回顾过去五年的泄密事件和重大漏洞，我们发现信任是通病。企业和最终用户必须对其使用的软件以及该类软件的开发者和提供者给予一定程度的信任。简单说，企业相信其供应链中的每个节点都有相应的安全和质量保障措施，这个假设无疑是危险的。

新思科技与Consortium for Information and Software Quality联合发起的名为《美国软件质量差的代价》(The Cost of Poor Software Quality in The US)的最新报告指出，勒索软件、加密劫持、IoT和OT攻击以及供应链攻击是2022年最热门的网络犯罪趋势。这里的病根仍然是信任问题：我们相信开源代码的维护人员和开发人员能够发现并修复漏洞。我们相信开源贡献者和我们拥有相同的想法和动机。遗憾的是，情况并非总是如此。请切记，开源项目会带来已知漏洞之外的运营风险。因此，降低运营风险还需要预测未来威胁并采取措施加以防范。

去年夏天，在npm（JavaScript运行时环境Node.js的默认包管理器）中发现了几个能够从嵌入在移动应用和网站的表单中获取敏感数据的恶意软件包。这些被下载了数千次的软件包对其他受到信赖的常见软件包进行域名误植攻击。这些软件包的任何版本都应被认为是易受攻击的和恶意的，如Icon-package、Ionicio和Ajax-libs等。

当您读到这个例子时，您是否知道自己使用了npm？贵组织中的安全责任人是否对其进行了适当的检查并能够确保安全？如果不是，您可能有点过于轻信了。清楚地掌握项目的声誉以及项目维护人员，这一点至关重要，因为他们开展这些活动的动机可能与您不同。恶意的维护人员可能会利用您对供应链的信任，在未经尽职调查的情况下将恶意代码植入您的应用程序。

版本控制问题



由开源代码的使用者进行维护

在Black Duck审计服务团队审计的1,481个包含风险评估的代码库中（见第7页），91%包含过时版本的开源组件。也就是说，尚未安装最新的升级包或补丁。

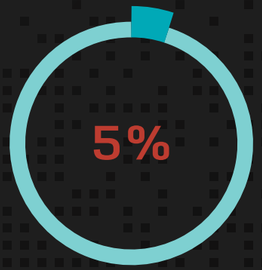
他们不对软件进行及时更新是情有可原的。DevSecOps团队可能认为引入新版本会带来意外后果，引入新版本产生的风险会超过由此带来的任何好处。嵌入式软件仅存在从外部来源引入的漏洞风险，因此风险最低。他们不对软件进行及时更新也可能是因为时间/资源问题。由于构建和测试新代码已经耗掉了许多团队的全部精力，因此，除了最关键的问题外，对现有软件进行更新可能会沦为较低优先级的事项。

对于包含过时版本开源组件的91%的代码库而言，有很大一部分原因是因为DevSecOps团队并不知道该开源组件的新版本已经可用，甚至根本不知道该组件的存在。企业最熟悉的通常是商业软件，因为这些软件的补丁和更新是自动推送的，所以无需他们时刻关注组件更新情况。开源软件的运作方式则完全不同。使用开源软件意味着用户有责任了解和控制组件的安全性和稳定性，并时刻关注组件的新版本和补丁。

修补失败

一年后，Log4Shell成为长尾漏洞的典型案例。尽管它受到了媒体关注，企业可以采取多种措施来确认其在代码库中的存在并进行补救，但在今年的审计中，我们发现其仍然存在。我们在5%的全部代码库和11%的Java代码库中发现了易受攻击的Log4J版本。

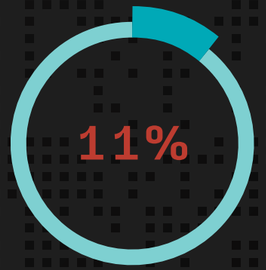
易受攻击的Log4J版本之所以仍然存在，最有可能的原因是企业根本不知道它们的存在。Log4J是基础组件，导致其在许多情况下不会被安全团队所看到。基础组件的修补工作也比表层的客户端组件更复杂，因此，也许安全团队知道它们的存在，只是还没有进行修复。



5%的被审计代码库中包含易受攻击的Log4J版本

我们预计在未来几年内，企业的基础组件中会出现类似Log4J的漏洞。其原因首先是企业对开源的固有信任。他们认为开源软件会像商业软件一样接受相同的安全检查和保护。其次是因为企业无法识别其应用程序中的开源组件。这究竟是因为缺乏安全措施或安全能力，还是两者兼而有之，目前尚不清楚。

避免开源、专有和商业软件带来业务风险的第一步是对企业使用的所有软件进行全面的盘点，无论其来自何处或是如何获得的。拥有了这个完整的清单 — 值得信赖的SBOM — 安全团队才能规划出前进的道路，并制订计划来应对Log4Shell等新披露的安全漏洞带来的风险。



11%的Java代码库中包含易受攻击的Log4J版本

[结语]

“要信任，但要验证”

“要信任，但要验证”源于战争时期，在当今供应链攻击大量涌入的背景下，这句话听起来很奇怪。信任您使用和开发的软件本身不是问题，无法验证其是否接受了必要的安全分析才是问题。

就连Gartner也在强调对供应链安全性进行充分验证的紧迫性。据 Gartner 预测，“到2025年，全球45%的企业将经历软件供应链攻击，比2021增加三倍。”在数字战争的背景下，企业如何才能为应对不可预测的情况做好充分准备呢?

信任的问题

如上文所述，盲目地信任应用程序（或包含应用程序的代码）的安全性可能会带来毁灭性后果（即安全漏洞）。应摒弃这种内在的信任，像攻击者一样思考问题。攻击者的本质是机会主义。其目标永远是最平坦的攻击路径和最大的收益。如果不能保证软件安全，您如何能够确保不会成为攻击目标呢?

预测每一条攻击路径是不可能的，但围绕着业务风险培养组织意识可以帮助企业减少威胁。今天，所有的企业都依赖软件，这意味着他们都在从事软件业务。一旦开始从这个角度考虑组织安全性，便能意识到盲目地相信软件安全性是危险的。用户有责任保护自己的软件，无论是继承的还是内部开发的。

通过SBOM进行验证

在对抗软件供应链攻击时，软件物料清单(SBOM)应该是首选武器。SBOM的概念源于制造业，典型的SBOM是一份详细列出产品中所含全部项目的清单。这样，一旦发现有缺陷的零部件，制造商便可以知道哪些产品受到了影响，并安排对其进行维修或更换。同理，维护准确的、最新的、列出开源组件的SBOM对于确保代码的高质量、合规性和安全性是必要的。与制造业一样，开源组件的SBOM允许用户快速查明风险组件，并合理确定补救措施的优先级。全面的SBOM列出了应用程序中的所有开源组件，以及这些组件的许可证、版本和补丁状态，是对供应链攻击的完美防御。

SBOM旨在帮助管理开源和第三方代码的使用，提供对应用程序“成分”的可视性，并标准化信息通信方式。除作为文档或记录的基本功能外，我们还建议用户将SBOM视为一个管理系统或者工具、实践和过程。。用户应该具备溯源意识来识别开源组件，然后将这些组件映射到漏洞数据。有效的开源管理有助于围绕战略和安全计划的改进展开对话，以实现成功的供应链风险管理。

2022年，96%的商业代码中包含开源组件，因此，了解应用程序中使用的组件应该是任何现代DevSecOps项目的基本要求。随着对业务风险和整体安全性的深入了解，您将不再相信自己是安全的，而是要亲自验证一下。

相关读物

- 白皮书：一往无前：GitOps与安全左移 Walking the Line: GitOps and Shift Left Security
- 文章：随着杠杆收购的萎缩，年底前的并购活动似乎萎靡不振（M&A Activity Looks Anemic Heading to Year-End as LBOs Shrivel）
- 文章：全球并购市场2022年上半年放缓，但显示出强劲迹象（lobal M&A market slows in 2022 first half—but shows signs of strength）
- 维基百科词条：要信任，但要验证（Trust, but verify）
- 新闻稿：网络安全成熟度模型认证(CMMC)计划的战略方向（Strategic Direction for Cybersecurity Maturity Model Certification (CMMC) Program）
- 网页：改善国家网络安全：NIST在2021年5月行政命令下的责任（Improving the Nation’s Cybersecurity: NIST’s Responsibilities Under the May 2021 Executive Order）
- 采访：Joe Jarzombek一对一采访（1:1 with Joe Jarzombek）
- 文章：Twitter放弃了开源开发（Twitter turns its back on open-source development）

新思科技与众不同

新思科技提供的集成解决方案，可以改变您构建和交付软件的方式，在应对业务风险的同时加速创新。与新思科技同行，您的开发人员可以在编写代码的时候快速兼顾安全。您的开发和DevSecOps团队可以在不影响速度的情况下在开发管道中自动进行安全测试。您的安全团队可以主动管理风险并将补救工作聚焦在对贵组织最重要的事情上。我们无与伦比的专业知识可以帮助您规划和执行所需的安全计划。只有新思科技能够满足您构建可信软件的一切需求。

SYNOPSYS®