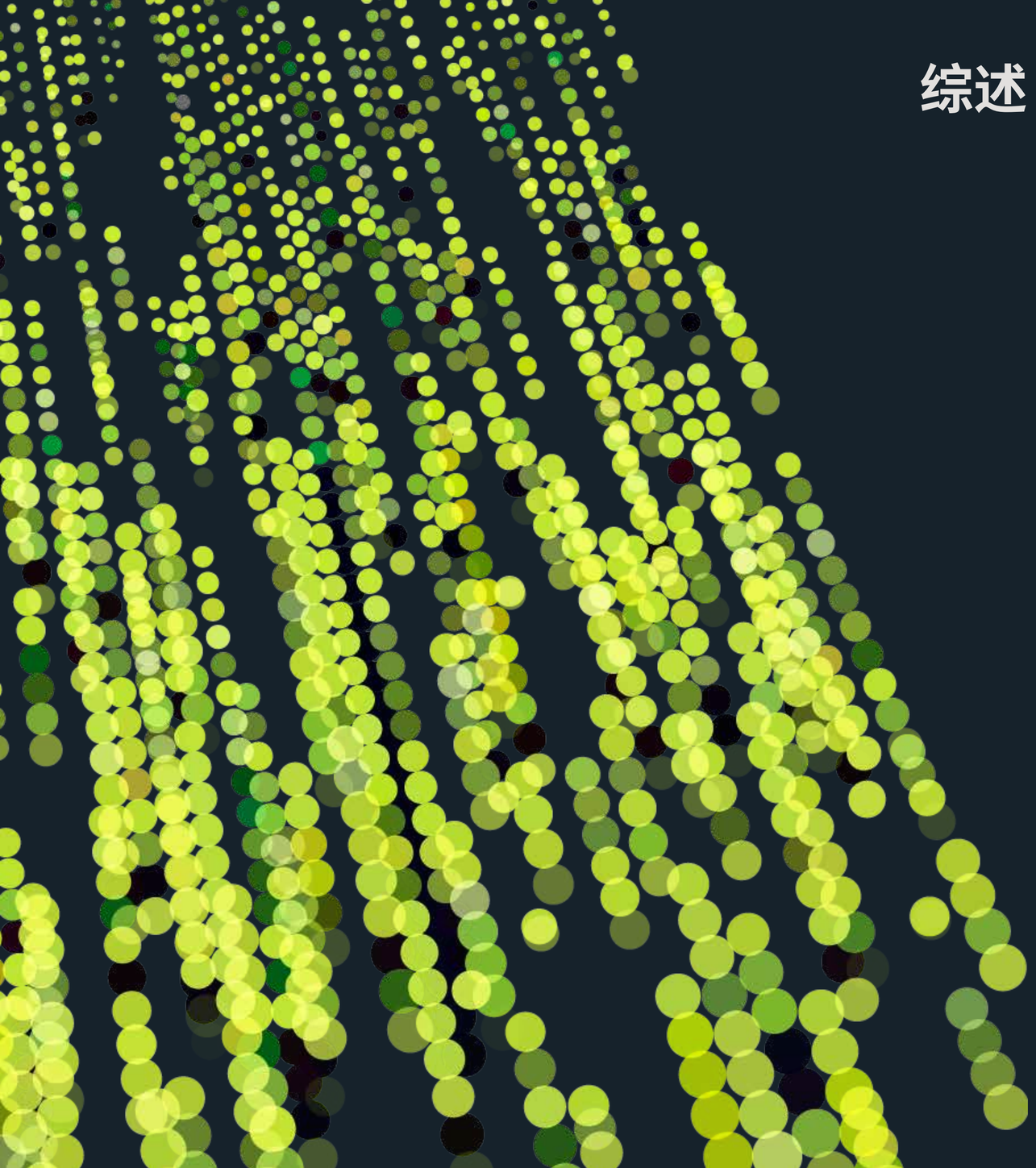
SYNOPSYS®

2024年开源安全和风险分析报告

您的开源供应链安全指南

目录



综述

本报告提供了一些建议，旨在帮助开源软件创建者和使用者负责任地管理软件，特别是在保障软件供应链安全的背景下。无论您是软件的使用者还是提供者，您都是软件供应链的一部分，需要保护您所使用的应用免受上游和下游风险的影响。在接下来的几页中，我们将探讨：

- 对开源安全的持续关注
- 为何说开发者需要改进以保持开源组件的持续更新
- 软件供应链管理需要软件物料清单 (SBOM)
- 如何防范AI编码工具带来的安全和知识产权合规风险

近十年来，开源安全和风险分析 (OSSRA) 报告的主题一直是“您是否知道贵组织的代码构成情况?”2024年，这个问题比以往任何时候都更加重要。如今，随着开源的广泛使用以及AI生成代码的日益增多，使用第三方代码构建的应用越来越多。

如果无法全面了解代码的构成情况，无论是您自己，还是您的供应商和最终用户，都将无法确定软件可能包含的风险。保障软件供应链的安全首先要知道代码中包含哪些开源组件，并识别它们各自的许可证、代码质量和潜在漏洞。

关于OSSRA 2024

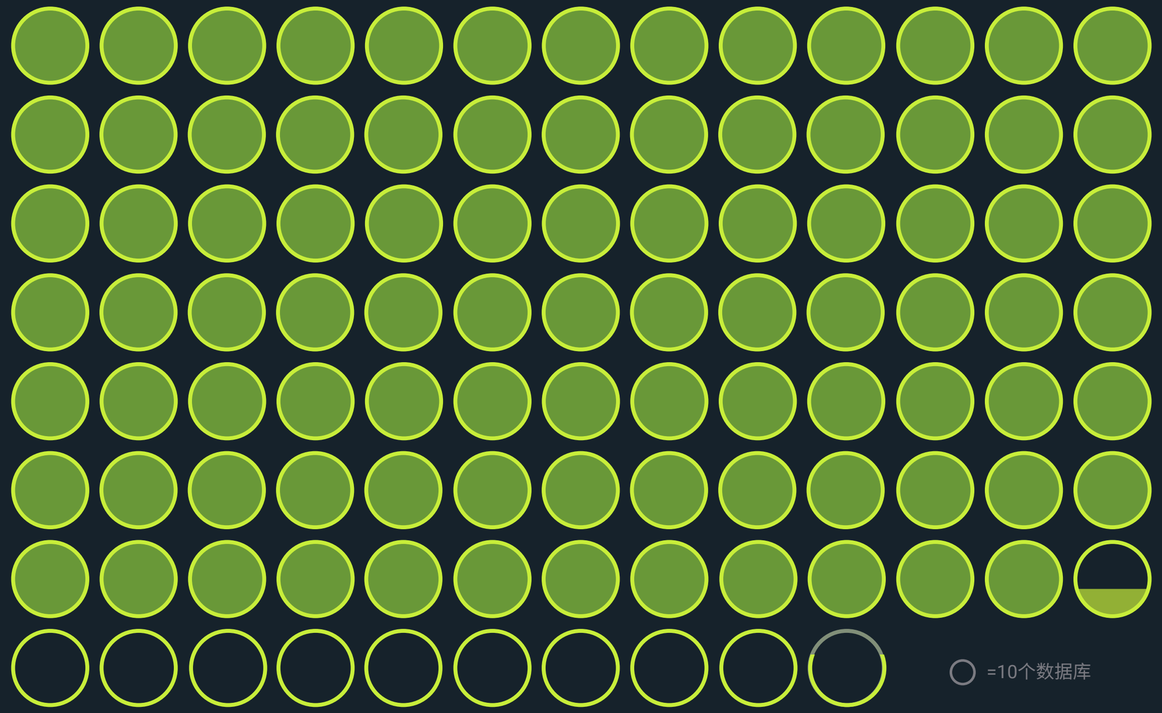
欢迎阅读2024年第9版开源安全和风险分析(OSSRA)报告。今年的OSSRA提供了新思网络安全研究中心（CyRC）对商业软件中的开源安全性、合规性、许可和代码质量风险当前状态的年度深入研究。我们分享这些调查研究结果，也是为了帮助安全、法务、风险和开发团队更好地了解安全和许可证风险状况。

本报告使用的数据来自新思科技Black Duck®审计服务团队在2023年间对来自17个行业的1,067个商业代码库的匿名调查结果。20多年来，审计服务团队一直在帮助世界各地的安全、开发和法务团队加强其项目的安全性和许可证合规。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。

该审计还提供全面、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许可证合规与安全风险。该知识库由CyRC创建、管理并积累多年，存储了来自3.1万+开源代码仓库的780万+开源组件。

本OSSRA报告强调了开源代码在软件领域的广泛使用，以及管理不善可能带来的风险。开源代码是当今各种企业和个人应用程序的基石。有效地识别、跟踪和管理开放源代码，对于成功实施软件安全计划非常重要，也是提高软件供应链安全水平的关键因素。

概述



2023年审查了1,067个代码库

936个代码库经过风险评估

96%

96%的被审代码库中包含开源代码

53%

53%的被审代码库中存在许可证冲突

77%

77%的开源代码存在于被审的代码库中

31%

31%的被审代码库中包含没有许可证或使用定制许可证的开源代码

10年

经过风险评估的代码库中，14%包含10年以上的漏洞

2.8年

经过风险评估的代码库中，漏洞平均年龄为2.8年

24月

经过风险评估的代码库中，49%包含24个月内未更新的组件

12月

经过风险评估的代码库中，1%包含至少12个月内未被更新/修补的组件

84%

经过风险评估的代码库中，84%包含漏洞

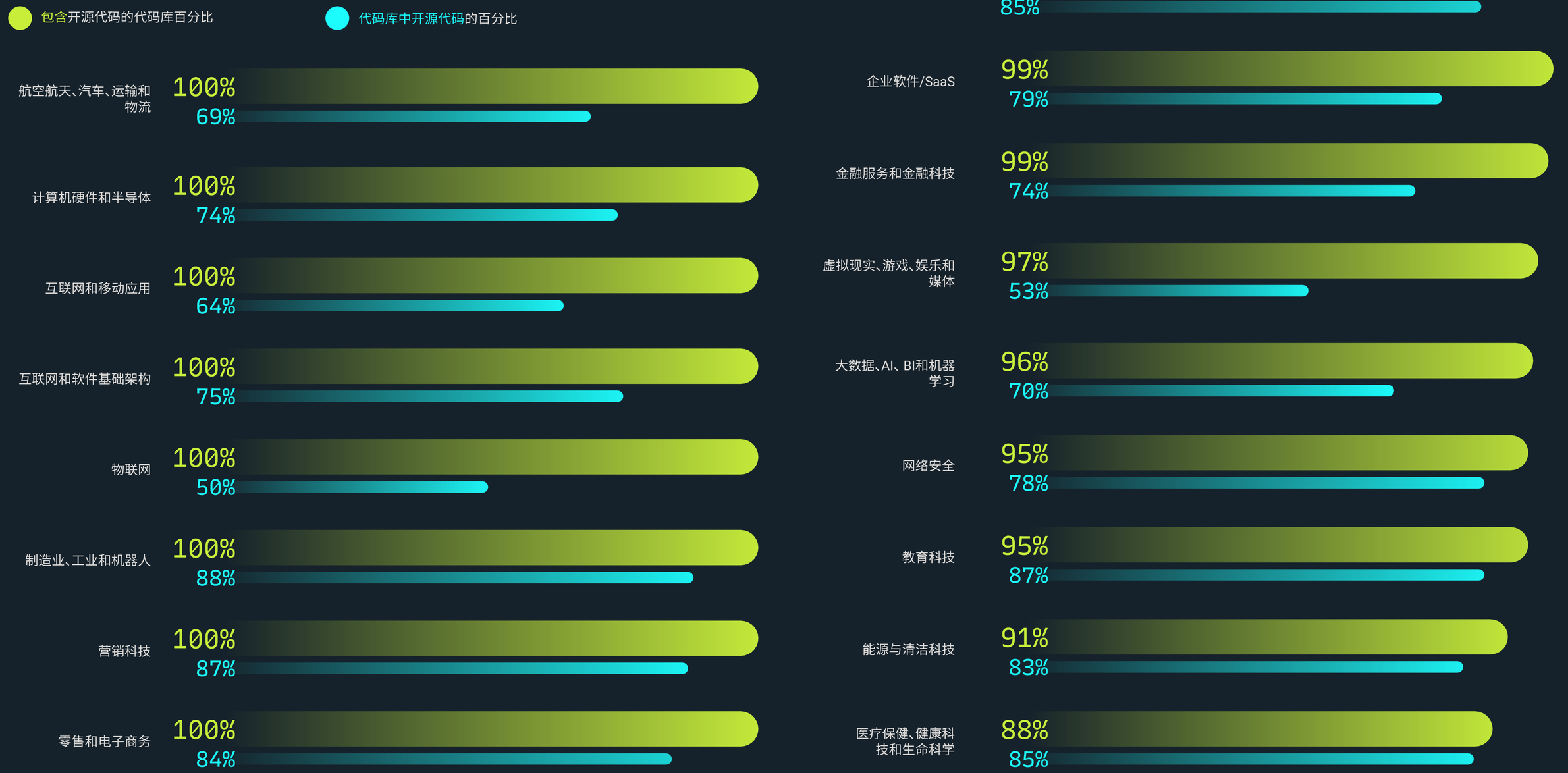
74%

经过风险评估的代码库中，74%包含高风险漏洞

91%

经过风险评估的代码库中，91%包含了比最新版本落后10个以上版本的组件

图1：本次研究审查的1,067个代码库，按行业划分



开源漏洞与安全

关于本次审计的说明

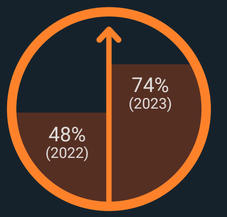
所有的Black Duck审计都会检查开源许可证的合规性，但客户可以自行决定放弃该审计的漏洞/运营风险评估部分。2023年，Black Duck审计服务团队共进行了1,067次审计。在这些审计中，88%的客户（936家）接受了安全和运营风险评估。在2024年的OSSRA报告中，“开源漏洞与安全性”以及“影响开源风险的运营因素”部分的数据基于包含风险评估的936个代码库，而“开源许可”部分的数据则基于全部的1,067个代码库。

在Black Duck审计服务团队为今年的OSSRA报告分析的1,067个代码库中，有96%包含开源代码。所有被审查的源代码和文件中，有77%来自开源代码。

今年，给定应用中的开源组件的平均数量为526个 — 证明自动安全测试非常重要甚至绝对必要的实际证据。如果只有区区几个组件，则手工测试也许是可行的，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，需要使用软件组成分析(SCA) 之类的自动化解决方案。与手工测试不同，自动安全测试可以快速且一致地执行，允许开发者在开发过程的早期阶段识别问题，而不会影响交付进度或生产力。

在包含风险评估的代码库中，84%包含至少一个已知开源漏洞。这些代码库有74%包含高风险漏洞，与2022年相比有显著增长，2022年只有48%的代码库包含高风险漏洞。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。

84%的代码库包含至少一个开源漏洞



包含高风险漏洞的代码库数量比去年增加了54%

2022至2023年间，高风险漏洞增加了54%（26个百分点），造成这种情况的原因可能不止一个。例如，有可能是经济衰退和随之而来的裁员导致用于寻找和修补漏洞的人员数量减少。此外，审查发现，91%的代码库包含了比最新版本落后10个或更多版本的组件，由此可以得出一个简单的结论：绝大多数开源软件使用者并没有更新他们使用的组件。

49%的代码库中包含过去24个月内未进行任何更新的组件，1%的代码库中包含至少12个月未被代码维护人员更新/修补的组件。

广义上说，术语“维护者”指的是那些领导开源项目的贡献者。他们可能是决定编译或发布哪些部分源代码的最终决策者，他们可能负责所有的代码审查工作，并在其名下托管较小的项目，他们也可能对项目的方向做出最终决定。他们的日常工作可能有所不同，但都包括审查拉取请求和其他提交、发布新版本的软件、分类和处理安全修复以及社区管理和协调。

大多数维护者都会努力使其参与的开源项目保持最新状态。实际上，许多公司还专门雇人来维护公司软件所依赖的开源项目。开源使用者也需要具备同样的勤奋精神。他们需要时刻关注使用的版本，定期进行更新，并在使用开源软件时注意软件卫生 — 只从那些拥

有健康的维护者和贡献者生态系统的项目下载。

“通用缺陷列表”(CWE)和“公共漏洞和暴露”(CVE)是用于识别和分类软件中安全缺陷和漏洞的常用列表。Black Duck Security Advisories (BDSA) 是新思科技的专有报告，旨在为客户提供比国家漏洞数据库 (NVD) CVE通知更高水平、更及时、更详细的信息。BDSA针对影响客户SBOM中项目的漏洞提供可操作的建议和细节，以帮助确保他们全面了解开源代码漏洞可能带来的风险。

如图2所示，CWE-707是CWE 20、79、80、97、937的支柱。CWE-707涉及从上游组件读取数据或发送数据到下游组件之前，没有满足安全需求。不能正确地净化输入可能会带来跨站脚本 (XSS) 和SQL注入等漏洞利用攻击。XSS是一种常见且危险的漏洞利用，与本报告中

Top 10的漏洞中，有8个都可以映射到CWE的同一个支柱缺陷，即CWE-707。也就是说，没有满足安全需求可能导致跨站脚本和SQL注入等攻击。

强调的大多数Top 10漏洞有关。

当攻击者利用网站中的漏洞发送通常使用JavaScript编写的恶意的畸形代码时，就会出现跨站脚本攻击。由于该输入没有正确净化或转义，导致攻击者可以操纵原本值得信赖的主机执行恶意任务。不过，大多数XSS攻击的最终目标并不是主机本身，而是Web应用的其他用户。恶意脚本一旦注入，便可用来窃取敏感信息，如会话cookies。

XSS不仅是十大漏洞之一，也是OWASP Top 10列表中的一个漏洞 — 该列表列出了10种最严重的Web应用安全风险。在其列表中，OWASP将XSS归类为“A03:2021 - 注入”。XSS漏洞的普遍存在与组织机构日益依赖基于Web的应用与客户交互有关。例如，许多电子商务公司、银行、互联网服务提供商和保险公司等组织机构都提供Web体验来吸引客户和合作伙伴。

数据再次清楚地表明，开发团队需要改进开源组件更新工作，特别是在涉及jQuery等常用的开源组件时。使用更易受攻击的旧版开源软件，后果可能很严重。例如，在审计发现的十大漏洞中，排名第二的BDSA-2020-0686 (CVE-2020-11022) 是影响从jQuery 1.2到3.5.0之间所有版本的XSS漏洞。该漏洞允许将不可信来源的HTML传递给jQuery的一种DOM操作方法 — 即使经过了清理也不例外 — 并且可能执行不可信的代码。

这个问题在jQuery 3.5.0中得到了修复，但正如我们的数据所示，有三分之一的被测代码库仍在使用易受攻击的jQuery版本。恶意数据可能被用来入侵系统，或者泄露密码和信用信息等敏感数据。正如前面提到的，跨站脚本是应用中最常见的漏洞之一，通常用于对浏览器中的各种语言解释器（如HTML或JavaScript）发动代码注入攻击。

jQuery并非天生不安全。事实上，它是一个维护良好的开源库，拥有庞大的用户、开发者和维护者社区。但是，随着普及度的提高，问题

缓解风险：安全地使用jQuery和其他常用开源软件的技巧

在审计中现的10大开源组件中，全部都是使用JavaScript编写的。审计中发现的大部分漏洞也与JavaScript库相关，尤其是过时版本的jQuery JavaScript库中的漏洞。

- 使用最新版本的jQuery。正如本报告所示，jQuery的旧版本通常都存在安全漏洞。
- 考虑订阅安全咨询服务，如Black Duck Security Advisories，以获取最新的漏洞信息。jQuery时不时就会出现新的安全漏洞。
- 使用安全的编码框架来帮助您识别并避免代码中的潜在安全漏洞。
- 使用自动安全测试，包括静态分析、软件组成分析和动态分析，贯穿整个软件开发生命周期来解决代码质量和安全问题。

往往也会随之而来。审计结果显示，jQuery是最有可能存在漏洞的组件，尽管本报告中列出的每个jQuery漏洞都已经有了可用的补丁。对于jquery用户（实际上是所有开源软件的用户）来说，意识到旧版本软件可能带来的潜在安全风险并采取措施减轻这些风险是非常重要的。

采取措施，防止漏洞进入您的软件供应链

- 创建并维护软件物料清单 (SBOM)。在防范软件供应链攻击的斗争中，拥有一个列出开源组件的准确并实时更新的SBOM，对于评估风险以及确保代码质量、合规性和安全至关重要。全面的SBOM列出了您的应用中包含的所有开源组件，以及这些组件的许可证、版本和补丁状态，构成了防范供应链攻击的有力武器，包括那些使用恶意软件包的攻击。
- 了解最新情况。确保您有办法了解新发现的恶意软件包、恶意软件和公开的开源漏洞。查阅新闻报道或定期发布的通知，以便针对影响SBOM中开源组件的问题获得实用性的建议和详细信息。
- 开展代码审查。在将软件引入项目之前，仔细检查其代码，看看是否存在任何已知漏洞。为了获取更多的信息，您可以考虑对源代码进行静态分析，以发现未知的安全缺陷。
- 积极主动。一个组件今天没有漏洞，不代表明天也没有。故意设计的恶意软件包可能永远不会被发现或标记为“可能有漏洞”。所以，在使用之前，请检查组件的健康状况和来源，以免将来出现安全问题。
- 使用自动化的软件组成分析 (SCA) 工具。SCA工具可以自动执行软件安全问题的识别、管理和缓解任务，使开发者能够专注于编写代码。此类工具可以评估开源和第三方代码。

Top 10的安全漏洞中，有8个可以追溯到同一个CWE

CWE项目将“支柱缺陷”定义为最高级别的缺陷，是与之相关的所有类别/变体缺陷的基础。

图2：Top 10 CVE/BDSA

39%

BDSA-2021-3651
Affected Component: jQuery
Associated CWE: CWE-546

29%

BDSA-2014-0063
Affected Component: jQuery
Associated CWE: CWE-79
Pillar CWE: CWE-707

33%

BDSA-2020-0686 (CVE-2020-11022)
Affected Component: jQuery
Associated CWE: CWE-707

28%

BDSA-2015-0567
Affected Component: jQuery
Associated CWE: CWE-937
Pillar CWE: CWE-707

33%

BDSA-2020-0964 (CVE-2020-11023)
Affected Component: jQuery
Associated CWE: CWE-80
Pillar CWE: CWE-707

17%

BDSA-2019-0423 (CVE-2019-8331)
Affected Component: Bootstrap
Associated CWE: CWE-97
Pillar CWE: CWE-707

32%

BDSA-2019-1138 (CVE-2019-11358)
Affected Component: jQuery
Associated CWE: CWE-20
Pillar CWE: CWE-707

17%

BDSA-2020-1173 (CVE-2020-7656)
Affected Component: jQuery
Associated CWE: CWE-80
Pillar CWE: CWE-707

30%

BDSA-2017-2930 (CVE-2015-9251)
Affected Component: jQuery
Associated CWE: CWE-79
Pillar CWE: CWE-707

17%

BDSA-2022-1987 (CVE-2022-31160)
Affected Component: jQuery UI
Associated CWE: CWE-79
Pillar CWE: CWE-707

为何某些BDSA没有CVE

公共数据源（如国家漏洞数据库 (NVD)）是获取公开披露的开源软件漏洞信息的首选渠道。但是，任何NVD CVE条目的报告都可能存在滞后。及时性一直都是影响NVD发布安全漏洞数据能力的因素之一。事实上，从漏洞首次披露到在NVD上发布，通常会间隔很长一段时间，一些研究报告称，从漏洞首次披露到在NVD上发布的平均时间间隔长达一个月之久。

NVD的另一个问题是，它提供的漏洞数据经常不完整。NVD中发布的许多CVE都不包括受影响的版本范围，并且通常太短，无法发挥作用。这通常是因为缺乏研究此类条目的资源造成的。

显然，仅仅依靠NVD来获取漏洞信息是不明智的。许多商业SCA解决方案不仅可以提供比NVD更丰富的漏洞数据，而且还可以更准确地发现问题，并在必要时帮助您更快地修复它们。例如，如果您正在使用新思科技的SCA解决方案Black Duck，则可以利用新思科技安全研究团队开发的开源漏洞通告“BDSA”。这些通告通常能够更早地通知您影响代码库的漏洞 — 提前于NVD几天甚至几周。BDSA还提供更完整的漏洞数据，提供超越当今任何其他商用产品的安全洞察、技术细节和升级/补丁指导。例如，对于我们在2023年Top 10漏洞列表中列出的漏洞，有三个BDSA在NVD中找不到对应的CVE。

BDSA-2021-3651

根据BDSA，某些版本的jQuery中包含对被劫持域名的注释引用。这是一个安全问题，最稳妥的解决方法就是删除被劫持域名的链接，正如第3.6.1版jQuery所做的那样，因为如果用户不知道域名的状态，他们仍然可能在尝试访问被劫持的网站时遭受未知攻击。虽然这些网站不能通过运行代码来访问，但标记这个问题还是有价值的，因为开发者或任何拥有访问权限的人都有可能不小心点开恶意网站。

新思科技CyRC团队建议将此BDSA标记为“信息性的”。当供应商提供的“修复”仅仅是在代码或产品文档中添加警告时，或者供应商拒绝了CVE或对发现的漏洞提出了质疑，认为这是组件的预期/设计行为时，可以使用这种标签。

类别

CWE-546: 可疑的注释

CVSS v3.1 评分

5.10

BDSA-2014-0063

这是一个较早的漏洞，最初在2014年1月被提出，与jQuery中因为缺乏用户提供的输入验证而导致的潜在XSS漏洞有关。该漏洞可能允许攻击者注入任意的web脚本并窃取受害者的会话cookies。

根据BDSA，函数会将HTML字符串解析为DOM节点的数组。传递给该函数的事件属性中所包含的任何脚本都会立即执行。如果函数调用者在将不可信的输入传递给函数之前没有正确地清理它，则可能导致XSS攻击的风险。攻击者可以利用这一点，制作一些恶意的HTML，然后让受害者访问。如果受害者的浏览器处理了这些代码，那么，其中包含的任何Web脚本都会在其系统上执行。

该漏洞在jQuery 3.0.0-rc1中得到了缓解。但这种缓解并没有清理恶意的输入，仍允许脚本执行。它只是修改了解析器的默认行为，使其在上下文未指定或者指定为“null/undefined”的情况下，创建一个新的文档。这样就会延迟被解析的HTML的执行，直到其被注入到文档中，从而给jQuery开发者提供了机会，使他们可以在函数调用后使用工具遍历已创建的DOM，并移除不安全的内容。

类别

CWE: 79: 在网页生成过程中不当地净化输入（“跨站脚本”）

CAPEC-588: 基于DOM的XXSS

CVSS v3.1 评分

8.60

BDSA-2015-0567

这也是一个较早的漏洞，与容易受到任意代码执行攻击的jQuery有关 — jQuery版本使用了未修补的UglifyJS解析器，容易受到恶意JavaScript文件发起的任意代码执行的攻击。最终，这可能允许攻击者运行恶意代码。

该漏洞已在 1.12.0 和 2.2.0.版本中修复。

类别

CWE Category A9: 使用包含已知漏洞的组件

CAPEC-251: 本地代码包含 (Local Code Inclusion)。The Common Attack Pattern Enumeration and Classification [CAPEC] 是公开的攻击模式目录，提供了全面的模式架构和分类体系。

CVSS v3.1 评分

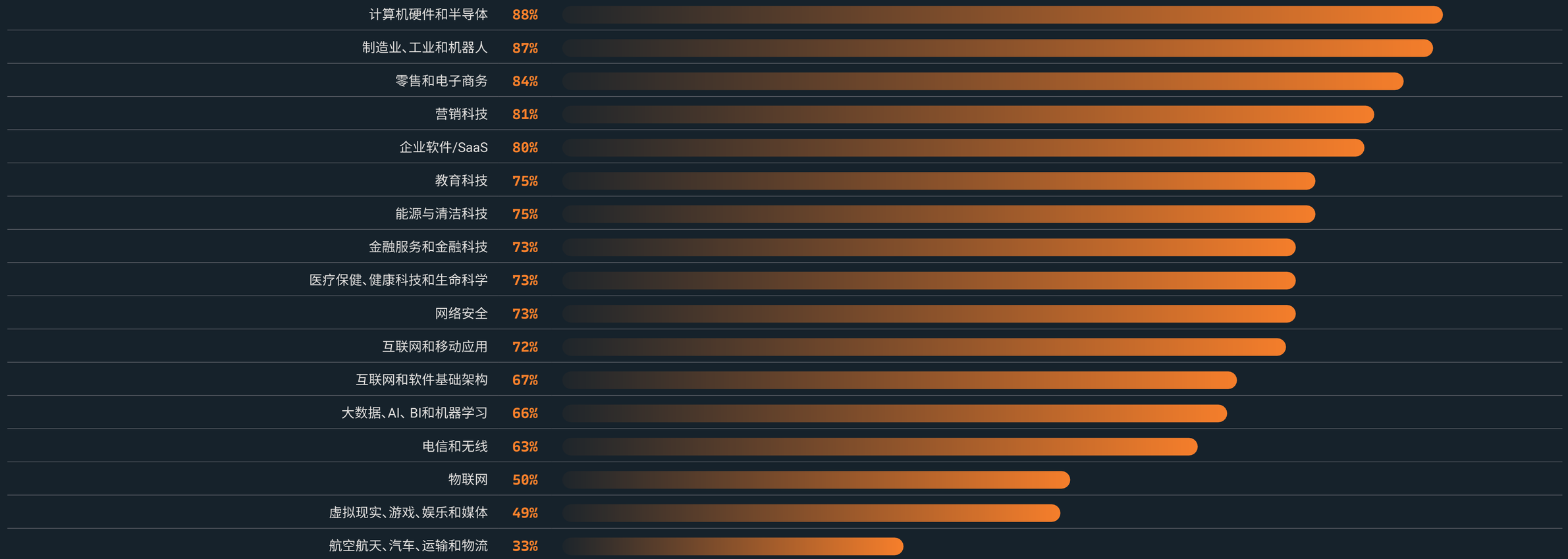
7.9

按行业划分的漏洞情况

与“计算机硬件和半导体”行业相关的代码库中，有88%包含高风险漏洞（严重程度评分为7分或更高），紧随其后的是“制造业、工业、机器人”以及“零售和电子商务”，分别为87%和84%。

每个行业都有类似发现。即使是占比最低的行业 — 航空航天、汽车、运输和物流 — 也仍然令人不安，该行业有1/3的代码库中包含高风险漏洞。如图1所示，我们在每个行业的代码库中都发现了开源代码，而且占比很高。图3表明，这些代码库中还包含组织机构尚未修补的大量已知开源漏洞，容易被利用。

图3：包含高风险漏洞的代码库的百分比

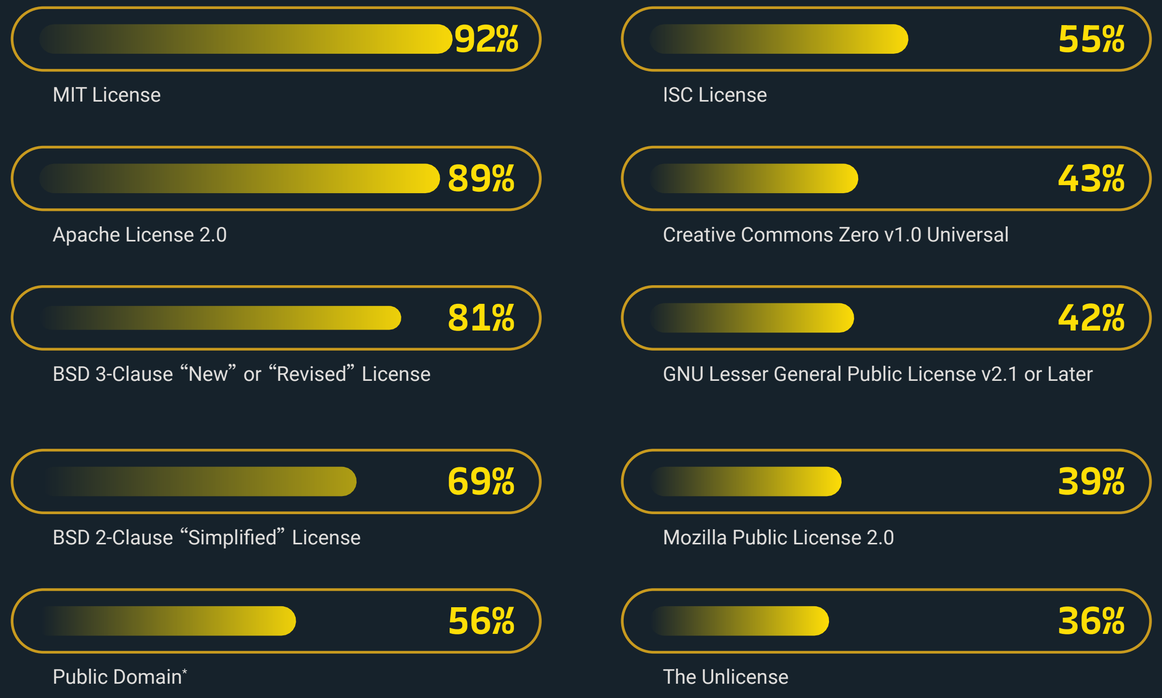


开源许可

有效的软件供应链管理需要同时考虑许可证和安全合规性。您使用开源组件和库来构建软件，并且知道这些组件受到开源许可证的约束，但您是否清楚这些许可证的具体内容?即使您的软件只涉及一个不合规的许可证，这也可能导致法律问题，失去利润丰厚的知识产权，花费大量时间进行补救并推迟产品的上市时间。

Black Duck审计服务团队发现，在2023年审计的代码库中，超过一半 (53%) 包含存在许可证冲突的开源软件。

图4：代码库中发现的Top 10许可证的百分比



*组件附带的声明表明它属于公共领域，但没有使用特定的公共领域许可证，比如Unlicense或Creative Commons

在Black Duck审计服务团队于2023年审计的开源软件中，有92%使用了MIT许可证。作为允许在专有软件中重用的宽松许可证，MIT许可证具有与其他软件许可证高兼容和低风险的特点。可以肯定的是，如果您在软件中引入第三方组件，则很可能会涉及一些常用的宽松许可证，如MIT、Apache、BSD、ISC和Unlicense。

需要注意的是，诸如“低风险”之类的术语只是一个参考，不能作为决定使用哪些开源软件的依据。例如，尽管Apache 2软件通常被认为使用了低风险的许可证，可以引入到使用GNU General Public License 3.0 (GPLv3)的项目中，但GPLv3软件却不能在Apache项目中使用。这是因为Apache软件基金会的许可证理念与GPLv3作者对版权法的解释导致了许可证之间的不兼容。对于开发者来说，最稳妥的做法是咨询本公司的企业政策和法务团队，以获得有关许可证合规问题的具体指导。

在2023年的审计中，Creative Commons是引发许可证冲突的最主要的原因。仅Creative Commons ShareAlike 3.0 (CC-SA 3.0)就引发了17%的许可证冲突。

Black Duck审计中频繁发现“代码片段” — 复制粘贴到源代码中的代码行。它们通常来自颇受欢迎的博客网站Stack Overflow，该网站根据Creative Commons ShareAlike许可证自动授权所有可公开访问的用户贡献。遗憾的是，这个一揽子许可证也涵盖了网站上发布的代码片段。我们之所以表示“遗憾”，是因为这些许可证根本不是针对软件设计的，Creative Commons在其常见问题解答中明确表示：“我们不建议对软件使用Creative Commons许可证。”某些情况下，CC-SA许可证可以被理解为具有类似于GNU Public License的“病毒”效应（即，任何源自Copyleft许可作品的作品也必须适用相同的Copyleft条款），这可能会引起法律方面的担忧。

了解许可证风险

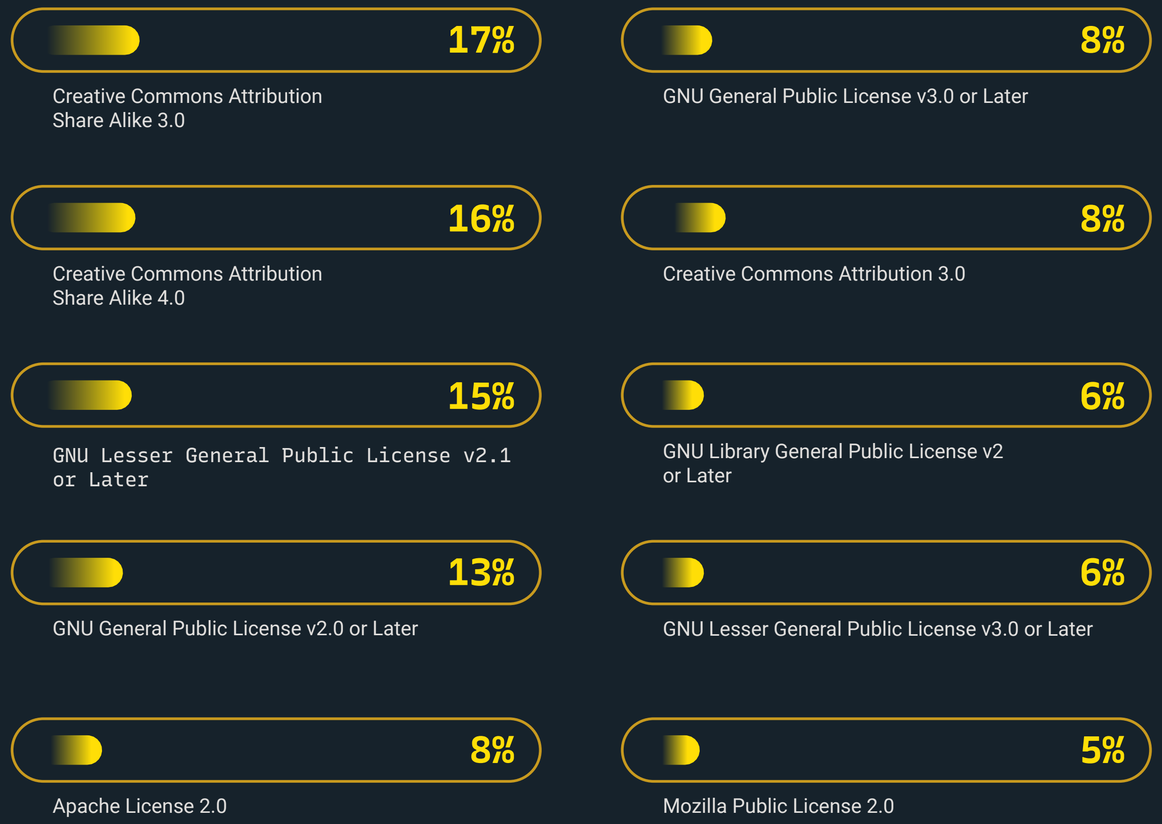
在美国和许多其他地方，创造性的工作（包括软件）默认即受专有版权的保护。未经创作者/作者以授权许可证的形式明确允许，任何人对该等软件的使用、复制、分发或修改均不合法。

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。如果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，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Copyleft许可证。如果商用闭源软件中包含由GPL许可的代码，就会产生冲突。

标准开源许可证的变体或定制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。JSON许可证基于宽松型MIT许可证，只不过添加了“该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”很难界定，许多律师都建议避免使用基于此类许可模式的软件，尤其是在并购的情况下。

在2023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，与去年的审计结果基本相同。在开源审计中，经常会在一些不像Stack Overflow那样有明确服务条款或软件条款的网站中发现代码片段。导致开源代码缺乏许可证条款的另一个常见原因是开发者使用了代码片段，但却没有将该片段的相关许可证一起带过来。另外，随着AI辅助编码工具的使用越来越普遍，也会导致没有关联许可证的代码出现问题（详见下一节）。

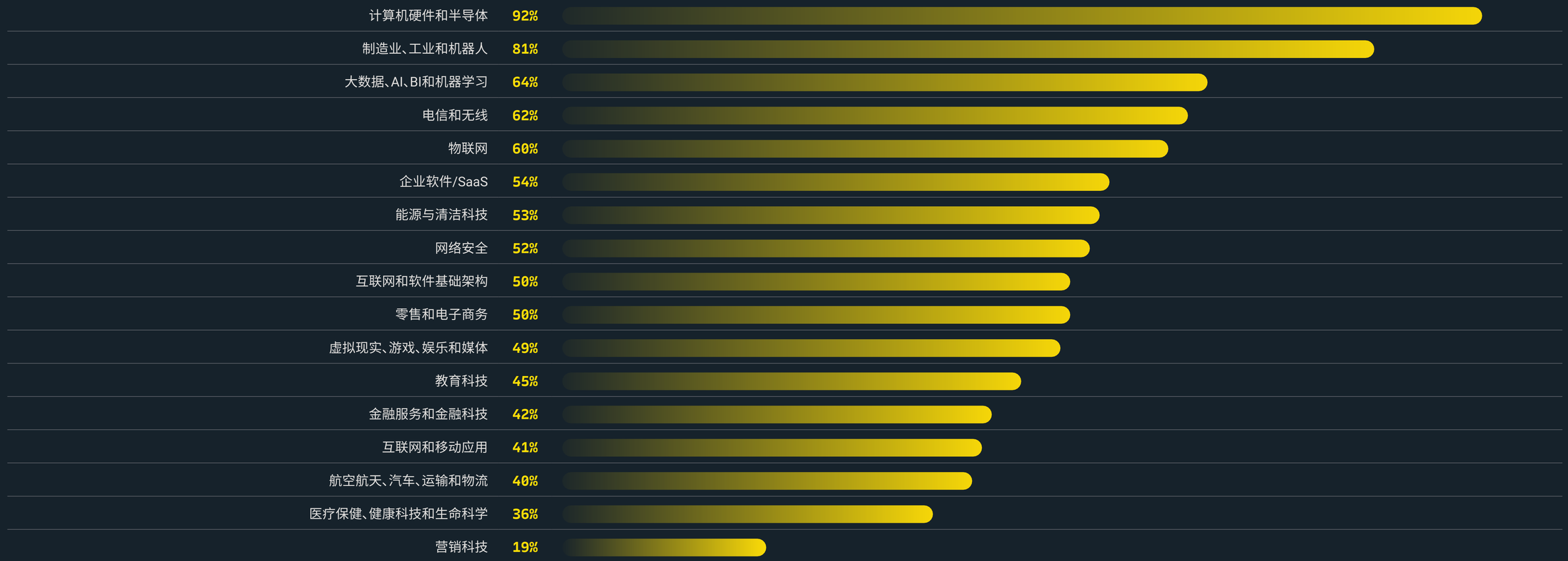
图5：存在冲突的Top 10许可证的百分比



导致多个行业具有如此高比例开源许可证风险（见图6）的原因可能包括：

- 许多冲突较多的行业都倾向于将其软件作为本地产品进行许可和分发。许多限制性的许可证只适用于这种方式分发的软件。而冲突较少的其他行业则更多地采用基于订阅或SaaS类型的部署方式，这些部署通常不被视为“分发”，也不受同样的许可条款的限制。
- 半导体和硬件公司严重依赖软件和固件，其中大部分都包含开源代码（见图1）。复杂的片上系统设计可能包含不同来源的数百万行代码。在这种规模下跟踪许可证和义务极具挑战性。
- 开源在底层系统软件、固件和驱动程序等方面非常普遍，这些都是硬件产品不可或缺的一部分。这些软件很多都是基于GPL类型的Copyleft许可证，分发时需满足程度极高的共享要求。
- 软件供应链在公司之间以及硬件设计者和制造商之间共享固件、驱动程序和工具，从而导致开源传播，但没有通过SBOM清单跟踪其来源或许可证。

图6：存在许可证冲突的代码库的百分比



防范AI编码工具引入的安全和知识产权合规风险

随着AI驱动的编码推荐工具的使用，围绕生成代码的所有权、版权和许可问题也随之产生。例如，一项针对GitHub、Microsoft和OpenAI的集体诉讼声称，GitHub Copilot —— 一种基于云的AI工具，为开发者在编码时提供自动完成式的推荐 —— 违反了版权法和软件许可要求。该诉讼还声称，Copilot推荐的代码使用了未经许可的内容，没有归属、版权声明，或者不遵守原始许可条款。

Copilot案例凸显出使用AI生成代码的法律复杂性。对软件开发者来说，在法律或政府做出解决这一问题的决策之前，避免使用AI辅助编码工具显然是避免因许可证或版权侵权而被起诉的最安全方法。

如果开发者想要使用AI辅助工具，应该谨慎行事，避免不必要的风险。至少，他们应该让公司询问AI工具提供商，看看推荐的内容中是否包含受开源许可证约束的源代码，如果是，是否可以将这些代码标出或从推荐中排除。

另一种解决办法是使用新思科技Black Duck等现成的代码扫描工具，利用代码片段分析功能来扫描源代码，并将每一行代码与它们可能来自的任何开源项目进行匹配。使得开发团队可以识别AI代码助手引入的开源代码所适用的许可证和相关条款。

软件供应链治理中开源许可证管理的最佳实践

- 对软件中的所有第三方软件组件进行全面清点，包括开源和商业软件。
- 注意AI辅助编码工具可能会产生违反许可证条款和侵犯知识产权的代码。
- 评估每个组件的许可条款和条件，并评估它们是否符合产品的预期用途。
- 检查不同组件之间的许可证兼容性，因为有些许可证彼此可能不兼容。
- 使用自动扫描工具来识别和跟踪每个组件的许可义务和限制。
- 实施一个流程，以确保许可证始终合规，包括定期许可证扫描和许可证合规程序的周期性审查。
- 针对新的或不熟悉的许可证建立审查流程和工作流。
- 确保法务、技术和业务的相关人员之间能够有效沟通，以便合理地安排和执行许可证清理工作（也就是公司决定是否可以使用某个组件的许可证的过程）。
- 记录所有的许可证清理活动，包括许可证评估与合规程序，以保留合规工作的证据，方便以后的审计。

影响开源风险的运营因素

理想情况下，开源软件的使用者应该只使用那些有着强大社区支持的组件。例如，每天都有来自数百个组织的成千上万名开发者改进Linux。然而，在Black Duck审计服务团队对经过风险评估的936个代码库进行审查时，发现有49%的代码库中包含了两年内未进行任何更新的开源软件。如果一个项目不再有人维护，尤其是较小的项目，就不会有功能升级、代码改进或者已知安全问题的修复。

这在开源项目中并不少见。一些报告显示，在2022年还在维护的Java和JavaScript开源项目中，有近20%在2023年不再有人维护，使得这些项目面临着漏洞利用和攻击的风险。开源软件很大程度上是志愿贡献者和维护者的产物。虽然有些组织（如Microsoft、Red Hat和Google）制定了一些激励计划来促进开源项目的维护和参与，但绝大多数公司并没有这样做。当维护者停止维护项目时，必然会导致安全风险升高。

在2023年分析的代码库中，有88%经过风险评估

在经过风险评估的代码库中，有49%包含了两年内未进行任何更新的开源代码

开源软件的使用者需要改进维护实践

Black Duck审计服务团队在2023年对经过风险评估的936个代码库进行审查时发现，91%的代码库中包含了比最新版本落后10个或以上版本的组件。

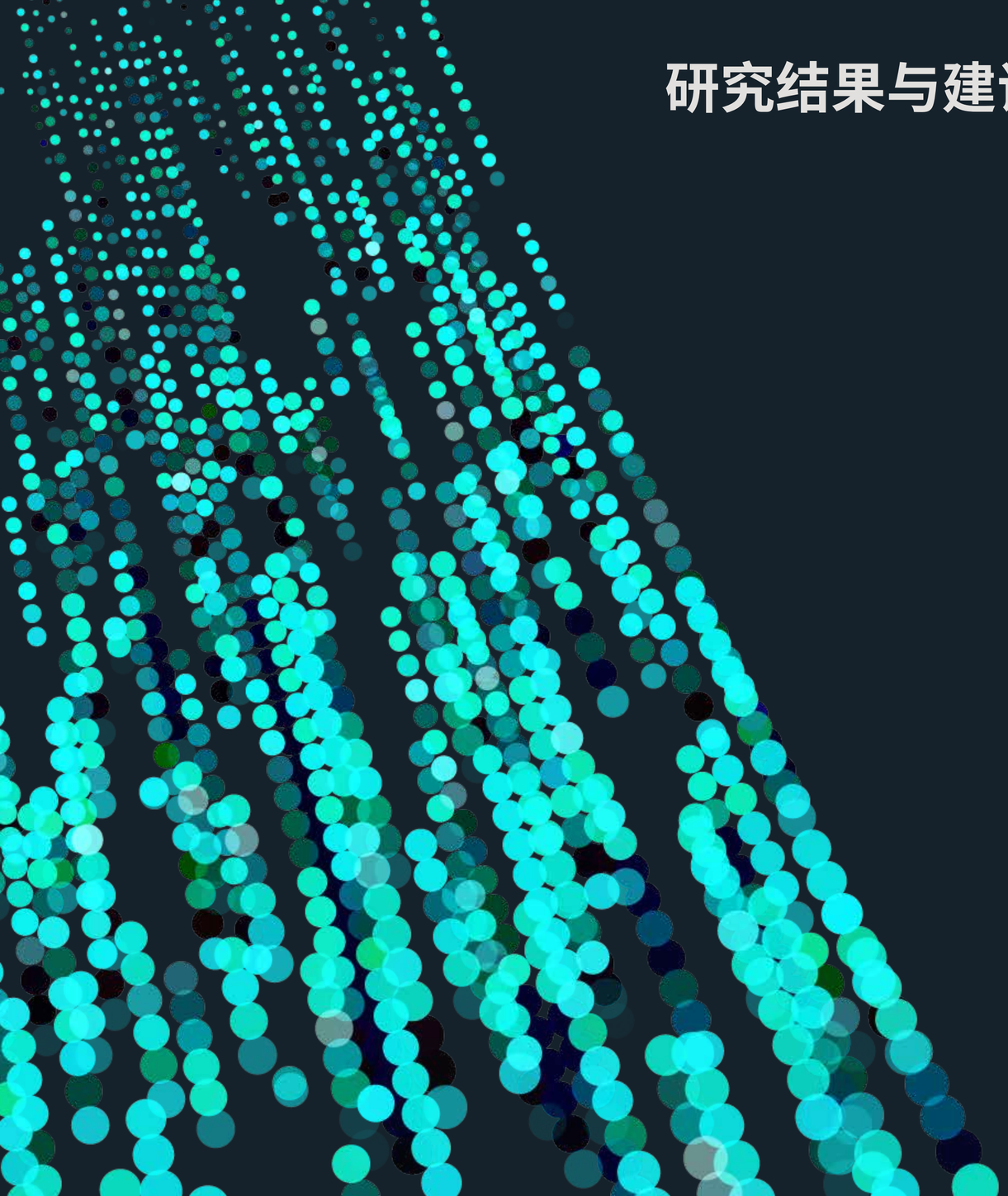
开源软件的使用者有时候不更新开源组件，这可能有一些合理的理由。如果他们知道自己使用的开源软件的情况，则可以做出正确的决定，但遗憾的是，很多人并不知道 —— 开发团队可能会认为，更新开源组件可能带来的意外风险大于升级到新版本所带来的好处。例如，嵌入式软件受到只能从外部来源引入的攻击风险可能很小。

这也可能是时间/资源的问题。许多团队在构建和测试新代码时，已经没有多余的时间和资源去更新现有软件，除非是非常紧急的问题。新思科技《2023年全球DevSecOps状况调查报告》指出，对1,000名IT安全专业人士进行的调查显示，28%的受访者表示其组织机构需要长达三周的时间来修补已部署应用中的重大安全风险/漏洞，另有20%的受访者表示，这可能需要一个月的时间。这些数字适用于所有的漏洞 —— 自有、商业、第三方软件和开源软件。

OSSRA报告近十年来一直都在强调，开源软件不同于商业软件—— 没有孰优孰劣的区别，只是单纯的不同 —— 需要不同的管理技巧。例如，商业软件和开源软件的补丁处理方式就大不相同。购买商业软件通常需要根据供应商管理程序进行一些审查，而开源软件可能只是由开发者自行下载。对开源软件的使用可能存在一些组织规则 —— 例如，只使用具有许可证的代码 —— 但在很多组织机构中，甚至连这样的指导都不存在。

使用商业软件的组织都知道，他们的软件会被自动“推送”补丁和更新，或者至少会收到供应商的通知，告诉他们有更新可以下载 —— 通常是非常重要的更新。但这种情况在开源软件中很少出现，开源软件的使用者需要自己关注组件的状态，自己去下载可用的新版本。

要保持对您使用的开源版本的了解，只有一种可行的解决方案，那就是制作准确、全面的开源软件清单，并且自动监测软件中开源组件的漏洞、更新情况和整体健康状况。



研究结果与建议

无论是个人开发者还是大型企业，都有责任维护软件供应链的安全以降低风险。随着软件供应链攻击数量的增加，有效地管理开源软件的使用、组件和依赖关系对于风险管理变得越来越重要。所有使用开源软件的组织机构 —— 正如本报告指出，这几乎包括了所有组织机构 —— 都应该主动管理开源风险，将其作为安全软件开发实践的一部分。

美国网络安全和基础设施安全局 (CISA) 于2023年底发布的《保障软件供应链安全：管理开源软件和软件物料清单的推荐做法》(Securing the Software Supply Chain: Recommended Practices for Managing Open Source Software and Software Bill of Materials) 为在软件供应链中使用开源软件提供了详细指导，包括：

- **安全团队通常会定义与开源组件相关的安全策略、流程和工具。**
  理想情况下，开发者应该从经过预先审核的内部仓库中选择具有所需功能的组件 —— 即已经使用SCA安全分析工具进行了初始漏洞评估 —— 然后在开发和/或构建阶段开展进一步扫描，以尽早发现问题。
- **跟踪开源软件的更新，并监控问题和漏洞。**
  当发现漏洞时，应评估受影响的软件，以确定组件的普及程度及其在产品中的使用情况。更新组件，或者，如果组件不再得到维护，应考虑寻找替代方案。
- **使用SBOM。**
  了解软件中包含哪些组件对于准确、完整地管理代码至关重要。SBOM是包含软件中组件细节和供应链关系的正式记录。SBOM可以提高软件的透明度并记录组件的来源。对于漏洞管理，SBOM可以支持漏洞的识别和修复。从代码质量的角度来看，SBOM的存在可能表明供应商在整个软件开发生命周期中使用了安全的软件开发实践。

第14028号行政命令 (EO) “提升国家网络安全”规定，软件供应商必须直接向采购方提供SBOM，或者在公共网站上公开其SBOM，并且政府和非政府方面可能都需要查看SBOM，以确保软件产品符合SBOM的最低要求。该行政命令还指示商务部以及国家电信和信息管理局 (NTIA) 发布《软件物料清单最低要求》(The Minimum Elements for a Software Bill of Materials (SBOM))，以简要说明SBOM所需的活动和数据，并给出满足SBOM要求的示例格式。该文件将SPDX和CycloneDX确定为两种最常用的机器可读SBOM格式。2023年中期，管理和预算办公室 (OMB) 发布了OMB-23-16备忘录，更新了早期的OMB备忘录，允许联邦机构对SBOM提出以下要求：

- SBOM可根据软件的重要性或其他标准而有所不同，具体由每个机构自行决定
- SBOM必须采用NTIA定义的其中一种格式

创建安全的软件开发框架

为了保障客户和用户的利益，软件生产商在保证软件供应链的安全方面扮演着重要的角色。美国国家标准与技术研究院 (NIST) 的“安全软件开发框架” (SSDF) 提出了一系列实践，作为以标准化方式安全开发软件的基准。美国政府已经暗示，符合NIST SSDF可能会成为美国政府直接或间接采购的所有软件的必要条件，而且软件供应商很可能需要在不久的将来自证其遵守SSDF。

评估工具（如新思科技SSDF评估服务 (Synopsys SSDF Readiness Assessment)）可以确定贵组织的软件开发实践是否与SSDF的实践和任务相一致，以便您可以自信地证明贵组织的软件开发过程符合SSDF标准。同样，新思科技SBOM服务建立在Black Duck审计服务流程的基础上，可以对您的软件进行全面的安全审计，并生成一个SBOM，这对那些还没有SBOM生成能力但却需要一个基准SBOM的组织来说是一项很有价值的服务。

软件生产商可能因为法规或合同的要求，需要提供经过审计的SBOM。软件使用者可能希望审查某个供应商制作的SBOM。这些情况下，需要一个声誉良好的第三方机构对软件进行审计。新思科技SBOM审计和验证服务就是基于这些验证过的流程对软件进行审计，并确认客户提供的SBOM是否准确地反映了供应链的情况。

了解代码的构成

本报告的研究结果总结如下：

- 在扫描的1,000多个代码库中，有96%包含开源代码
- 77%的源代码和文件来自开源
- 53%的代码库存在开源许可证冲突
- 84%的代码库在安全风险评估中发现了漏洞；74%有高风险漏洞
- 91%的代码库中包含比最新版本落后10个或更多版本的组件

无论贵组织是开发还是使用软件，几乎可以肯定的是，软件中都包含开源组件。您是否清楚地知道这些组件是什么，以及它们是否会带来安全或许可证风险?2024年度报告显示，96%的代码中包含开源组件，因此，了解代码的具体情况是非常重要的。当风险评估发现91%的代码库都在使用远远落后于最新版本的开源组件时，软件使用者需要更好地保持代码的及时更新，尤其是在涉及到常用的开源组件时。

始终及时更新开源软件与贵团队开发的代码同样重要。创建并维护一个SBOM，记录您的代码中包含了哪些软件组件，以及它们的版本、许可证和来源。定期升级开源软件，尤其是在使用了活跃项目的开源库的情况下。

如果缺乏对代码的全面了解，并且没有采用主动的软件卫生实践，您的软件就会暴露在开源漏洞和知识产权合规问题的潜在攻击之下。首先，您应该使用自动化SCA工具，在SDLC的早期阶段发现并解决安全、代码质量和许可证问题，因为您需要毫无疑问地知道代码构成情况。.

如果缺乏对代码的全面了解，并且没有采用主动的软件卫生实践，您的软件就会暴露在开源漏洞和知识产权合规问题的潜在攻击之下。

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。

欲了解有关新思科技软件完整性小组的更多信息，请访问：www.synopsys.com/software.

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。

CWE
通用缺陷列表（Common Weakness Enumeration, CWE）是由社区开发的列出软件和硬件缺陷类型的列表，分为三层。CWE涵盖了600多个类别，包括缓冲区溢出、路径/目录树遍历错误、竞争条件、跨站脚本、硬编码密码和不安全随机数等。

CVE
公共漏洞和暴露（Common Vulnerabilities and Exposures, CVE）列出了公开披露的信息安全缺陷的列表。

Black Duck Security Advisory (BDSA)
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。BDSA提供当天通知、实际可操作的漏洞消减指导和规避方案信息、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。

代码片段
代码片段是开发者可以复制粘贴到自己代码中的一小段可重用的代码。即使软件中只包含一小段的开源代码片段，软件用户也必须遵守与该代码片段相关的任何许可协议。

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，包括如何使用和重新分发这些组件，用于阐述最终用户义务的一组条款和条件。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。
>
> **著佐权许可证(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佐权许可证的开源代码应十分谨慎，因为它的使用可能会带来整个代码库的知识产权问题。

软件组成分析(SCA)
一种用于自动化开源软件管理流程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。

贡献者

《开源安全和风险分析报告》是由新思科技软件完整性小组的多个团队共同完成的，包括审计服务、咨询、研究、法律和市场团队。他们的辛苦付出使OSSRA在过去十年成为了有关开源代码质量、安全性及许可证合规性的权威报告。

特别感谢Nancy Bernstein、Scott Handy、Siobhan Hunter、Matt Jacobs、Natalie Lightner、Merin McDonell、Mike McGuire、Phil Odence、Rie Sekine、Liz Samet、Jenny Stout和Jack Taylor为今年的报告做出的贡献。

Rachel Bay连续九年参与了OSSRA报告的编制工作，并展现了她令人难以置信的设计才华。能够撰写这份报告是我的荣幸 — Fred Bals