2021 年开源软件供应链安全风险研究报告

国家计算机网络应急技术处理协调中心

2021 年 6 月

前　言

“开源”是指源代码、文档等设计内容开放的开发模式，是群智协同、开放共享、持续创新的理念和生产方式。2020 年，根据 Synopsys 发布的《开源安全和风险分析报告》显示，开源使用数量占比较高，在教育、金融、医疗等传统行业渗透率已超过 60%，开源软件已成为企业构建信息技术的重要选择。

国家政策上，2021 年 3 月 12 日，开源首次被明确列入《中华人民共和国国民经济和社会发展第十四个五年规划和 2035 年远景目标纲要》，支持数字技术开源社区等创新联合体发展，完善开源知识产权和法律体系，鼓励企业开放软件源代码、硬件设计和应用服务。

开源蓬勃发展一方面可以突破技术壁垒、推动创新，另一方面考虑到国际竞争关系错综复杂，开源软件安全作为软件供应链安全的重要环节，面临着安全漏洞、知识产权、软件供应链安全等相关风险。在此背景下，认识和了解开源安全风险情况是至关重要的。

国家互联网应急中心联合棱镜七彩开源安全研究团队持续对开源软件供应链安全进行跟踪分析。《2019 年开源软件风险研究报告》主要从 GitHub 热门开源软件视角出发，对开源软件安全风险进行了分析。本报告从全新视角带来开源安全风险新的发现与突破。报告共分为五部分，第一部分，首先介绍开源漏洞的发展现状及趋势；第二部分，聚焦开源组件生态库的安全风险；第三部分，重点围绕组件按依赖层级漏洞传播范围分析；第四部分，对文件级漏洞潜在安全风险及波及范围进行讨论；第五部分，对开源使用者和关注者如何在开源领域蓬勃发展下，更安全的拥抱开源生态提出了建设性意见。

目 录

前 言 1

一、开源漏洞发展现状及趋势 3

发现一：开源软件漏洞整体呈增长趋势，2020 年增长率略有下降 3

发现二：CVE 官方未收录的开源软件漏洞数逐年递增 4

发现三：开源软件漏洞由 POC 披露到 NVD 首次公开时间长达 11 年 4

发现四：近 4 年，高危及以上开源漏洞占比均超 40% 5

发现五：2020 年，最主要缺陷类型为 CWE-79 6

二、开源组件生态安全风险分析 8

发现六：开源组件生态中的漏洞数呈上涨趋势，2020 年环比增长 40% 8

发现七：近 6 年中 Maven 仓库漏洞数量最多 9

发现八：超半数仓库的漏洞数均较上年有所增长 10

发现九：2020 年高危漏洞占比最高，相比去年增加 2.6 倍左右 10

发现十：2020 年，含高危以上漏洞占比最多仓库是 Rubygems 12

发现十一：平均每版本漏洞最多的 TOP 25 组件约五成来自 Composer 仓库 12

三、组件漏洞依赖层级传播范围分析 15

发现十二：一级传播影响范围扩大 125 倍，二级传播影响范围扩大 173 倍 15

发现十三：npm 仓库中的组件经 2 轮传播，影响组件数量最多 16

发现十四：一级传播影响范围最广的仓库是 Composer 16

发现十五：二级传播影响范围最广的仓库是 Nuget 17

发现十六：传播影响范围最小的仓库是 Maven 18

四、开源文件潜在漏洞风险传播分析 20

发现十七：超 80%漏洞文件在开源项目具有同源文件 20

发现十八：漏洞文件在开源项目中传播范围扩大 54 倍 21

案例分析 21

五、开源安全风险建议 23

一、开源漏洞发展现状及趋势

开源软件具有代码公开、易获取、可重用的特点，这一特点是开源软件热度攀升的重要原因。随着开源软件的广泛使用，一旦软件发现安全漏洞，必将给开发、安全团队带来严峻的挑战。然而，开源漏洞信息往往散落分布在各大社区，很多漏洞信息不能及时被官方收录。同时，对于软件使用者，由于缺少漏洞信息跟踪能力，使得漏洞修复具有滞后性，提升了软件被攻击的风险，为软件供应链安全管控增加了难度。

本次研究收录了官方漏洞库、开源社区等渠道的数据[1]，并统一收录整理成开源漏洞知识库。通过从中选取 2015 年至 2020 年发布的开源漏洞为研究对象，本报告展示了近 6 年开源安全漏洞发展现状及趋势。

发现一：开源软件漏洞整体呈增长趋势，2020 年增长率略有下降

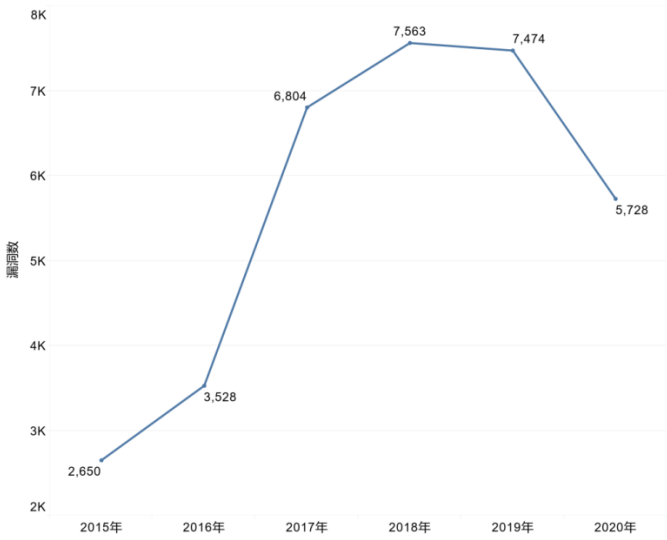

图 1　开源漏洞时间分布

根据调查结果，相比 2015 年漏洞数据，近 5 年的漏洞数量均有不

[1] 国家信息安全漏洞 CNVD 共享平台（http://www.cnvd.org.cn/）、美国国家漏洞库（https://nvd.nist.gov/）、通用漏洞披露库（https://cve.mitre.org/）等

同程度增长。2018 年是开源项目快速增长的一年，根据 GitHub 官方数据显示，GitHub 代码仓库中超过 1/3 的开源项目创建于 2018 年，2018 年新增开源漏洞数也创下近 6 年新高，新增 7563 个漏洞，相较于 2015 年翻了 2.85 倍；2017 年漏洞增长速度最快，环比增长率为 92.86%；2019 年与 2020 年增长率略有下降，2020 年发布的漏洞数较 2019 年发布漏洞数少了 1746 条。

发现二：CVE 官方未收录的开源软件漏洞数逐年递增

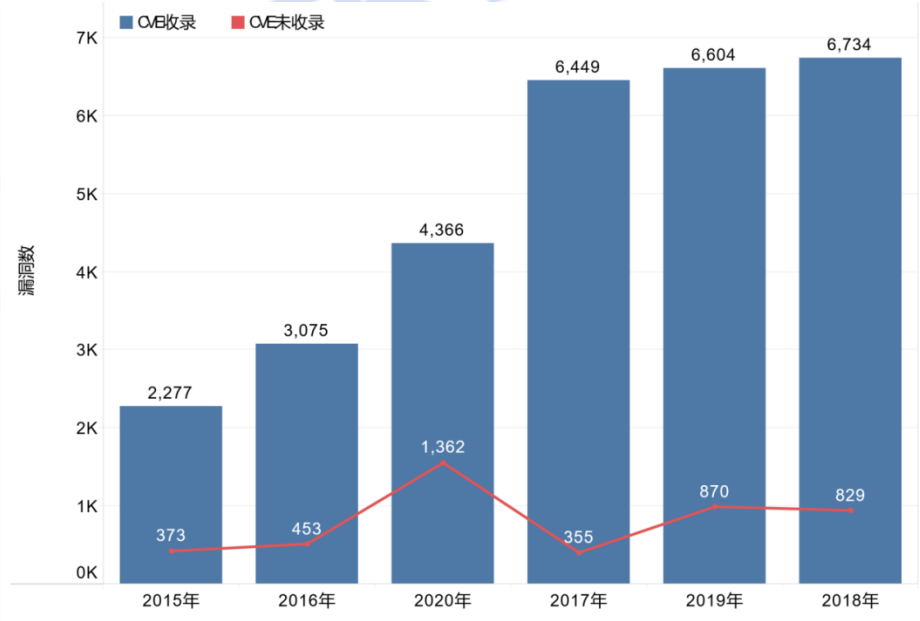

图 2　CVE 官方未收录开源漏洞情况

对 CVE 官方网站[2]进行统计，可发现 2020 年发布的开源漏洞中未被 CVE 官方收录漏洞有 1362 个，占 2020 年发布漏洞总数的 23.78%；CVE 官方未收录数据呈上长趋势，增长率逐年递增，2018 年环比 2017 年增长速度达 133.52%。

发现三：开源软件漏洞由 POC 披露到 NVD 首次公开时间长达 11 年

2020 年发布的开源漏洞中，编号为 CVE-2009-4067 的 Linux 内核

[2] https://cve.mitre.org

的 Auerswald Linux USB 驱动程序的缓冲区溢出漏洞由 POC 披露到 NVD 首次公开时间长达 11 年。POC 信息在 2009 年 10 月 19 日披露[3]；该漏洞于 2009 年 11 月 24 日获得 CVE 编号，但未公开漏洞具体信息；直到 2020 年 11 月 2 日 NVD 官方才将其发布。

开源软件的使用者仅关注官方漏洞库（如 NVD 等）可能无法及时获取漏洞信息，需综合考虑更多渠道的漏洞数据。

发现四：近 4 年，高危及以上开源漏洞占比均超 40%

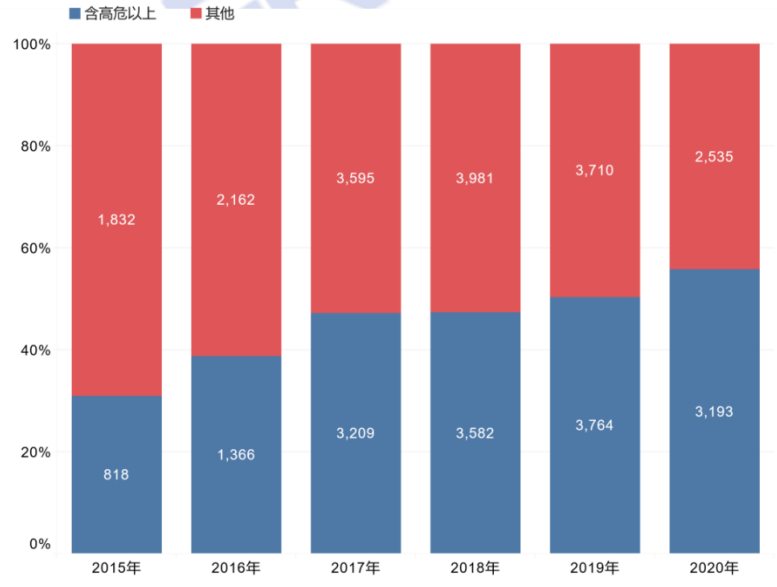

图 3　含高危以上漏洞占比

调查结果显示，近 6 年高危及以上漏洞占比逐年递增，从 2015 年占比 30.87%增长至 2020 年占比 56%；其中，2017 年至 2020 年高危及以上漏洞占比均超过 40%；2020 年，超危漏洞占比为 8.83%，高危漏洞占比为 46.91%，占 2020 年新增漏洞超 5 成。

[3] https://www.exploit-db.com/exploits/35957

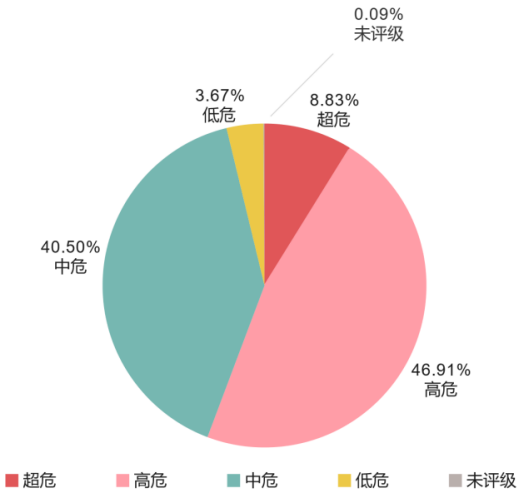

图 4　2020 年漏洞危害等级占比

发现五：2020 年，最主要缺陷类型为 CWE-79

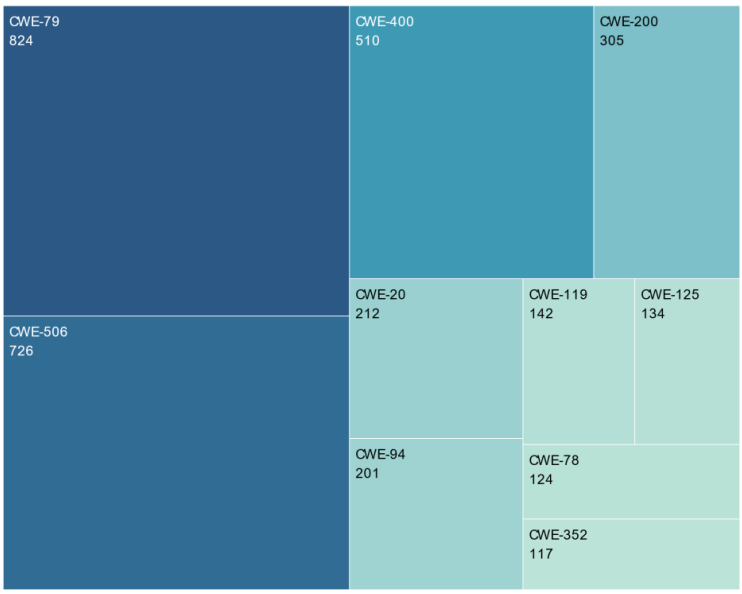

图 5　2020 年开源漏洞 TOP 10 CWE 缺陷类型

调查结果显示，缺陷类型 CWE-79 数量最多，占 2020 年新增开源漏洞的 14%左右。表 1 列出了 TOP 10 CWE 缺陷类型，这些缺陷类型很容易并被利用，往往通过系统信息暴露、窃取数据或阻止应用程序正常工作等方式，对系统造成安全风险。了解开源漏洞 Top10 CWE 可以帮助开发人员、测试人员、用户、项目经理以及安全研究人员深入

了解当前最严重的安全漏洞。

表 1　2020 年开源漏洞 TOP 10 CWE 缺陷类型

CWE 编号	中文名称	个数
CWE-79	在 Web 页面生成时对输入的转义处理不恰当（跨站脚本）	824
CWE-506	内嵌的恶意代码	726
CWE-400	未加控制的资源消耗（资源穷尽）	510
CWE-200	信息暴露	305
CWE-20	输入验证不恰当	212
CWE-94	对生成代码的控制不恰当（代码注入）	201
CWE-119	内存缓冲区边界内操作的限制不恰当	142
CWE-125	跨界内存读	134
CWE-78	OS 命令中使用的特殊元素转义处理不恰当（OS 命令注入）	124
CWE-325	缺少必要的密码学步骤	117

二、开源组件生态安全风险分析

开源组件生态蓬勃发展，重要原因是组件独立、可复用。组件化可以大幅度提高开发效率、可测试性、可复用性、提升应用性能。同时，组件化能够屏蔽逻辑，帮助迅速定位问题，易于维护和迭代更新。组件标准化使得优质好用的组件越来越多，用户也更愿意使用，形成一个良性循环的开源组件生态库。

开源组件被广泛使用，根据官方数据显示，Maven 仓库数据量已达 650 万+，Nuget 仓库累计下载量超 930 亿，Rubygems 仓库累计下载量超 712 亿，PyPI 仓库使用人数超过 49 万。

本报告选取了 CocoaPods[4]、Composer[5]、Go[6]、Maven[7]、npm[8]、Nuget[9]、PyPI[10]、Rubygems[11]这 8 个主流的仓库作为研究对象，分析近 6 年各仓库新增漏洞数据，帮助解开源组件生态安全风险情况。

发现六：开源组件生态中的漏洞数呈上涨趋势，2020 年环比增长 40%

根据调查结果，近 6 年开源组件生态中漏洞数逐年递增。其中，2020 年新增漏洞数为 3426，环比去年增长 40%；2017 年增长速度最快，环比增长 49%；近 3 年增长速度呈上升趋势，2020 年新增漏洞数是 2015 年的 4.48 倍。

[4] http://cocoapods.org/

[5] https://packagist.org

[6] https://pkg.go.dev

[7] http://maven.org

[8] https://www.npmjs.com

[9] https://www.nuget.org

[10] https://pypi.org/

[11] https://rubygems.org

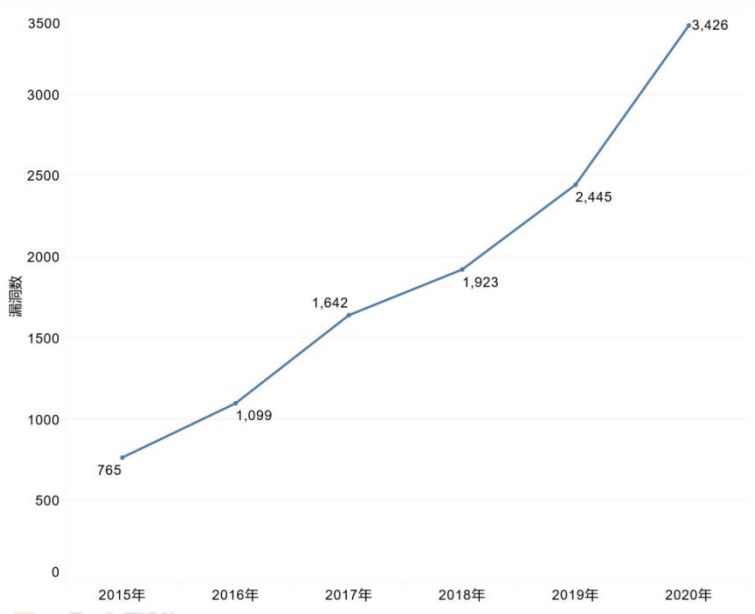

图 6　开源组件生态漏洞时间分布

发现七：近 6 年中 Maven 仓库漏洞数量最多

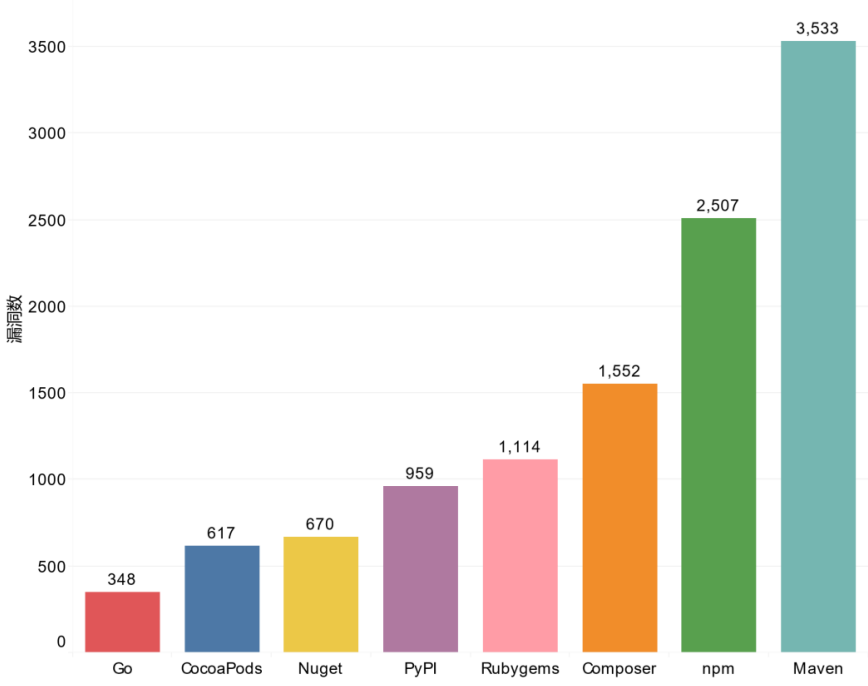

图 7　近 6 年中各组件仓库漏洞情况

调查结果显示，近 6 年中漏洞数量最多是 Maven 仓库，漏洞数量为 3533 个；Go 仓库漏洞数量最少，漏洞数量为 348 个；平均每个仓库漏洞数量为 1413 个。

发现八：超半数仓库的漏洞数均较上年有所增长

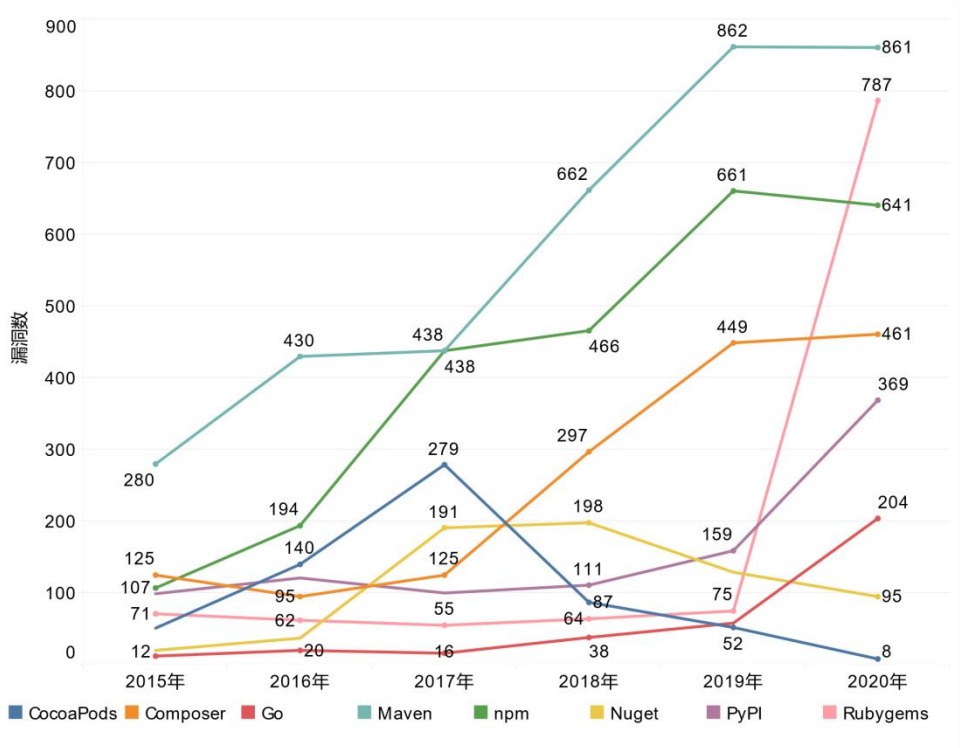

图 8　近 6 年各仓库漏洞分布图

调查结果显示，近 6 年，Composer、Go、Maven、npm、PyPI、Rubygems 6 种仓库的漏洞数均有不同程度的上涨；Rubygems 仓库漏洞增长速度最快，相比去年翻 10.5 倍；Go 仓库和 PyPI 仓库增长率其次，环比增长率分别为 252%和 132%；Maven 仓库 2020 年新增漏洞数基本与去年持平。

发现九：2020 年高危漏洞占比最高，相比去年增加 2.6 倍左右

调查结果显示，2020 年新增漏洞中，高危漏洞占比最高，数量为 1826 个；超危漏洞逐年递增，2020 年数量有所下降，环比下降 53%，2019 年新增数量最多，新增数量为 468 个；高危漏洞逐年递增，2020 年增长速度最快，相比去年增加 2.62 倍；中危漏洞呈现平稳增长趋势，2018 年增长速度最快，环比增长率为 48.13%；低危漏洞逐年递增，2020

年增长速度有所下降，2019 年新增数量最多，新增数量为 171 个。

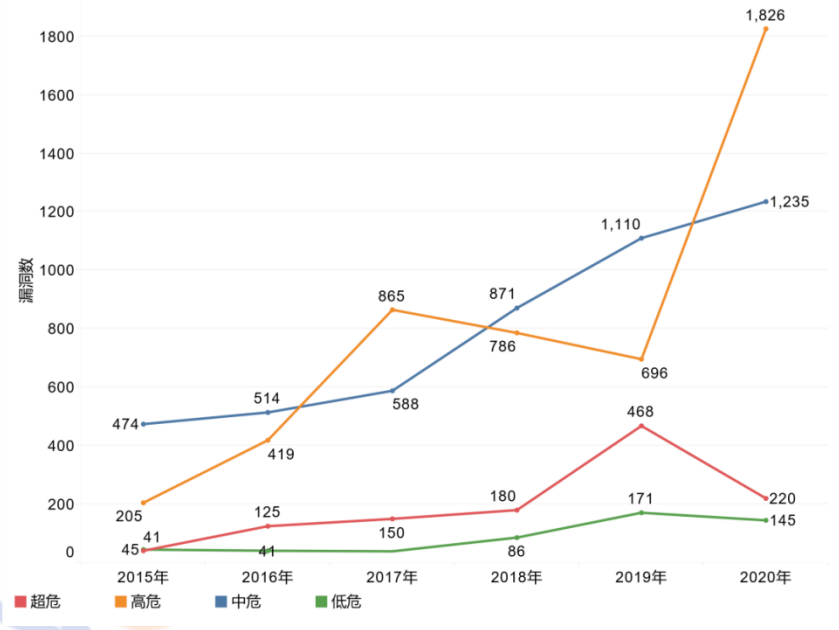

图 9　近 6 年新增漏洞风险等级时间分布

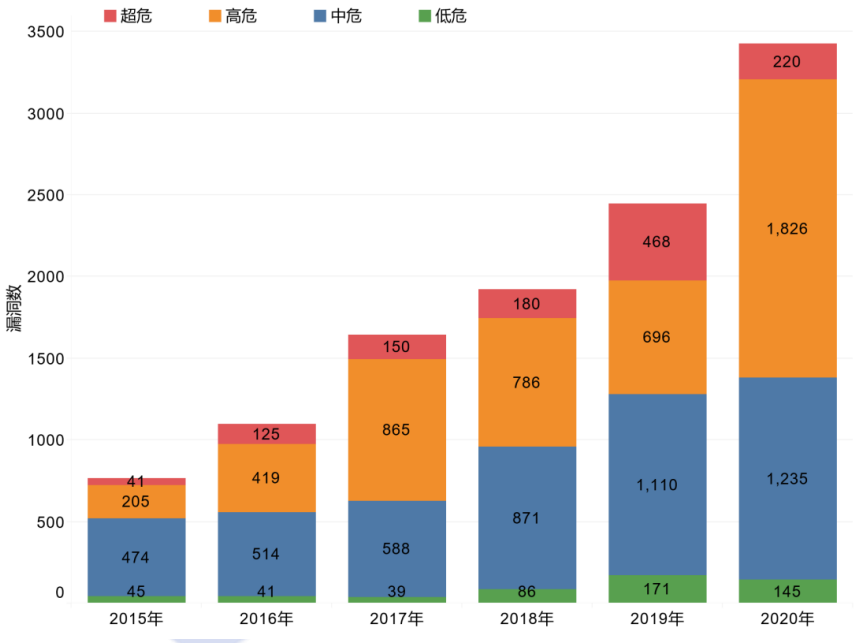

图 10　近 6 年新增漏洞各风险等级占比

发现十：2020 年，含高危以上漏洞占比最多仓库是 Rubygems

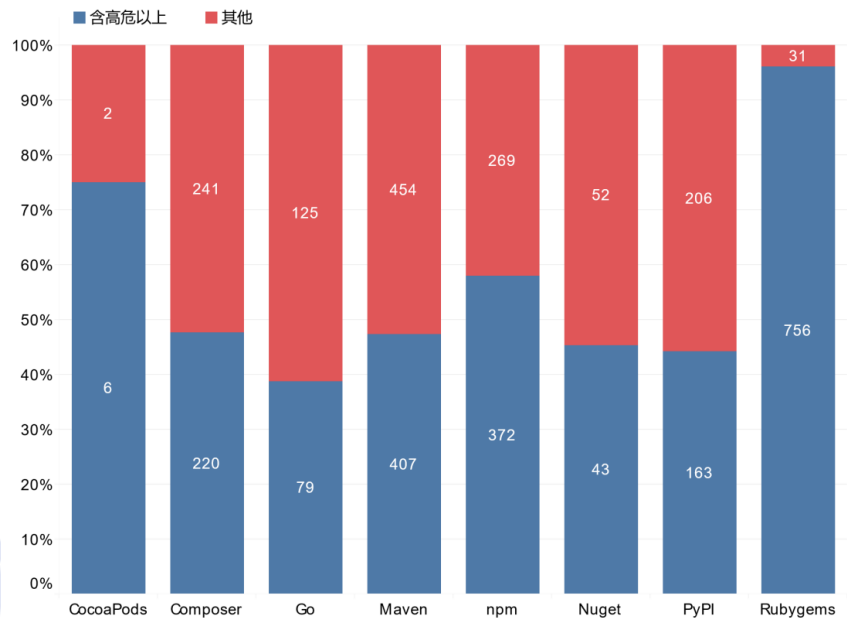

图 11　2020 年各仓库中含高危以上漏洞占比

调查结果显示，超八成组件含高危以上漏洞占比均超过 40%。2020 年, Rubygems 仓库含高危以上漏洞占比最多，占 Rubygems 仓库新增漏洞的 96%；Go 仓库含高危以上漏洞占比最少，占 2020 年 Go 仓库新增漏洞的 39%。

发现十一：平均每版本漏洞最多的 TOP 25 组件约五成来自 Composer 仓库

针对 2020 年各仓库新增漏洞，分析得到平均每版本漏洞数量最多的 TOP 25 组件。考虑到各仓库不同组件的版本数量各不相同，本报告采取平均每版本漏洞数作为计算依据，即：平均每版本漏洞数=组件全版本漏洞数/组件版本数。

研究发现，平均版本漏洞最多的 TOP25 中，Composer 仓库的组件数占比最多，共计 12 个，占比约 5 成左右；PyPI 仓库的组件数排名第二，共计 7 个；平均版本漏洞数最多的组件来自 Maven 仓库，漏洞数量为 47 个，下图列示各仓库中平均版本漏洞 TOP25 组件分布：

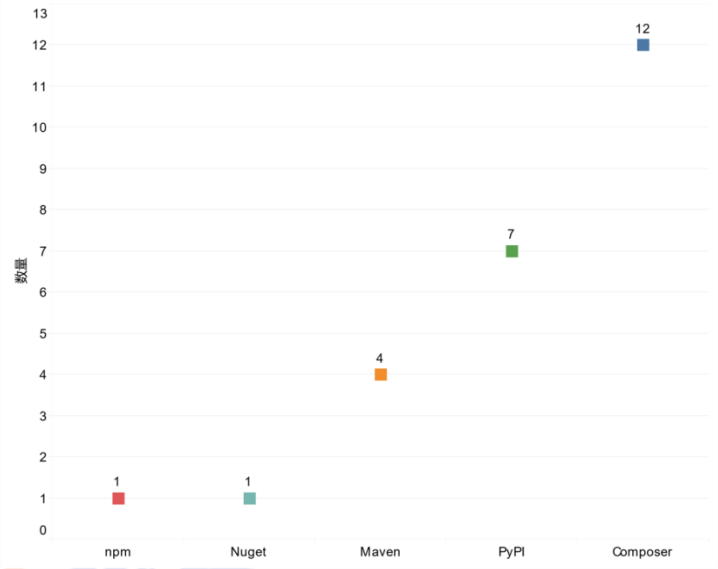

图 12 平均版本漏洞最多 TOP25 组件仓库分布

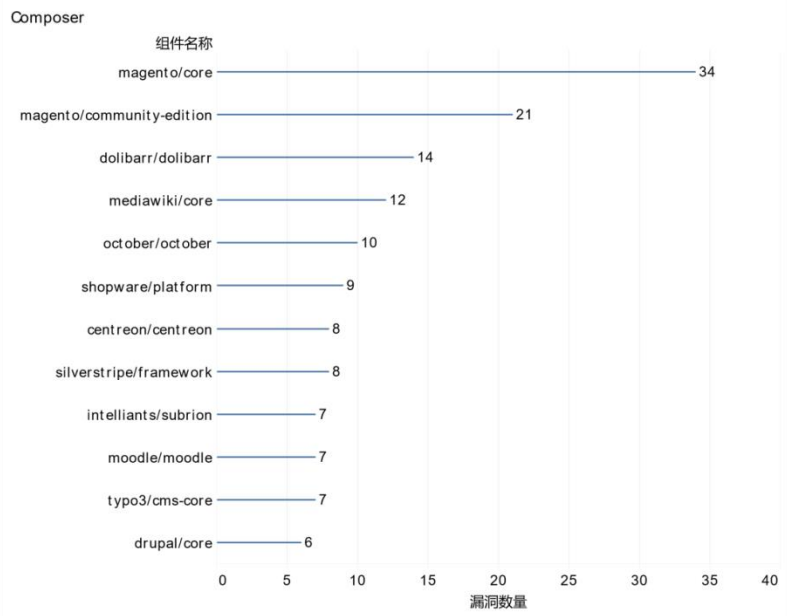

图 13 Composer 仓库组件分布

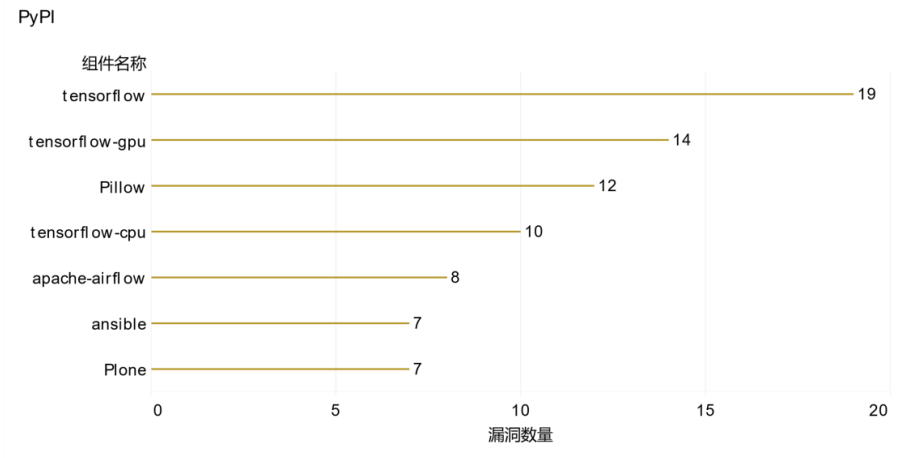

图 14 PyPI 仓库组件分布

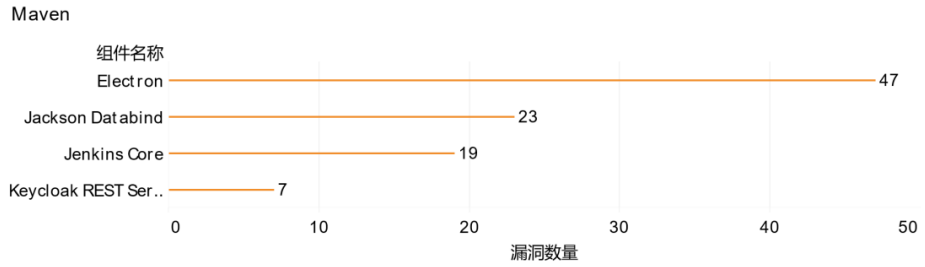

图 15 Maven 仓库组件分布

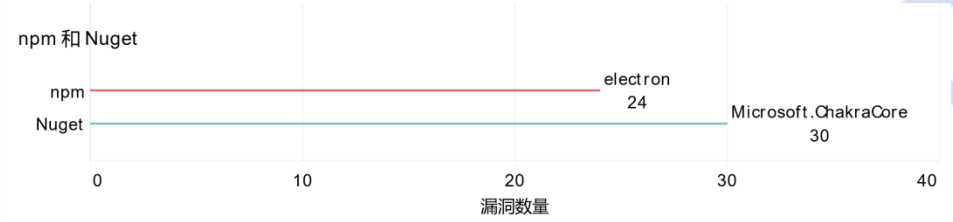

图 16 npm 和 Nuget 仓库组件分布

三、组件漏洞依赖层级传播范围分析

软件工程中经常引用组件来实现某些功能，组件之间存在相互依赖关系，按依赖关系可分为直接依赖和间接依赖，即组件 A 依赖组件 B，组件 B 依赖组件 C，那么组件 A→组件 B 和组件 B→组件 C 的依赖关系称为直接依赖，而组件 A→组件 C 的依赖关系称为间接依赖。组件存在安全漏洞，组件之间又存在相互依赖关系，导致漏洞在组件之间存在传播风险。

本报告以 Maven、npm、Rubygems 、PyPI、Composer、Nuget 6 个仓库中含已公开安全漏洞的开源组件为研究对象，共计 6,416 个[12]，对其做两轮漏洞传播模拟实验，研究开源组件漏洞依赖层级传播范围。第一轮实验，查找直接依赖这 6,416 个开源组件的组件集合，为方便定义，称该组件范围为一级传播；第二轮实验，查找直接依赖一级传播组件的组件集合，该组件集合均间接依赖这 6,416 个开源组件，称该组件范围为二级传播。

发现十二：一级传播影响范围扩大 125 倍，二级传播影响范围扩大 173 倍

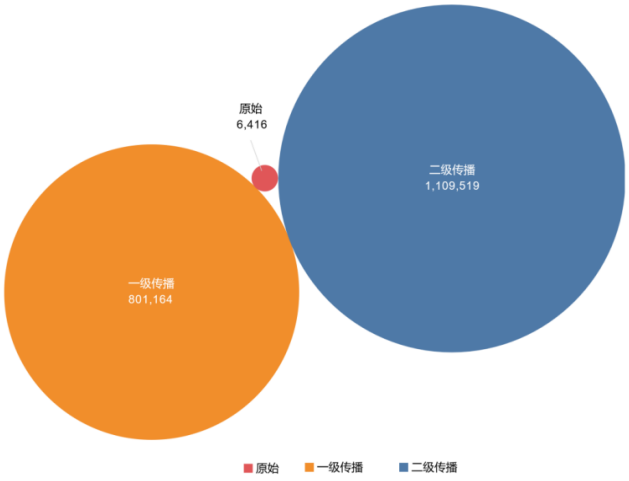

图 17 组件漏洞依赖层级传播范围

[12] 截至 2020 年 12 月，Maven、npm、Rubygems 、PyPI、Composer、Nuget 6 个仓库中含已公开安全漏洞的开源组件共计 6,416 个。

调查结果显示，原始样本中 6,416 个组件，受组件依赖关系的影响，一级传播一共波及 801,164 个组件，其影响范围扩大 125 倍。第二轮实验中，发现二级传播一共波及 1,109,519 个组件，影响范围相比原始样本 6,416 个组件扩大 173 倍。

发现十三：npm 仓库中的组件经 2 轮传播，影响组件数量最多

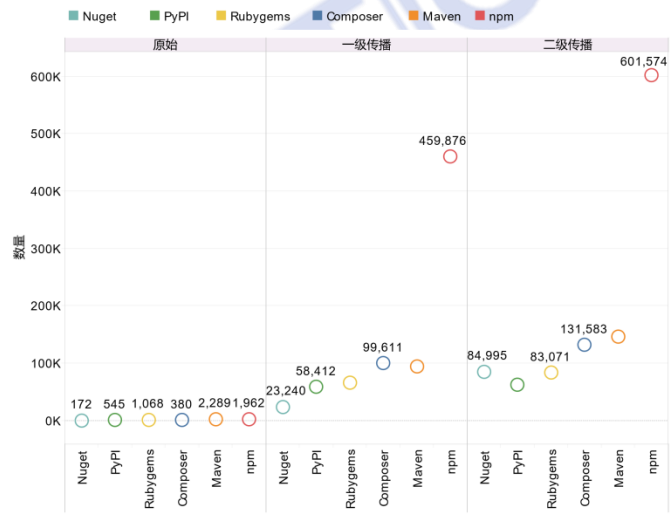

图 18 各仓库组件漏洞传播范围

调查结果显示，Maven、npm、Rubygems 、PyPI、Composer、Nuget 6 个仓库选取样本中，npm 仓库原始含漏洞组件数量为 1,962 个，npm 仓库漏洞组件数量在原始样本中仅次于 Maven 仓库，排在第二位。经 2 轮模拟传播实验，发现 6 组仓库中波及范围最广是 npm 仓库。npm 仓库原始样本中共有 1,962 个含有漏洞的组件，经过一级传播共波及 459,876 个组件，漏洞的影响范围扩大了 234 倍；二级传播共波及 601,574 个组件，范围比最初 1,962 个组件扩大了 307 倍。

发现十四：一级传播影响范围最广的仓库是 Composer

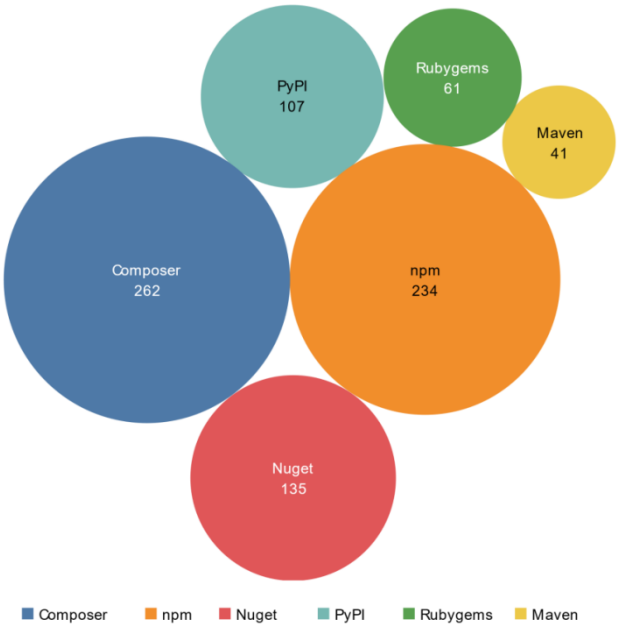

图 19 各仓库一级传播影响范围

调查结果显示，Composer 仓库原始含漏洞组件数量为 380 个，为 6 个仓库中原始样本中含漏洞组件数量的第 5 位。经 1 次传播，一级传播波影响范围最广的仓库是 Composer。经过一级传播共波及 99,611 个组件，漏洞的影响范围扩大了 262 倍。

发现十五：二级传播影响范围最广的仓库是 Nuget

调查结果显示，Nuget 仓库原始含漏洞组件数量为 172 个，为 6 组中含漏洞组件数量最少的仓库。经 2 次传播，二级传播波影响范围最广的仓库是 Nuget。经过一级传播共波及 23,240 个组件，漏洞的影响范围扩大了 135 倍；二级传播共波及 84,995 个组件，范围比最初 172 个组件扩大了 494 倍。

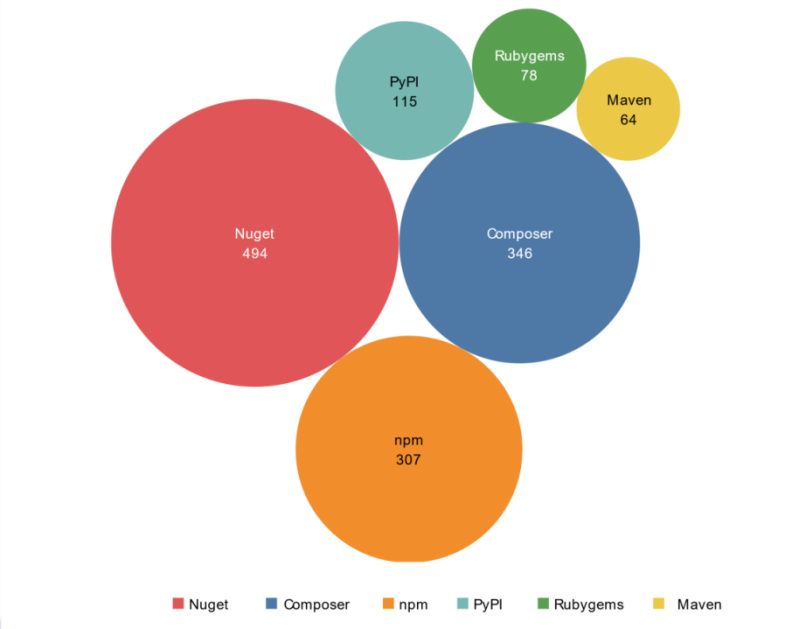

图 20 各仓库二级传播影响范围

发现十六：传播影响范围最小的仓库是 Maven

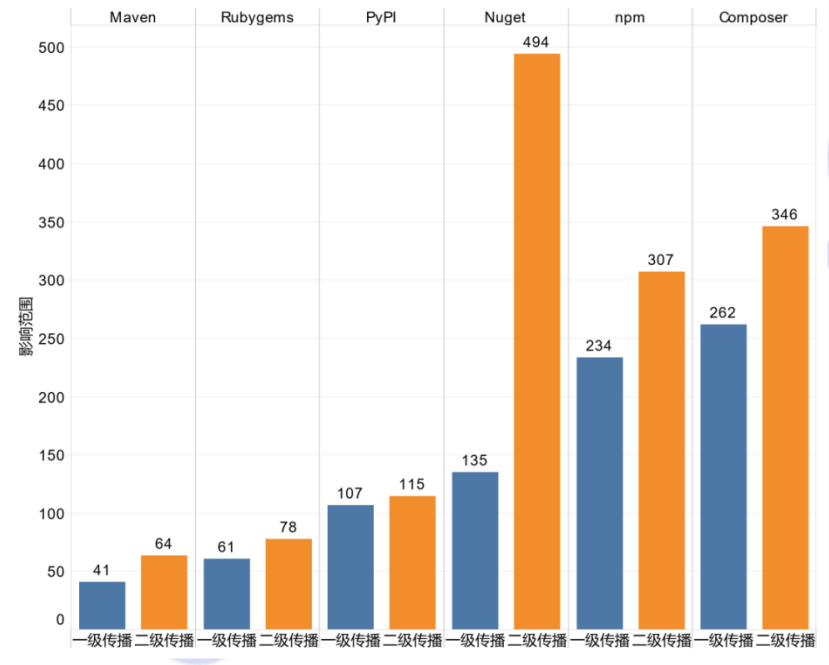

图 21 两轮漏洞传播组件漏洞影响范围分布图

调查结果显示，Maven 仓库原始含漏洞组件数量为 2,289 个，经过 2 次传播，6 组仓库中受漏洞影响范围最小是 Maven 仓库。经过一

级传播共波及 94,724 个组件,漏洞的影响范围扩大了 41 倍；二级传播共波及 145,827 个组件，范围比最初 2,289 个组件扩大了 64 倍。

从整体上看，开源组件生态中漏洞影响范围远超预期，组件间的依赖层级关系会导致组件之间漏洞存在传播风险。因此，要保证软件的安全风险控制，应通过自动化的手段识别软件工程中的组件成分，梳理组件间的依赖关系；在已知成分清单基础上对组件漏洞风险实施管控；同时，还要对已知成分进行动态监控，建立组件生态的漏洞威胁警报，在动态变化中将安全漏洞风险降到最低。

四、开源文件潜在漏洞风险传播分析

开源项目中往往存在相互引用关系，同一开源文件可能被多个项目所引用或包含。考虑到开源文件这一特性，本研究选取已公开漏洞中定位至文件级的开源文件为研究对象，共计 17,570 个，依托本研究团队的开源项目知识库[13]，对 17,570 个漏洞文件进行同源分析，识别开源生态中包含此漏洞文件的开源项目范围。

发现十七：超 80%漏洞文件在开源项目具有同源文件

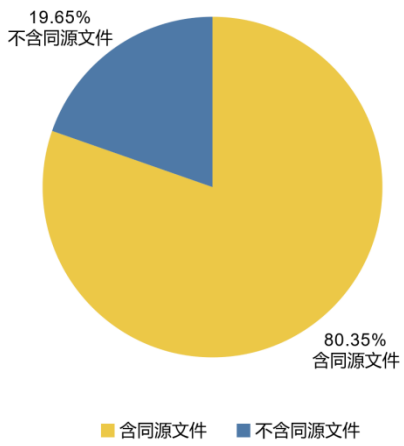

图 22 漏洞文件同源占比分布

调查结果显示，选取的 17,570 个含漏洞的开源文件中有 80.35%的文件可在开源项目中找到同源文件，共计 14,118 个，其余的 3,452 个文件未找到同源文件。

[13] 开源项目知识库是指对已公开的开源项目进行收录、清洗、整理形成的库集合。收录渠道包含：开源代码托管平台 GitHub（https://github.com）、开源代码托管平台 sourceforge（https://www.sourceforge.net）、开源代码托管平台 bitbucket（https://bitbucket.org）、开源代码托管平台 gitee(https://www.gitee.com)等各大开源社区

发现十八：漏洞文件在开源项目中传播范围扩大 54 倍

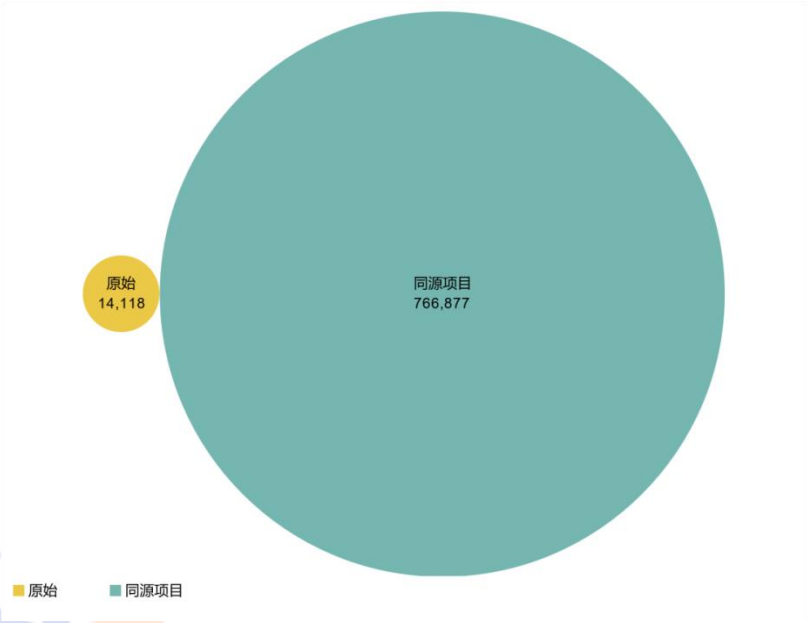

图 23 漏洞文件在开源项目中传播范围分布

通过对含同源文件的 14,118 个含有漏洞的开源文件进行分析，在不考虑同一开源项目不同版本的前提下，这些漏洞文件被 766,877 个开源项目所引用，漏洞文件在开源项目中传播范围扩大 54 倍。如果考虑同一开源项目的不同版本，这些漏洞文件被 2,410,476 个开源项目所引用，漏洞文件在开源项目中传播范围将扩大 171 倍。

案例分析

LibTIFF 项目中 tif_next.c 被曝出有 2 个中危漏洞 CVE-2015-1547 和 CVE-2015-8784。为了解该漏洞文件在开源项目中的引用情况，本研究团队对 tif_next.c 漏洞文件进行同源分析，在开源项目知识库中共发现有 237 个开源项目包含 tif_next.c 文件，如果考虑项目不同版本，共有 1000 个开源项目中引用 tif_next.c 文件。表 2 列出了 6 大托管平台引用 tif_next.c 文件的部分同源开源项目，经对比，所有同源文件均仅删减了注释行，而代码逻辑以及函数变量名称均未改变。

表 2 6 大托管平台 tif_next.c 文件的同源开源项目举例

托管平台	开源项目名称	版本号	同源文件路径
GitHub	reactos/reactos[14]	backups/ros-branch-0_4_2@73087	见注释[15]
Gitee	mirrors-opencv[16]	2.4.10	见注释[17]
Gitlab	limbo[18]	v2.2.1-Limbo-armv7-hf	见注释[19]
Bitbucket	xray[20]	FirstAddedTBB	见注释[21]
Sourceforge	wxhaskell[22]	wxInstall-Abriline-32-0.1	见注释[23]
CodePlex	casaengine[24]	master	见注释[25]

从整体上看，本次研究发现相同的文件被多个开源项目所引用的现象远多于预期。考虑到漏洞利用的复杂性，本研究团队认为这些结构一致的同源文件具有潜在漏洞风险，漏洞是否能真正的被利用，还需要深入研究。

[14] 项目下载地址：https://github.com/reactos/reactos/tags?after=ReactOS-0.4.5

[15] reactos-backups-ros-branch-0_4_2-73087\reactos-backups-ros-branch-0_4_2-73087\reactos\dll\3rdparty\libtiff

[16] 项目下载地址：https://gitee.com/mirrors/opencv/tags?page=4

[17] gitee-mirrors-opencv-2.4.10\opencv\3rdparty\libtiff

[18] 项目下载地址：https://gitlab.com/bob447008888/limbo/-/tags?page=2&sort=updated_desc

[19] gitlab-limbo-v2.2.1-Limbo-armv7-hf\limbo-v2.2.1-Limbo-armv7-hf\jni\SDL_image\external\tiff-4.0.3\libtiff

[20] 项目下载地址：https://bitbucket.org/sentike/xray/downloads/?tab=tags

[21] sentike-xray-a2c911aa2e5b\sentike-xray-a2c911aa2e5b\3rd party\FreeImage\FreeImage\Source\LibTIFF4

[22] 项目下载地址：https://sourceforge.net/projects/wxhaskell/

[23] wxInstall-Abriline-32-0.1\wxInstall-Abriline-32-0.1\wxWidgets\src\tiff\libtiff

[24] 项目下载地址：https://archive.codeplex.com/?p=casaengine

[25] casaengine\sourceCode\casaengine\external\FreeImage\Source\LibTIFF4

五、开源安全风险建议

开源生态带来的正面效应已在信息经济生活中发挥重要影响，如何在安全可控的情况下使用开源，已成为开源生态的关键任务。开源安全风险防范措施应贯穿软件开发的整个生命周期。本报告给出了如下六点建议:

（一）建立开源管理领导组织。随着软件开发过程中开源软件的使用越来越多，开源软件事实上已经成为了软件开发的核心基础设施。开源软件由于其特殊性，从软件供应链视角看，将横跨采购、选型设计、研发编码、运维交付等多个环节，需要跨组织、跨部门协同管理。具备条件的企业、机构建议设立开源管理办公室或领导小组，全面学习开源生态知识，了解开源生态运作机制，开展开源风险意识培训，强化开源风险管控手段。

（二）识别开源成分。开源软件全面渗透至软件供应链体系中，需准确识别软件中的开源成分（包括开源源代码成分、开源二进制成分、引用依赖的开源组件成分等），形成开源成分清单和图谱，做到对软件开源成分的可知可控。同时精确绘制开源组件依赖链条，防止安全风险随着开源组件依赖链条的逐层传播，是进行开源安全风险防范的基础。

（三）修复已知开源漏洞。已知软件的开源成分清单和图谱，明确开源成分及依赖链条后，通过相关工具检测或知识库查询，可有效识别已知开源漏洞。结合项目实际情况进行漏洞修复，降低软件安全风险；对于未修复或短期内无法修复的漏洞，确定安全风险接受清单，明确相关应急响应机制，最大程度降低风险。已知的开源漏洞修复，相较于代码安全审计，往往是将带有漏洞的组件升级至最新或较安全的组件版本，操作简单，易于实施，是一个投入产出比较高的风控措施。

（四）建立开源威胁情报体系。软件安全是动态发展的，开源软

件的威胁情报由不同的组织（如开源社区、开源基金会、开源项目方）共同贡献分享。开源威胁情报呈现无统一组织，散落在互联网海量信息中。需建立对已知开源成分及依赖链条漏洞威胁情报的实时监控跟踪机制，搜集多维度多渠道的开源威胁情报，并针对企业、机构已有的开源成分清单和图谱，在第一时间内将有效的开源威胁情报同步给相关责任人。

（五）弃用过老的开源组件和版本，及时安全更新。较老的组件往往存在大量的安全漏洞，也存在软件供应链安全隐患（如停止运维更新升级）。在使用开源软件，应关注其热门程度、受欢迎趋势、社区口碑、更新时效性等因素。应选择较新的安全的开源组件，并做到及时安全更新，降低开源安全风险。

（六）使用左移安全工具。在测试或者交付验收环节，针对开源安全风险进行处置往往会投入更大的工作量和成本。通过使用左移开源安全治理工具的方式，将开源安全风险治理集成到软件开发全生命周期过程中。在需求设计、组件选型、编码集成等早期研发过程中及早发现开源风险问题，从而降低开源安全治理成本。