构建安全的软件供应链

——Javascript 环境下的 SCA 工具研发项目结题报告

毛懿诺 集成电路与微纳电子创新学院
罗康灵 计算与智能创新学院
沈高畅 计算与智能创新学院
指导老师 朱东来

摘要

随着开源软件的广泛应用，软件供应链安全风险日益凸显。软件成分分析（SCA）作为识别和管理开源组件风险的关键技术，其重要性不言而喻。本项目旨在研发一款具备核心功能的 SCA 工具，以帮助开发者和企业快速发现项目中的开源组件及其安全漏洞。

本项目基于给定离线漏洞库进行开发。工具的核心实现主要包括三个模块：首先，通过依赖关系解析，对目标软件进行成分扫描；其次，根据找到的依赖文件和构建出的依赖树进行漏洞匹配；最后，生成 SBOM 文件，输出漏洞匹配结果。

目前，本工具已成功实现对 js 项目的自动化分析。测试结果表明，工具能够准确识别出项目引用的开源组件及其版本信息，并对其中含有漏洞的项目实现了有效检出。本项目的成果为软件供应链安全提供了一种轻量级、自动化的解决方案，具备一定的实用价值和推广前景。

关键词

软件成分分析（SCA）；软件供应链安全；开源组件；漏洞管理；自动化工具

引言

在数字化转型浪潮中，软件已成为企业核心资产，但开源组件的泛滥使用带来了前所未有的安全挑战。根据 Synopsys 报告，96%的商业软件包含开源组件，其中 85%存在已知漏洞。软件成分分析(SCA)正从可选动作变为必选项，其核心价值在于让看不见的风险"可视化"。近年来，Log4j、Spring4Shell 等重大漏洞频发，平均每个应用包含 158 个开源漏洞，SCA 能自动化识别项目中所有第三方组件及其关联漏洞，相比人工审查效率提升 300%以上。另外，2023 年供应链攻击同比增长 650%，攻击者常通过篡改开源组件实施入侵。SCA 能检测组件是否被恶意修改，识别非预期依赖关系，阻断攻击链条。即使没有安全漏洞问题，开发团队平均 30%时间耗费在解决依赖冲突上，SCA 自动生成的依赖关系图谱能显著降低维护成本。在这样的背景下，金融行业《开源软件应用风险管理指引》明确要求建立 SBOM 清单，工信部《网络安全产业高质量发展三年行动计划》也着重强调软件供应链安全。SCA 已经成为贯穿软件研发全生命周期的重要工具。

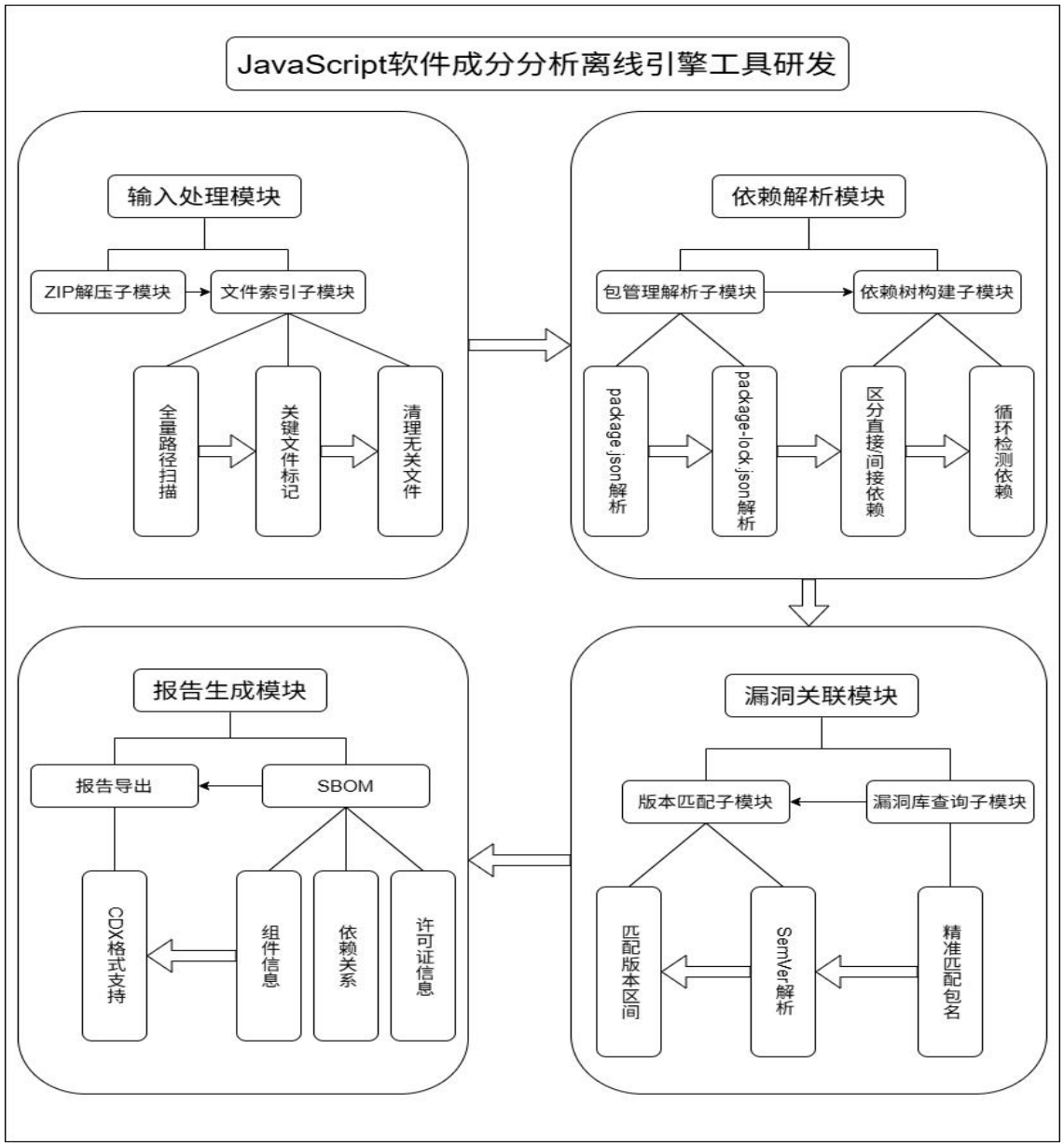
正文

工作流程

1. 进行市场调研，主要是确认各类 js 项目的依赖文件存储路径也即项目结构、依赖包命名规则、依赖描述方法等。
2. 绘制模块图，确认各成员的分工，确定每项工作的预定截止日期

3. 着手研发工作，互相交流各自需要输入和输出的数据，保证团队合作的顺畅
4. 所有人员完成工作后合并成果，形成可运行的项目，进行测试
5. 针对测试中发现的问题不断修改
6. 确认最终成果，结项

项目结构



1-项目模块图

本项目一共分为四个模块，分别为输入处理模块、依赖解析模块、漏洞关联模块和报告生成模块。

第一步，输入处理模块。由于我们所获得的项目文件是一个压缩包，在处理之前必然需要先将其解压缩，这一部分由 ZIP 解压子模块来实现。考虑到使用者

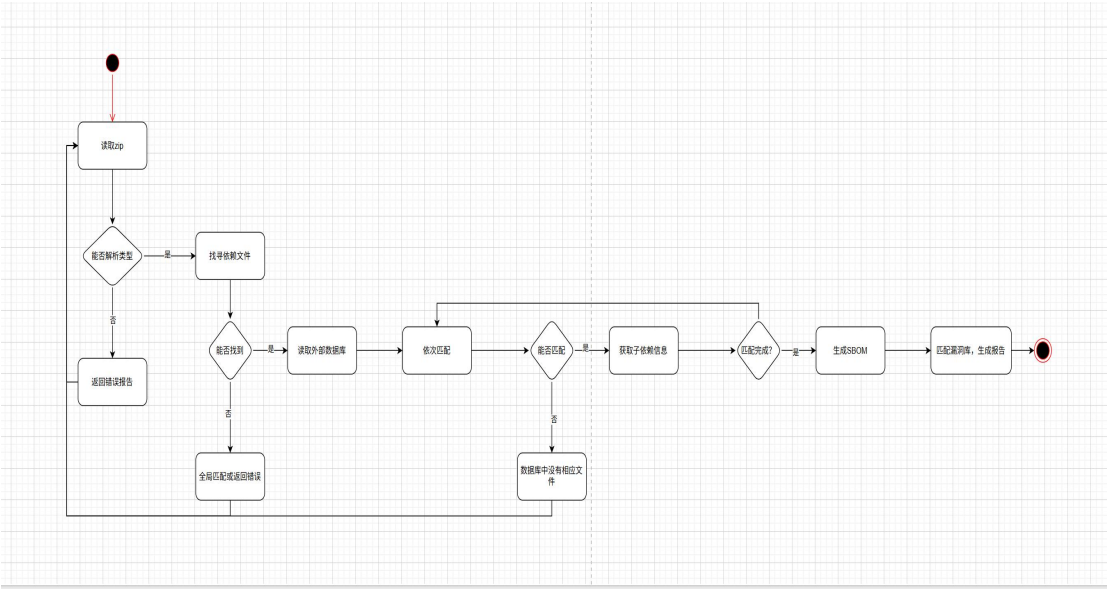
不一定会给出压缩包的名字，我们设置了查找文件夹和直接指定文件两种寻找项目的方式。而后，在文件索引子模块中，我们实现了从解压后的压缩包中找出我们所需的依赖文件，即 package_lock.json 或者 package.json，并把它们存放到指定路径下，这两个文件是下一步依赖树解析中的必需品，有了它们，依赖解析模块才可以正常工作。

第二步，依赖解析模块。绝大部分的项目中含有 package_lock.json 文件，也称锁文件。这个文件中详细地列出了该项目所有的直接依赖和间接依赖，可以很方便的构建我们所需要的依赖树。这一步共分两个子模块，包括包管理解析子模块和依赖树构建子模块，前者解析项目含有的两个依赖文件，后者根据前者提取出的信息，整理出项目的依赖路径。由于一个依赖本身可能含有别的依赖，因此在构建依赖树的过程中，我们需要不断地循环检测，逐步将项目解析为最小的部分，生成一棵完整的依赖树。

第三步，漏洞关联模块。这个模块是寻找项目漏洞的核心，也是唯一一步需要用到外部数据库的地方。根据需求，我们开发的 SCA 工具应当是一款离线的软件成分分析工具，这样才能够匹配企业生产环境的需求。因此，这一步中我们会关联到内部网络上的一个漏洞数据库，精准查询我们解析出来的依赖路径中的所有项目是否存在漏洞库中已知的漏洞，这里必须小心的是版本号的问题，一个相同的开源项目可能较老的版本有漏洞并且在下一个版本中修复，也有可能原先没有漏洞，但在更新之后出现问题，且一个项目中可能多次使用到了不同版本的该开源项目，因此必须非常准确的进行版本号匹配。另外，由于管理机制的不同，相同的开源项目在项目依赖文件中和漏洞库中可能存在着不同的前后缀，这一点必须通过完整的调研来确定究竟怎样的名称格式可以认定为两者是同一个开源项目的不同称呼。

第四步，报告生成模块。这一步相对简单，主要是输出我们所找到的漏洞信息，生成所需要的 SBOM 文件。生成文件已经拥有了相对成熟完整的开源代码，无需很大改动便可以直接使用。

工具工作流程



2-SCA 工具工作流程图

本 SCA 工具的核心工作是一个自动化、多阶段的数据处理与分析流程。从用户发起请求开始，到最终生成详细的安全报告，整个流程可分为五个核心阶段：目标获取、成分解析、漏洞匹配、风险评估与报告生成。

阶段一：目标获取与预处理

流程的起点是用户提交待扫描的目标。如前文所讲，用户可以选择直接告诉程序压缩包路径或所在文件夹，程序会自动寻找并识别该项目。这一阶段的目的是为后续分析准备一个统一、可访问的本地文件系统环境。

阶段二：成分识别与解析

这是整个流程的核心识别阶段。系统会深入预处理后的项目目录，全面识别软件成分。主要是通过遍历项目文件，寻找已知的依赖管理配置文件。通过解析这些文件，可以直接、准确地获取项目声明的直接依赖和传递依赖关系，形成详细的组件列表。

在此阶段结束后，系统将生成一个标准化的组件列表，其中每一项都包含了组件名称、版本号等关键信息，并为每个组件生成唯一的标识符（如 CPE 或 PURL），为下一步的漏洞查询做好准备。

阶段三：漏洞数据匹配

在此阶段，系统将上一阶段产出的组件列表与本地集成的漏洞数据库进行关联查询。查询的核心是使用组件的唯一标识符和版本号，在漏洞库中寻找所有匹配的已知安全漏洞。此阶段的输出是一个“组件-漏洞”的映射关系列表。

阶段四：结果整合与报告生成

这是流程的最终输出阶段。系统将前面所有阶段的分析结果（组件清单、漏洞列表、风险等级）进行整合与格式化，并生成易于用户理解和操作的可视化报告。

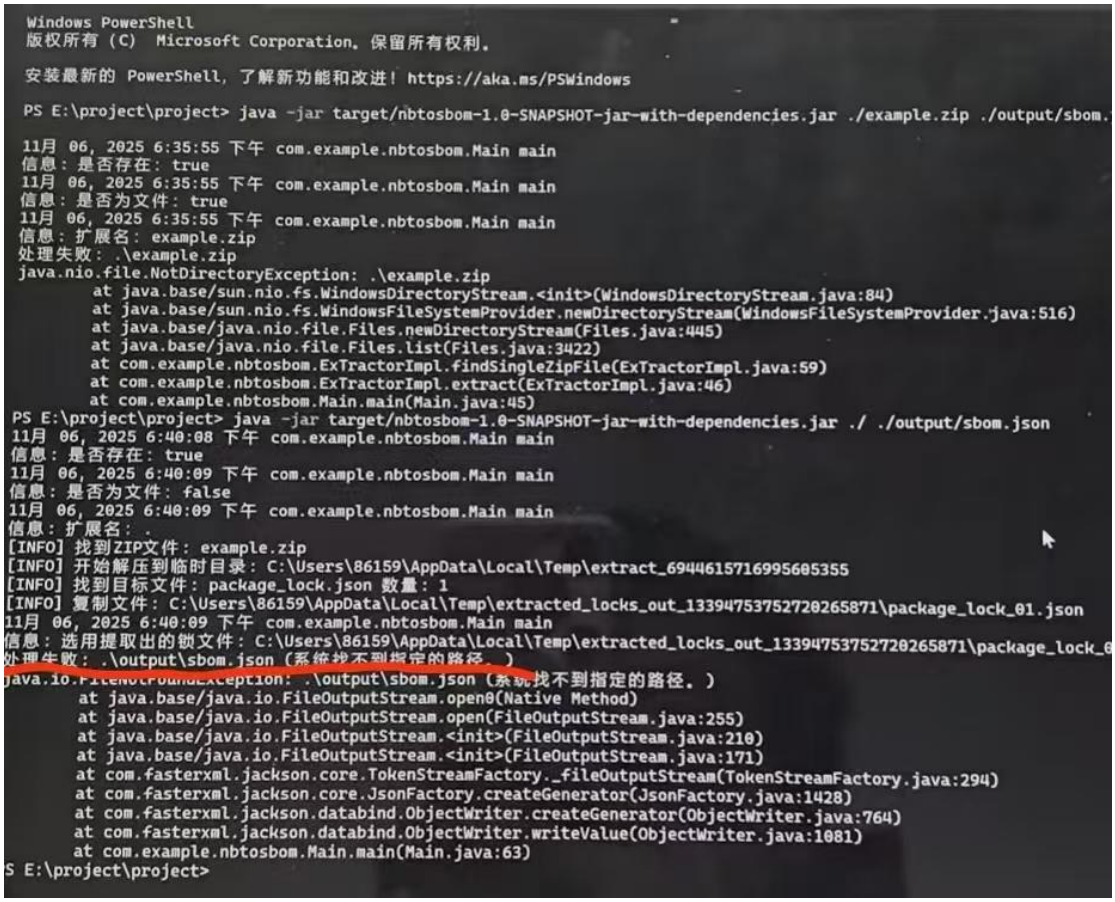
项目运行反馈

```
PS C:\sgc\szdk\pj\project> .\Runner.exe
11月 27, 2025 10:55:52 下午 com.example.nbtosbom.Main main
信息: 扩展名: ..
[INFO] 找到ZIP文件: example.zip
[INFO] 开始解压到临时目录: C:\Users\sgche\AppData\Local\Temp\extract_3398800603404706717
[INFO] 找到目标文件: package-lock.json 数量: 1
[INFO] 复制文件: C:\Users\sgche\AppData\Local\Temp\extracted_locks_out_5824769534807895885\package-lock_01.json
11月 27, 2025 10:55:52 下午 com.example.nbtosbom.Main main
信息: 选用提取出的锁文件: C:\Users\sgche\AppData\Local\Temp\extracted_locks_out_5824769534807895885\package-lock_01.json
SBOM物料清单 已生成: C:\sgc\szdk\pj\project\.\sbom.json
11月 27, 2025 10:55:53 下午 com.example.nbtosbom.SbomVulnEnricher queryVulnerabilitiesForPurl
信息: 当前数据库: jdbc:mysql://10.176.37.194/osschain_bachelor
11月 27, 2025 10:55:53 下午 com.example.nbtosbom.SbomVulnEnricher queryVulnerabilitiesForPurl
信息: 查询 purl: pkg:npm/busboy@0.3.1
11月 27, 2025 10:55:53 下午 com.example.nbtosbom.SbomVulnEnricher queryVulnerabilitiesForPurl
信息: 当前数据库: jdbc:mysql://10.176.37.194/osschain_bachelor
11月 27, 2025 10:55:53 下午 com.example.nbtosbom.SbomVulnEnricher queryVulnerabilitiesForPurl
信息: 查询 purl: pkg:npm/dicer@0.3.0
11月 27, 2025 10:55:53 下午 com.example.nbtosbom.SbomVulnEnricher queryVulnerabilitiesForPurl
信息: 当前数据库: jdbc:mysql://10.176.37.194/osschain_bachelor
11月 27, 2025 10:55:53 下午 com.example.nbtosbom.SbomVulnEnricher queryVulnerabilitiesForPurl
信息: 查询 purl: pkg:npm/express-fileupload@1.3.1
11月 27, 2025 10:55:53 下午 com.example.nbtosbom.SbomVulnEnricher queryVulnerabilitiesForPurl
信息: 当前数据库: jdbc:mysql://10.176.37.194/osschain_bachelor
11月 27, 2025 10:55:53 下午 com.example.nbtosbom.SbomVulnEnricher queryVulnerabilitiesForPurl
信息: 查询 purl: pkg:npm/streamsearch@0.1.2
带漏洞信息的 SBOM 文件 已生成: C:\sgc\szdk\pj\project\.\sbom-enrich.json
```

3-运行过程中反馈的日志

一个合格的工具即使出错，也应当让使用者知道它目前完成了什么工作、哪一步出了错、可能的原因是什么。如上图所示，我们研发的 SCA 项目在运行过程中会生成详细的日志文件，包括了寻找压缩包、解压过程、寻找依赖清单文件、数据库匹配过程、当前查询以及运行结果。这样，如果有一步出了错，比如文件

找不到（如下图）、漏洞库连接不上等，使用者就可以清楚的知道问题所在。



4-一种找不到文件的错误

总结与展望

工作总结

本项目围绕软件供应链安全的核心需求，成功设计并实现了一款具备核心功能的软件成分分析（SCA）工具原型。在整个研发周期中，我们系统性地完成了从技术选型、系统设计、编码实现到功能测试的全过程，主要工作成果可总结如下：

1.完成了系统性的技术选型与设计： 基于项目目标，我们设计了包含项目扫描、成分分析、漏洞匹配和报告生成四大核心模块的系统架构，确保了工具的可扩展性和可维护性。

2.实现了 SCA 核心工作流程： 工具成功实现了从“目标获取”到“报告生成”的自动化分析流水线。通过依赖清单解析的策略，能够精准地识别项目中的开源组件及其版本信息。

3.构建了本地漏洞知识库并实现了精准匹配：通过集成离线数据库作为核心数据源，我们开发了基于组件标识（CPE/PURL）和版本范围的匹配引擎，能够有效关联组件与已知的安全漏洞（CVE）。

4.开发了可视化报告功能： 工具能够将扫描结果生成清晰、直观的 HTML 格式报告，详细列出组件清单、漏洞详情（包括 CVSS 风险等级）、修复建议等，为用户提供了直接可行的安全洞察。

5.通过了系统化测试验证： 通过分别对多种含有漏洞、不含有漏洞、含有特

殊开源组件的项目包进行测试，验证了本工具在成分识别准确率、漏洞检出能力以及扫描性能方面均达到了预期目标，具备了一定的实用价值。

参考文献

[1] 陈春荣《基于软件成分分析的 Java 软件漏洞检测系统的设计与实现》，北京邮电大学，2025.5.16-6.15，2025 年第 06 期

后记

时光荏苒，当为这份结题报告画上最后一个句点时，也意味着我们为期数月的 SCA 工具研发项目即将正式落下帷幕。回首这段从零到一的探索之旅，心中充满了无限的感慨与感激。

作为一名大二的学生，这是我首次完整地参与一个具有明确工程目标和现实意义的研发项目。从最初面对“软件供应链安全”这个庞大概念的迷茫，到为了厘清一个技术细节而反复查阅文献，再到为了一个程序 Bug 而与队友挑灯夜战……这段经历带给我的远不止一份可运行的代码和一份完整的报告。它让我深刻地体会到，将课堂上学到的理论知识转化为解决实际问题的能力，是一个充满挑战却又无比迷人的过程。我们不仅学会了如何设计系统架构、如何集成第三方 API、如何进行有效的测试，更重要的是，我们学会了如何在困难面前协作共进、如何科学地管理项目周期。这份宝贵的实践经历，将是我大学生涯中一笔珍贵的财富。

在此，我谨以最诚挚的谢意，献给所有在此过程中给予我们支持和帮助的人。首先，我要衷心感谢我们的指导老师，朱东来老师。是您在我们选题迷茫时为我们指明了方向，在我们遇到技术瓶颈时给予我们关键的启发。您严谨的治学态度、深厚的专业学识以及不厌其烦的悉心指导，是支撑我们完成这个项目的最大动力。其次，感谢学校为我们规划了这样一个珍贵的实践机会，提供了良好的实验环境与丰富的学术资源，使得我们能够心无旁骛地投身于项目研发工作，极大程度地积累实践经验。最后，感谢项目组的每一位成员的智慧、汗水与包容。我们曾为某个技术方案激烈争论，也曾为每一次突破而共同欢呼。正是我们高效的协作与无私的奉献，才让一个个孤立的模块最终汇聚成一个完整的系统。能与诸位优秀的同伴并肩作战，是我莫大的荣幸。

由于我们的学识与经验尚浅，报告中难免存在疏漏与不足之处，恳请各位老师和专家批评指正。

谨以此文，为这段充满挑战与收获的旅程作结。