OWASP

Open Web Application Security Project

# 开源组件风险治理实践

开源网安 汪杰

# 目录 CONTENT

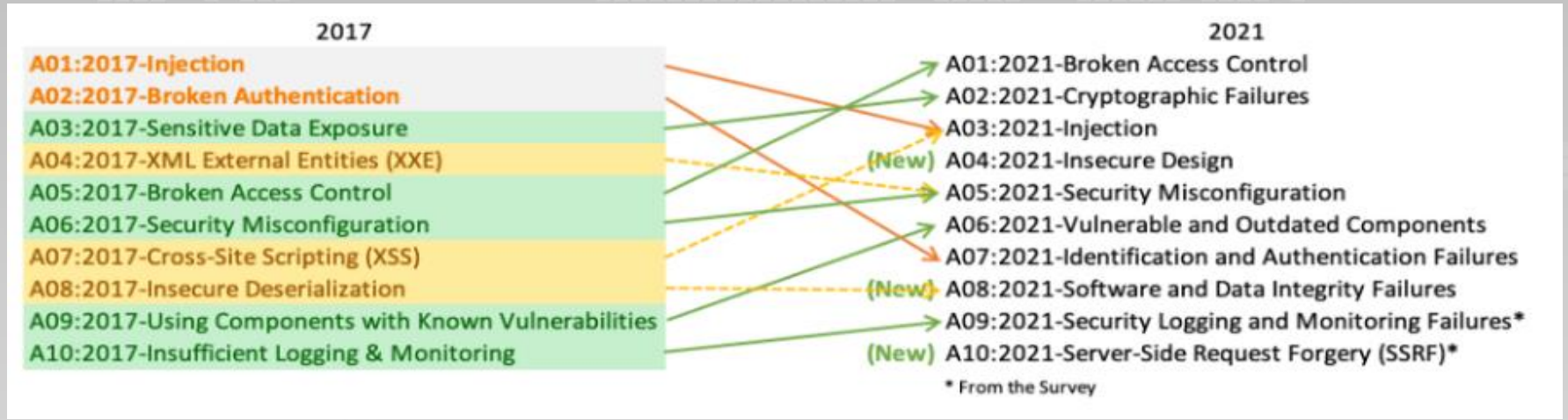
OWASP
Open Web Application
Security Project

# 开源组件的重要性

# 开源组件的重要性

**A06:2021 自带缺陷和过时的组件**

平均事件发生率：8.77%
总CVE数：0
社区推荐排名：2



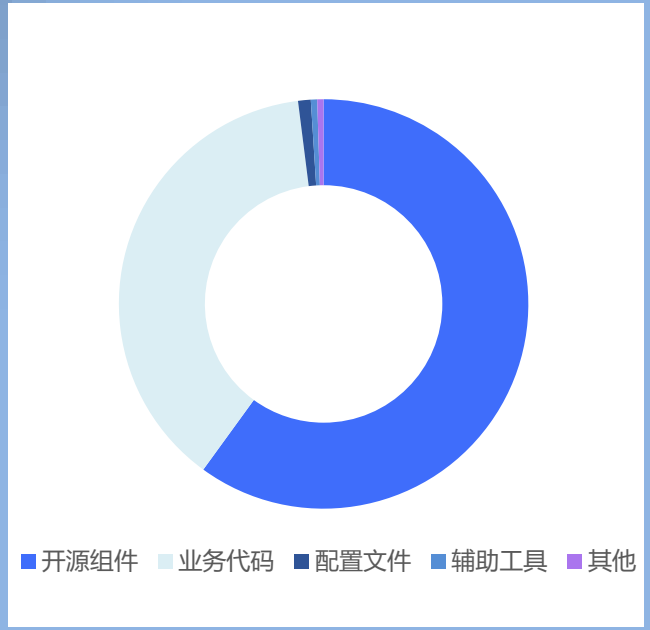

OWASP
Open Web Application Security Project


**下载量：开源组件被广泛使用**

- 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。

**需求量**：Sonatype 的《2020软件供应链报告》指出，到2020年，世界各地的开发者对开源组件和容器的需求，将超过1.5万亿个

**构成**：Gartner 表示，现代软件大多数是被“组装”出来的，不是被“开发”出来的。




**开源组件占60%**

据不完全统计，通常一个应用的开源组件占应用的60%～80%左右

**业务代码38%**

客户自己的业务代码约占整个项目的38%

**配置文件1%**

程序运行时相对应的配置文件约占1%左右

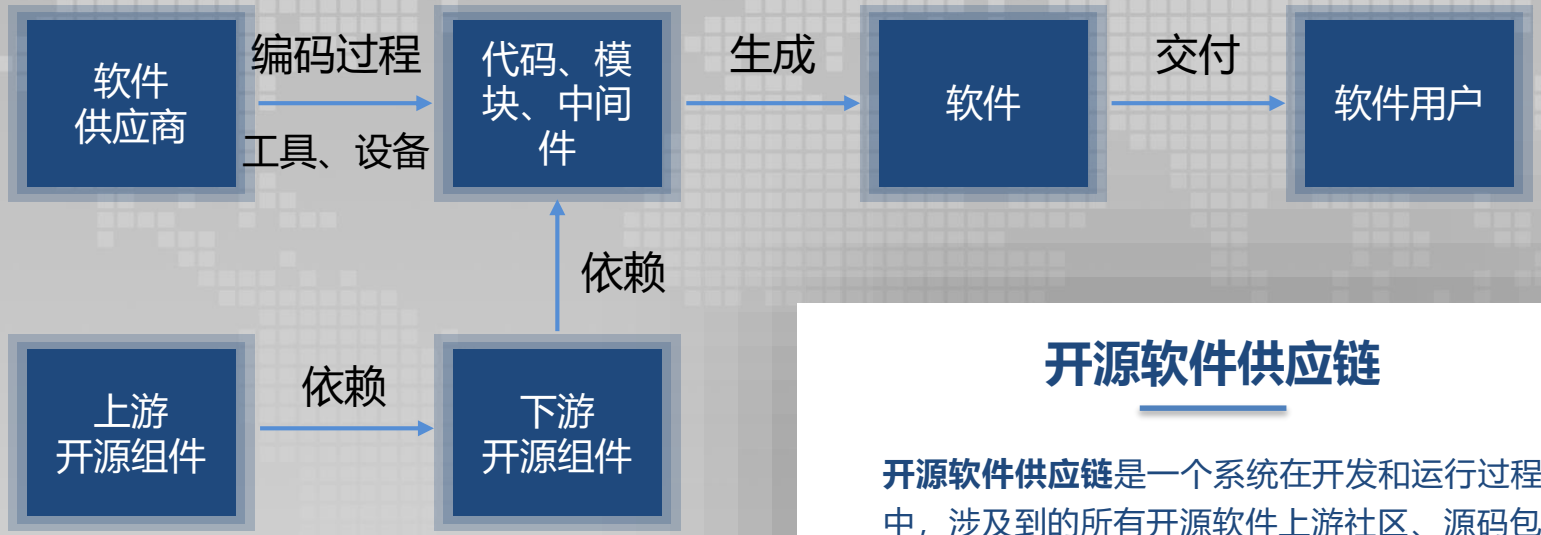
**构建脚本等辅助工具0.5%**

项目构建，打包等脚本程序和配置

**其他0.5%**

其他附加的程序资源

# 开源软件供应链




## 开源软件供应链

**开源软件供应链**是一个系统在开发和运行过程中，涉及到的所有开源软件上游社区、源码包、二进制包、包管理器、仓库等等,按照依赖、组合等形成的供应关系网络，**是开源组件和软件供应链的结合。**

OWASP

Open Web Application Security Project

# 开源组件的安全挑战

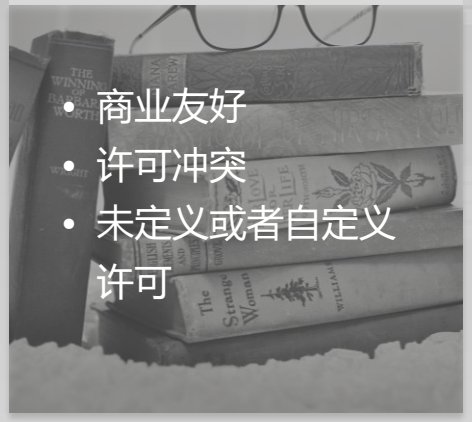
# 开源安全风险

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。

## 安全漏洞

- 来源分散CVE,NVD,CNNVD,CNVD,NPM,github,官网等

## 知识产权



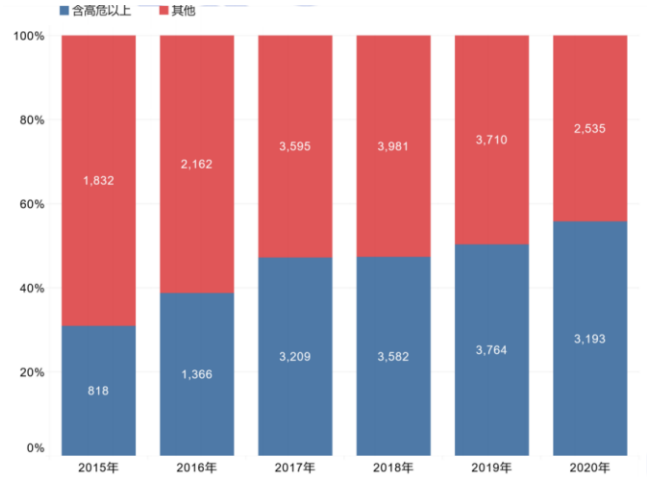
- 商业友好
- 许可冲突
- 未定义或者自定义许可

## 软件供应链安全

- 依赖传播，关系复杂
- 同源拷贝
- 上游维护性差

# 开源安全漏洞

## CVE官方未收录开源漏洞情况




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%

数据来源于：国家计算机网络应急技术处理协调中心《 2021 年开源软件供应链安全风险研究报告 》

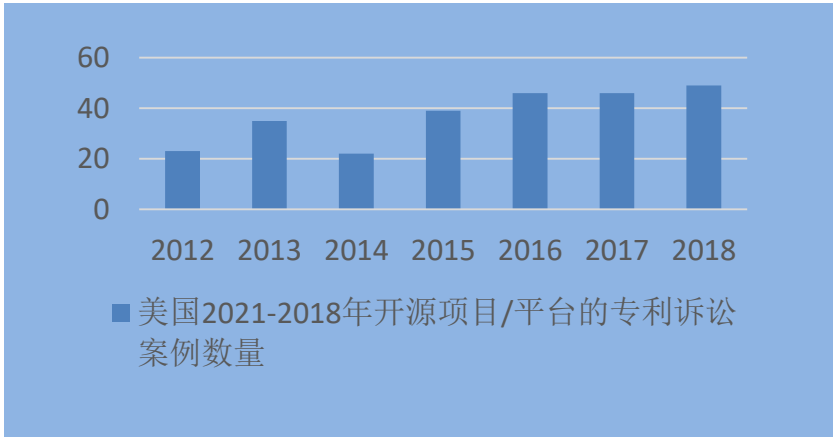
## 含高危以上漏洞占比

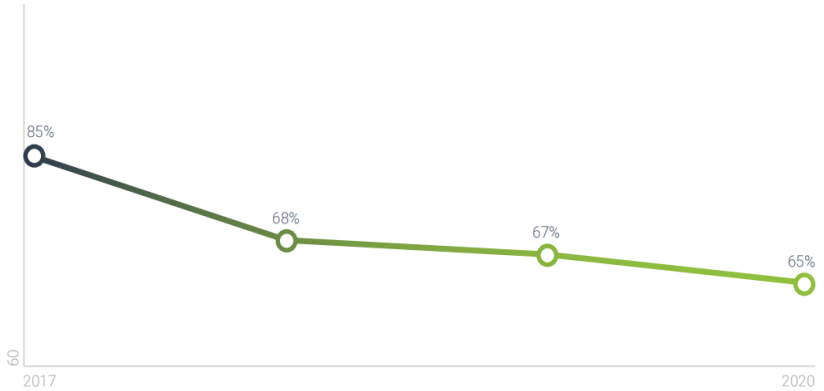

近4年，高危及以上开源漏洞占比均超40%，2020年，超危漏洞占比为8.83%，高危漏洞占比为46.91%，占2020年新增漏洞超5成

OWASP
Open Web Application Security Project

# 开源知识产权

- **许可冲突：**Black Duck审计服务团队发现，2020年的被审代码库中有**65%包含存在许可证冲突的开源代码**，比2019年略有减少。纵观存在许可证冲突的代码库，近四分之三与某个版本的“GNU通用公共许可证”存在冲突。
- **未定义或自定义许可：26%的被审代码库使用了没有许可证或定制许可证的开源代码。**使用定制开源代码许可证的代码库是否存在可能的IP和其他法律问题，需要评估后才能确定。




数据来源：Unified Patents ,2019年11月

# 开源软件供应链风险

**上游**

缺陷漏洞代码

恶意代码

**下游**

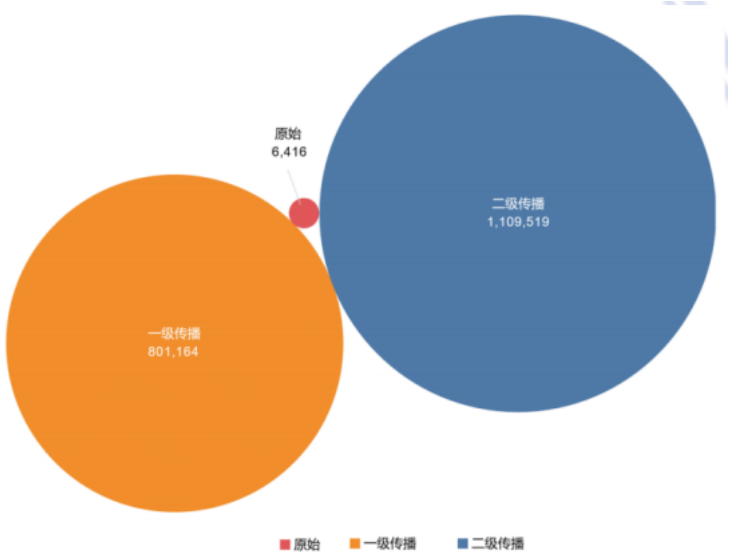
依赖引用

文件引用

**所谓上游**，是一个相对的概念，意为“靠近源头的一方”。在开源社区中，主要指的是开源社区维护的主干版本。而所谓下游，是指基于上游开源项目衍生出的项目或产品。在 Github、Gitee 等代码管理平台中对开源项目的 fork 操作，就是一种对上游代码的拓展。

# 开源软件供应链风险：开源依赖

## 组件依赖层级传播



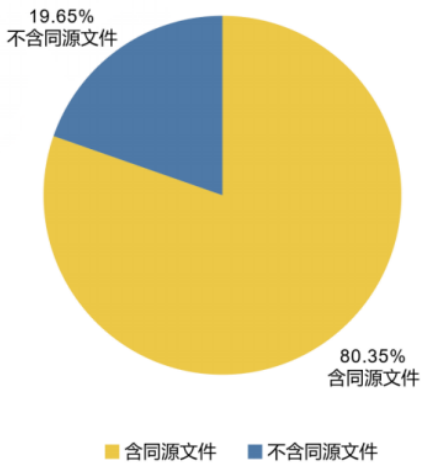

相关数据来源于：国家计算机网络应急技术处理协调中心《 2021 年开源软件供应链安全风险研究报告 》

- ✓ 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。

> 一级传播影响范围扩大**125**倍
> 二级传播影响范围扩大**173**倍

OWASP
Open Web Application Security Project

# 开源软件供应链风险：开源文件拷贝

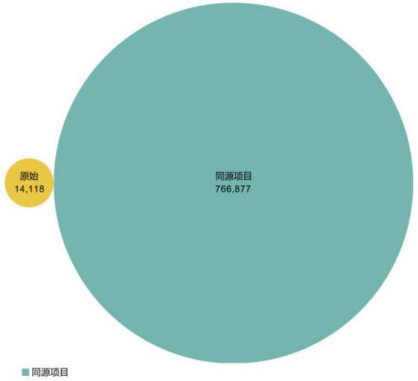



漏洞文件同源占比分布

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14,118个，其余的3,452个文件未找到同源文件。

相关数据来源于：国家计算机网络应急技术处理协调中心《 2021 年开源软件供应链安全风险研究报告 》

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。




漏洞文件在开源项目中传播范围分布

OWASP
Open Web Application
Security Project

# 开源软件供应链风险：开源项目维护性

开源项目活跃度低，维护能力不足等原因，造成供应链中开源组件缺乏维护

## 项目内部

**91%**

在Black Duck审计服务团队2020年审计的1,500多个代码库中，居然有91%使用了在过去两年中没有发生任何 开发活动的开源依赖项，**这意味着91%的被审代码库中包含在过去两年中没有进行过功能升级、代码优化和任何安全问题修复的依赖项。**

**85%**

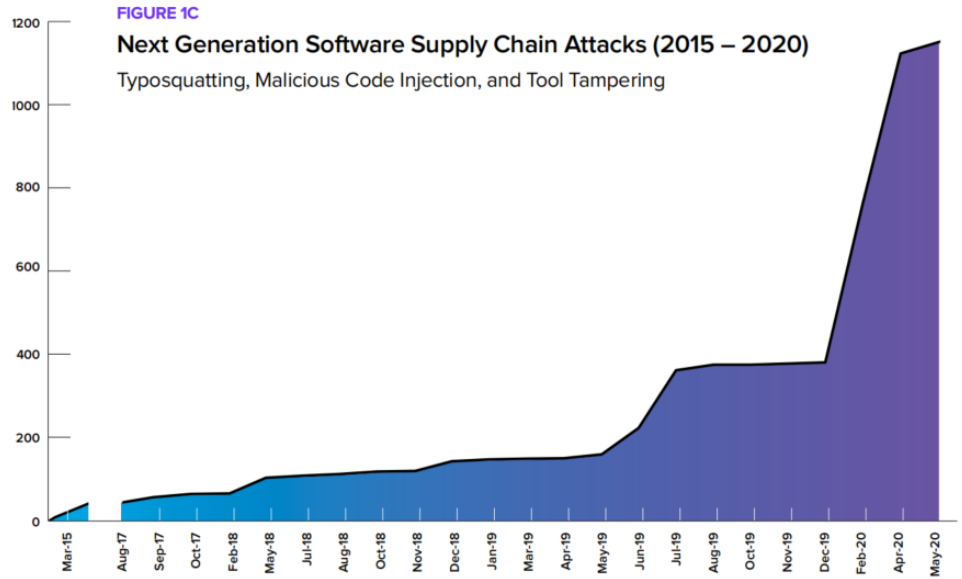
Black Duck审计服务团队2020年审计的代码库中，**85%的代码库含有至少四年未曾更新的开源依赖项。**也就是说，代码库使用的开源库并非最新版本，甚至经常是很旧的版本。如前所述，开发团队显然难以维护开源依赖项的时新性。

## 项目外部

JavaScript 的模块化标准库core-js ,这个库通过 npm 软件包管理器下载，每周居然高达2600万次。甚至连苹果公司的网络服务，也用到了这个库，但是由于作者Denis Pushkarev除了写程序外，平时爱好是飚摩托车，结果在一次事故中，他因驾驶摩托撞人,致一死一伤，被判处18个月有期徒刑。

另一个JavaScript加密库jsrsasign也遇到过类似的挑战。自2018年4月以来，项目就没有任何活动。但是在npm上却有350个项目完全依赖这个库，项目也受到了微软、Mozilla这样有影响力公司的青睐。

# 下一代软件供应链攻击




数据源于Sonatype 的《2020软件供应链报告》

攻击者不在被动的等待已知公开漏洞信息的披露，而是更加主动的攻击上游开源组件

- 报告指出，在过去的 12 个月中，其共记录了 929 次下一代软件供应链攻击。相比之下，2015 年 2 月至 2019 年 6 月之间记录的此类攻击则只有 216 起。**面向开源的下一代网络攻击增长 430%**

# 下一代软件供应链攻击：案例

2019.1

Pytz3-dev
这个pypi包的作者似乎复制了pytz包代码，然后添加恶意代码，该包每月下载47次

2019.1

smartsearchwp
2019.1发布，2020.6从npm移除，其中包括提供后门以支持数据外逃恶意代码

2019.3

Rubygems.org
Simple-captcha2-0.2.3
Datgrid-1.0.6
包含一个有第三方插入的代码执行后门

2019.4

Bootstrap-saas有人删除了bootstrap-sass v3.2.0.2 并立即发v3.2.0.3版本，其中注入恶意代码

2019.6

RubyGems包括chrome_taker,color_hacker,aloha_anal-yser,get-text,ruby_nmap,get-texts等从库中删除，因为包含加密挖掘或cookie/密码窃取的代码

2020.4

362 RubyGems
用于拼写和加密挖掘的恶意软件，其中包括“atlas-client”（被开发人员下载2100次）

2020.1

Pyhton3-dateutil和jilyfish两个木门pypi包被发现从受感染开发者的项目中窃取SSH和GPG秘钥

2019.10

Basic authable2017年发布的由于其恶意性质被gems仓库删除

2019.7

Libpeshnx该包被报告为包含已知漏洞

# 开源软件供应链风险：开源项目维护性

开源项目活跃度低，维护能力不足等原因，造成供应链中开源组件缺乏维护

91%

•在Black Duck审计服务团队2020年审计的1,500多个代码库中，居然有91%使用了在过去两年中没有发生任何 开发活动的开源依赖项，这意味着91%的被审代码库中包含在过去两年中没有进行过功能升级、代码优化和任何安全问题修复的依赖项。

85%

•Black Duck审计服务团队2020年审计的代码库中，85%的代码库含有至少四年未曾更新的开源依赖项。也就是说，代码库使用的开源库并非最新版本，甚至经常是很旧的版本。如前所述，开发团队显然难以维护开源依赖项的时新性。

- JavaScript 的模块化标准库 core-js ，这个库通过 npm 软件包管理器下载，每周居然高达 2600 万次。甚至连苹果公司的网络服务，也用到了这个库，但是由于作者Denis Pushkarev除了写程序外，平时爱好是飚摩托车，结果在一次事故中，他因驾驶摩托撞人,致一死一伤，被判处18个月有期徒刑。

- 另一个 JavaScript 加密库 jsrsasign 也遇到过类似的挑战。自 2018 年 4 月以来，项目就没有任何活动。但是在 npm 上却有 350 个项目完全依赖这个库，项目也受到了微软、Mozilla 这样有影响力公司的青睐。

OWASP

Open Web Application Security Project

# 开源组件的治理实践

# 开源组件治理实践

## 建立开源组织体系

- 软件供应链角度：包括软件开发，软件交付和软件使用等多个环节，需要跨组织、跨部门协同管理
- 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

## 建立开源管理体系

围绕开源技术进行全生命周期的理念，即从开源技术的**导入、使用、更新、升级和退出各个阶段进行管理**

| | |
|---|---|
| **导 入** | 从软件开发阶段就建立开源软件使用的统一策略用什么、用什么版本、确认来源。建立安全准入机制，引入开源软件前先评估安全风险 |
| **使用\更新\升级** | 建立开源软件使用、更新和升级的规范和流程制度，持续检测，持续跟踪漏洞情报和响应 |
| **退 出** | 对于老旧的开源组件或者不符合安全标准的开源组件要及时做好退出管理，做好记录和追踪，并通过设置黑名单、阻断策略规则来隔离退出的组件 |

## 利用软件成分分析（SCA）工具

# 开源组件治理实践

利用软件成分分析工具

**SCA**

问 题

1. 安全漏洞和知识产权的数据问题，如何确保数据的完整性、实时性和准确性？
2. 怎么找出项目到底用了哪些开源组件以及它们的依赖信息？
3. 怎么找出项目中的开源文件或者片段代码拷贝信息？
4. 面对上游的不可控性，下游如何应对？

# 数据问题

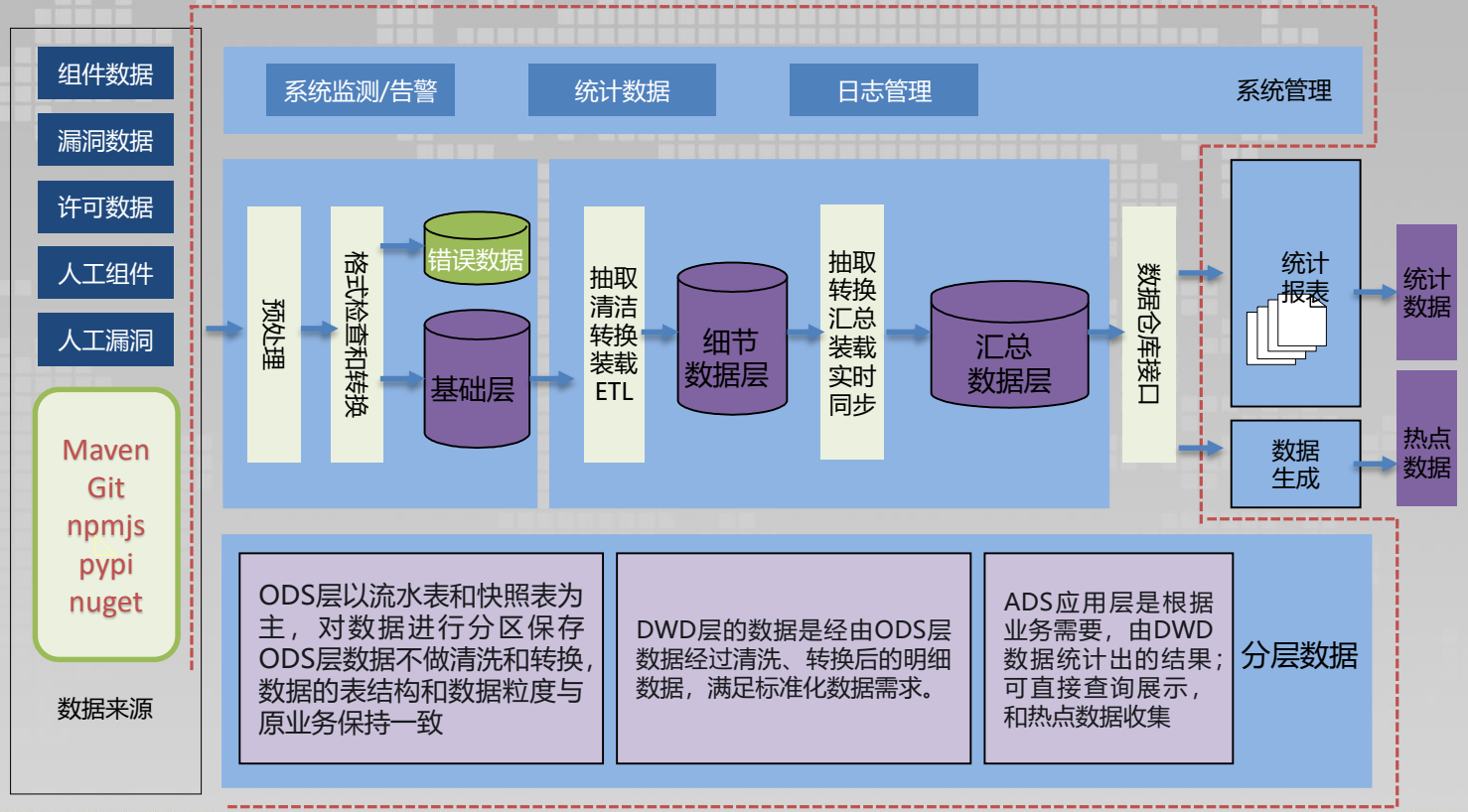
- **官方数据不准确**：cpe和描述信息范围扩大，忽略了更小粒度的组件信息。如CVE-2015-0254这个漏洞（XML外部实体注入），官网显示影响的是taglibs组件，但是真实原因是taglibs引用了更小粒度的jstl标签导致的。但是jstl这个组件确没有报该漏洞
- **数据延迟**：NVD数据比官方延迟很大，最长的漏洞延迟时间是11年，CVE-2009-4067在2009年11月24日获得CVE编号，而直到2020年11月2日NVD官方才发布

- Jackson-databind组件
- **版本推荐难**：同时维护多个版本，如2.6.*和2.7.*不兼容
- **修复建议难**：黑名单文件位置变化

| jackson-databind版本系列 | 无黑名单机制 | 引入黑名单机制（BeanDeserializerFactory.java） | 变更黑名单文件路径（SubTypeValidator.java） |
|---|---|---|---|
| 2.0.* | All | | |
| 2.1.* | All | | |
| 2.2.* | All | | |
| 2.3.* | All | | |
| 2.4.* | All | | |
| 2.5.* | All | | |
| 2.6.* | 2.6.0-2.6.7 | 2.6.7.1-2.6.7.4 | |
| 2.7.* | 2.7.0-2.7.9 | 2.7.9.1 | 2.7.9.2-2.7.9.7 |
| 2.8.* | 2.8.0-2.8.8 | 2.8.8.1-2.8.10 | 2.8.11-2.8.11.6 |
| 2.9.* | 2.9.0.pr1-2.9.0.pr2 | 2.9.0,2.9.0.pr3-2.9.3 | 2.9.4-2.9.10.8 |
| 2.10.* | | | All |
| 2.11.* | | | All |
| 2.12.* | | | All |

# 数据治理

1、需要一些高效的系统或者工具，处理好数据的收集，优化和存储的过程
2、需要专业的数据团队，对组件和漏洞的数据要进行有效的分析
3、利用大数据技术，结合大数据平台构造数据仓库




OWASP
Open Web Application
Security Project

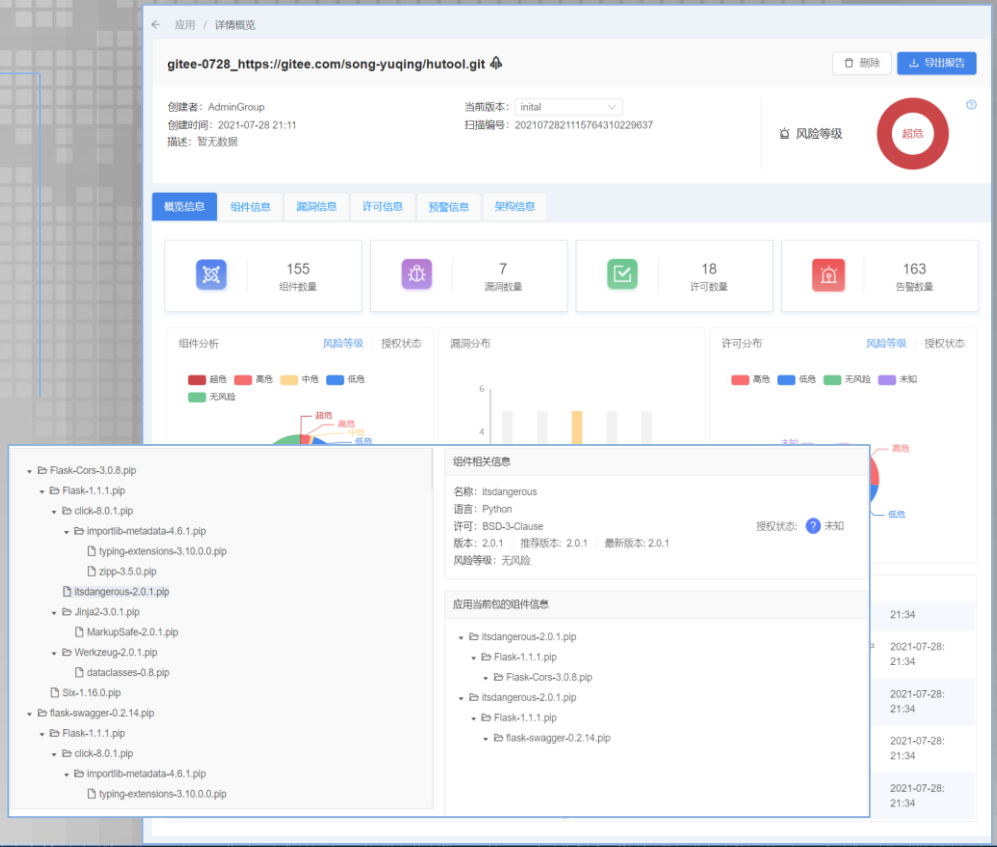
# 开源组件治理

## 找

- 依据软件成分分析技术，分析应用的开源成分，并将信息通过资产分布视图，多维度展示，可快速了解掌握应用的资产信息
- 支持N级依赖的解析，找出应用所有的直接和间接依赖信息，追根溯源，找出所有引用的上游开源组件
- 支持SBOM清单的分析、导出和导入

## 管

- 引入审核流程
- 建立项目-应用-组件的关系，定位组件的位置，便于使用和管理
- 建立黑白名单机制，加强对组件的管理
- 支持策略和规则配置，对项目应用进行有效控制，如阻断构建等




OWASP
Open Web Application Security Project

# 开源代码治理

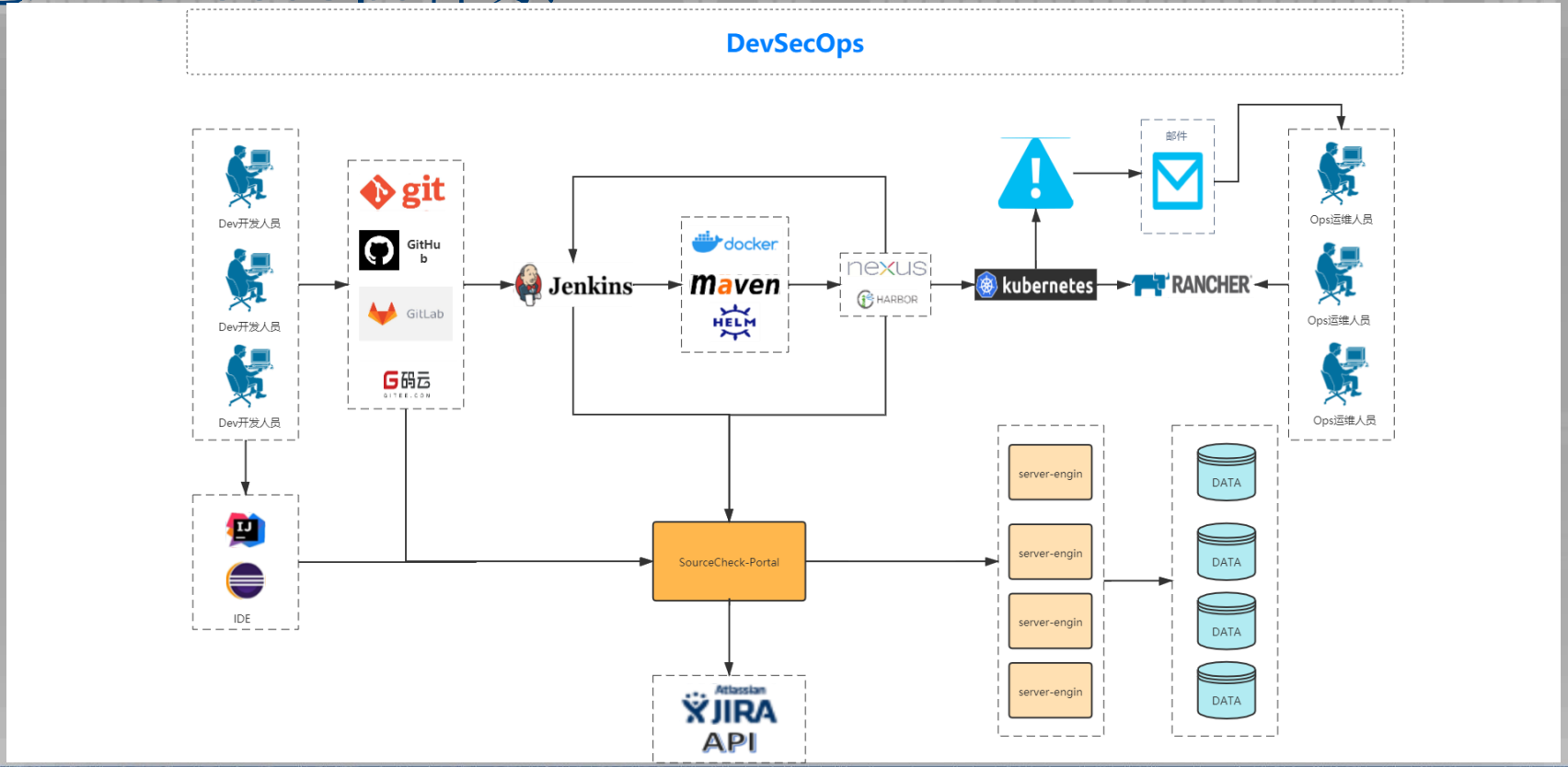
## 实 现

- 从组件包、文件、函数和代码片段四个角度找出应用中的开源成分，粒度由高到低，逐层分析，找到非直接引用开源组件的其他形式的开源成分。
- 在第一步的基础上，可以从不同的维度和粒度上计算代码的自研率。

## 探 索

- 漏洞可达性分析
- 同源漏洞分析

# 建立DevSecOps体系




OWASP
Open Web Application Security Project