OWASP
Open Web Application
Security Project

企业级 DevSecOps
开源治理方案演进之路

JFrog

THE SPEAKER

- 李威
- JFrog中国DevOps解决方案架构师
- DevOps教练，GDevOps、TGO鲲鹏会金牌讲师。曾就职于京东、烽火等互联网企业及传统企业，十年一线开发及运维经验，带领团队从零到一实践DevOps转型。

内容

1. 企业开源软件治理的挑战
2. “4步工作法”应对开源风险
3. 开源治理最佳实践
4. 问答

开源不等于安全

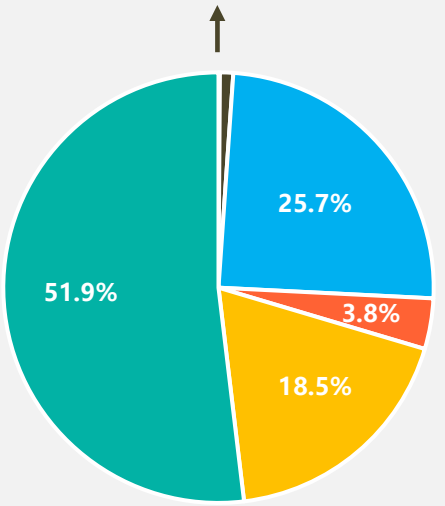
开源 != 安全

- 开源组件几乎没有安全测试
- 提供方没有安全意识
- 组件使用者不关心第三方组件代码
- 一个漏洞，影响范围广
- 开源社区分享精神，维护者轻易把项目管理权交给其它人员
- 黑客的目标一般是开源组件
- 78% 的漏洞存在于间接依赖关系中

商业 != 安全

- 商业软件依赖开源(被动依赖)
- Linux: redhat, Suse
- Openstack
- Kubenates
- Hadoop

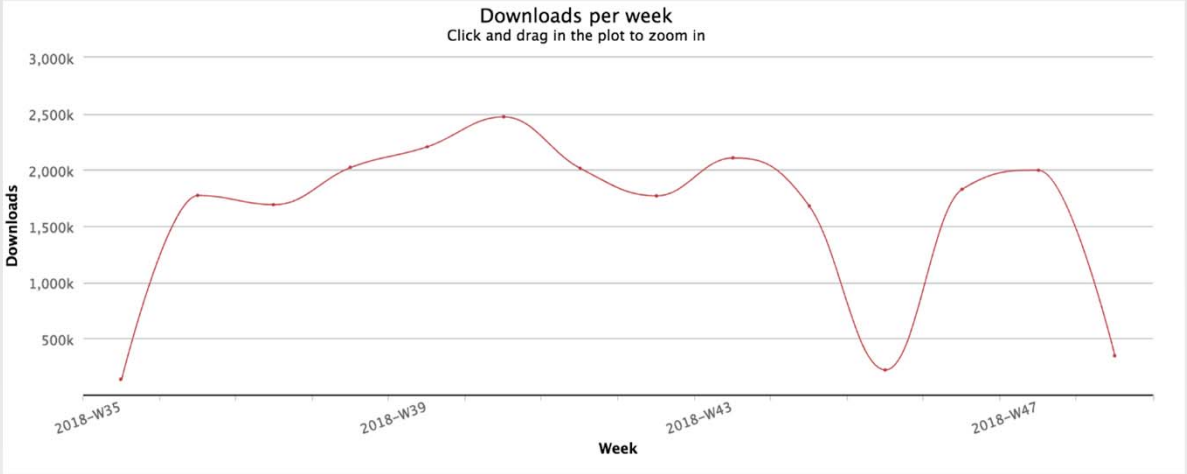
我们的代码 <0.1% 在整个软件中



- Self
- Linux
- Landscape
- Node_modules
- Node.js
- Kubernetes

Npm：event-stream事件

event-stream包是一个Node.js流数据的JavaScript软件包。起因是 event-stream 项目的作者由于时间和精力有限，将其维护工作交给另一位开发者Right9ctrl，该开发者获得了event-stream的控制权，将恶意代码注入。注入的恶意代码将会窃取比特币用户钱包内的私钥并发送至一个域名。



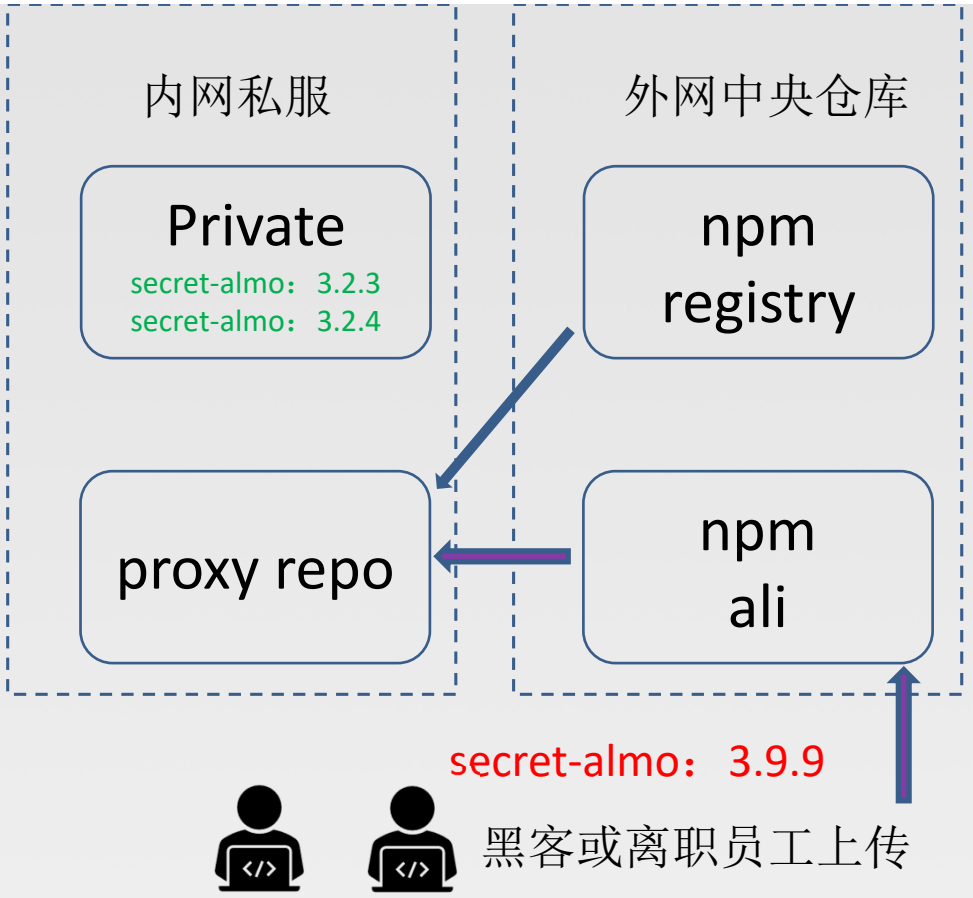
- 周下载量在200万+次
- 持续时间为2.5个月
- 大约2000万+次的下载量
- 其他开源组件也有依赖

软件供应链攻击（依赖混淆攻击）

需要依赖 secret-almo 组件

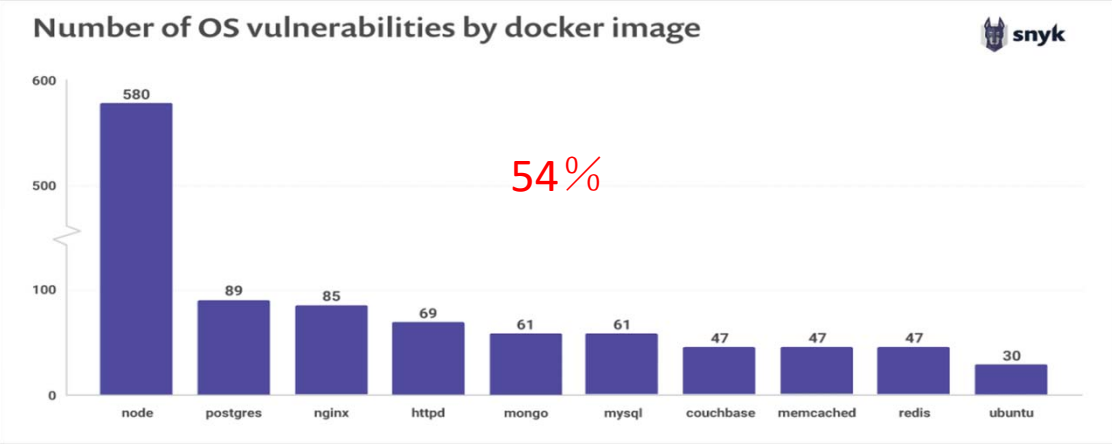
secret-almo: ~ 3.2.0 匹配3.2.x最新版本
secret-almo: ^ 3.0.0 匹配3.x.x最新版本
secret-almo: * 匹配最新版本

需要获取3.2.4
实际获取3.9.9



各种语言的漏洞情况

Heartbleed



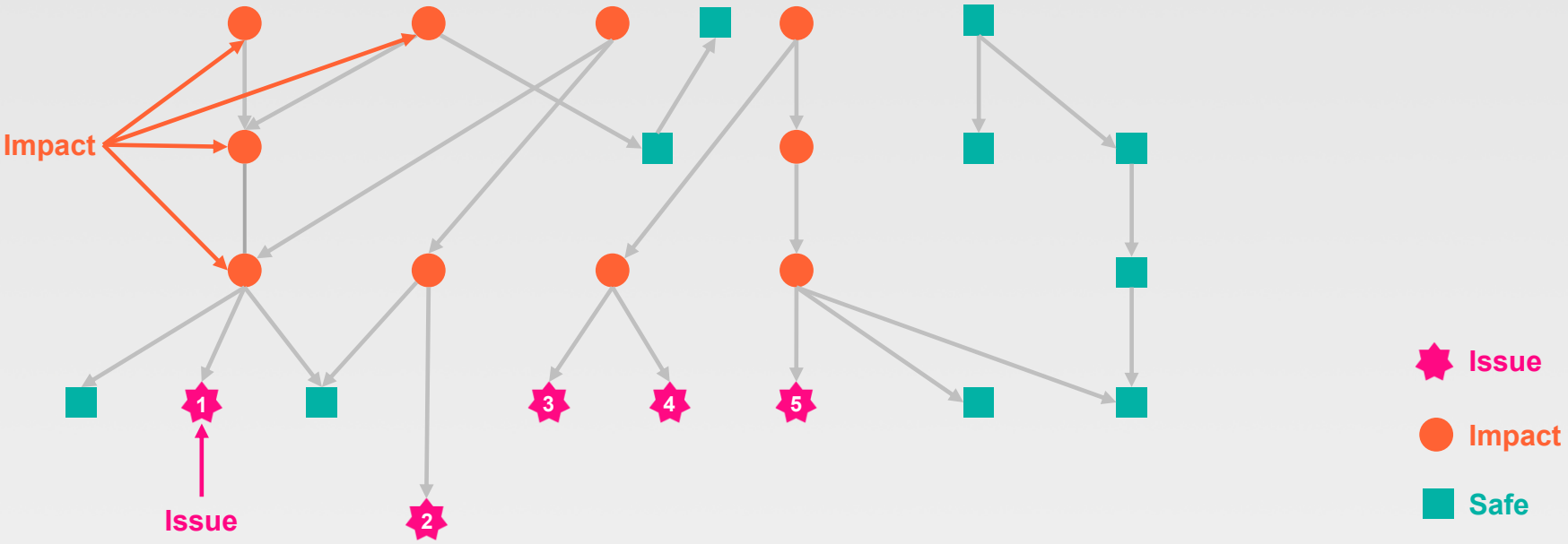
30% Docker 镜像
包含已知漏洞

14% Npm package
包含已知漏洞

59% Maven 已发现
漏洞仍未修复

第三方组件安全管控难点

无法定位问题影响范围 -- 精准分析难



企业开源治理痛点

某个漏洞爆发了，如何排查哪些项目组引用了？如何保证没有漏报？

- 全公司发邮件，自查，导致不相干研发也会受到，不堪其扰，拒绝配合。

如何阻止漏洞包的下载，及时止损？

- 全公司发邮件，自行阻止。导致研发在不知情的情况下仍然使用带高危漏洞的版本上线。

如何尽早的将安全问题左移到开发解决？

- 无，安全风险压力完全堆积在安全团队，存在带伤上线风险。

企业开源治理痛点

大多数企业缺少开源组件及软件的协议分析、漏洞评估及修复能力

企业使用开源软件缺乏安全评估、法务评估和引入流程

企业不清楚项目中使用了多少开源软件和开源组件

企业在开源软件或组件出现漏洞时，无法快速定位到漏洞组件的影响范围，并及时止损，禁止漏洞组件下载

内容

如何管控治理?

统一入口 & 控制源头

jcenter，dockerHub 等大的第三方源站

依赖组件分析管理

通过搭建内网私服制品库统一管理并通过构建工具进行组件分析

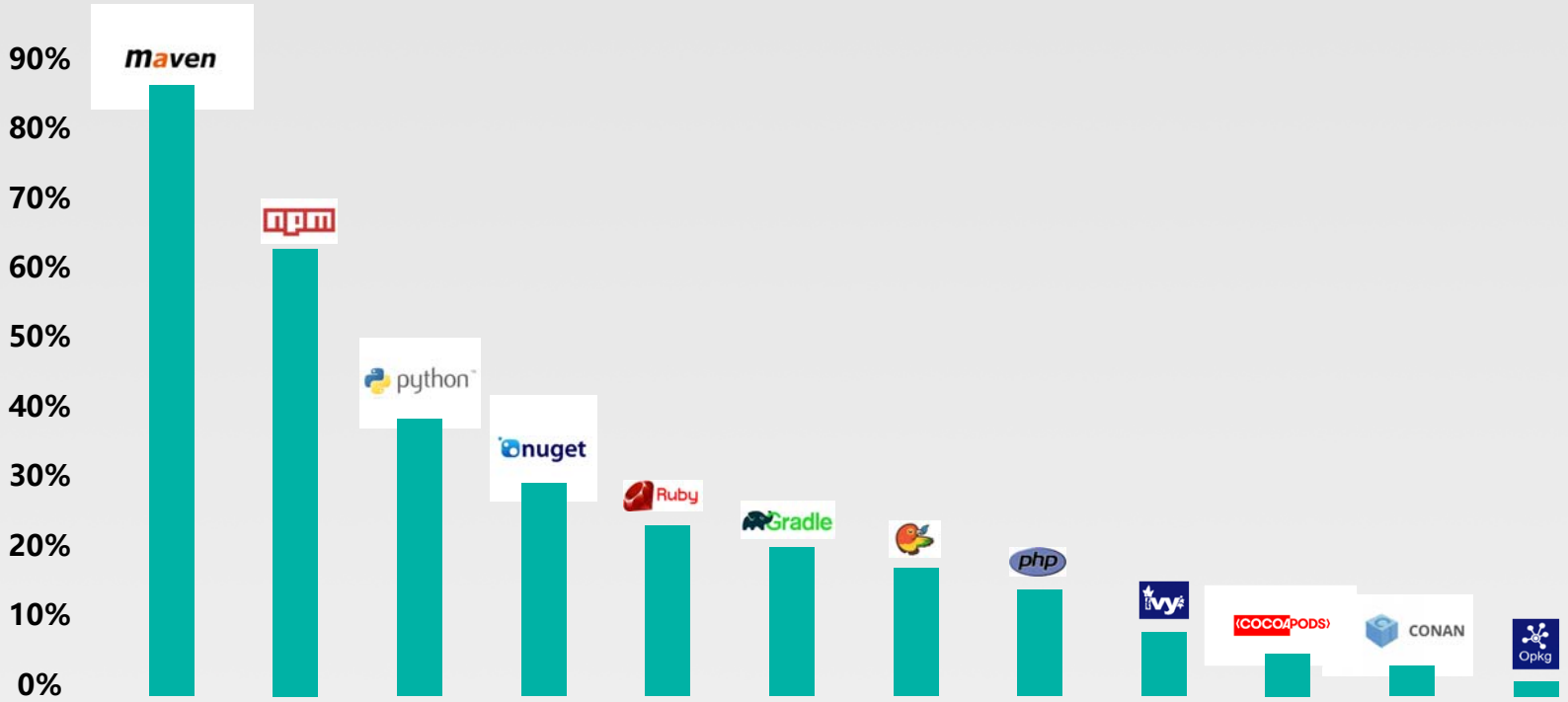
漏洞数据源

NVD、CNVD、VlunDB

安全左移 & 全生命周期管理

从开发、构建、部署、运行进行全生命周期管控 & 治理

全球二进制依赖管理工具使用情况

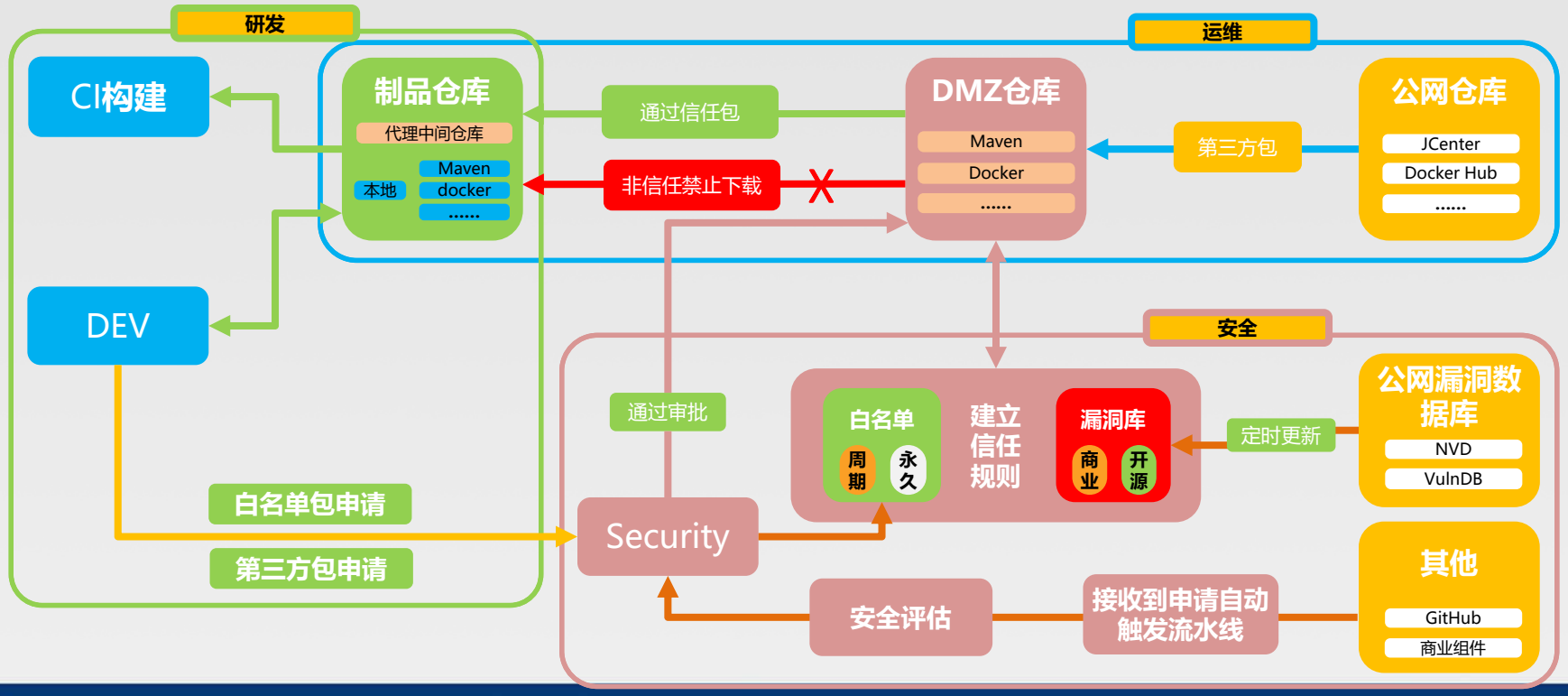


第一步：单一可信源


Nexus
NPM
Docker Registry
Nuget
制品库管理统一管理
docker
maven
npm
python
gradle
运维人员
开发团队
开发团队
开发团队
开发团队
开发团队
开发团队
开发团队
安全管控
下载更快
可信互联网共有仓库
统一开源漏洞扫描
互联网
研发测试区
生产区

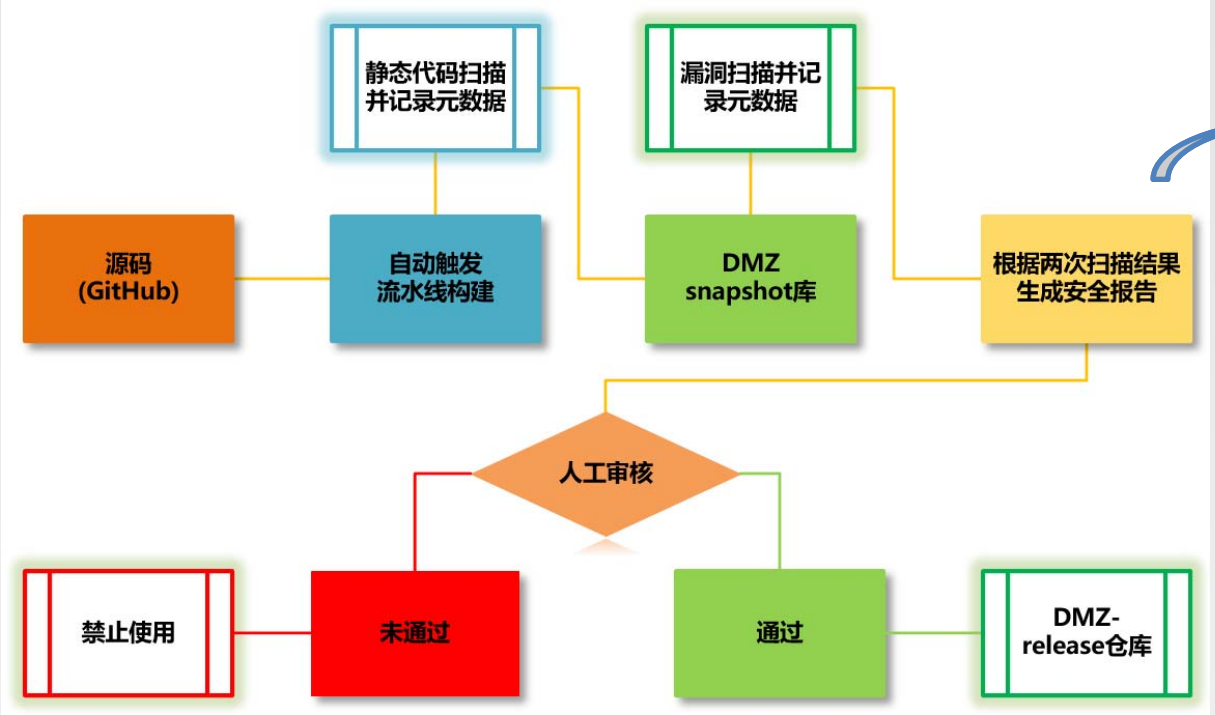
第三方组件及商业组件

非仓库第三方自构建包申请						
申请人	申请原因	包名	版本	作者	Github地址	其他
张	某系统	x.jar	1.0	王	github.com	.....



第三方源码管理方案

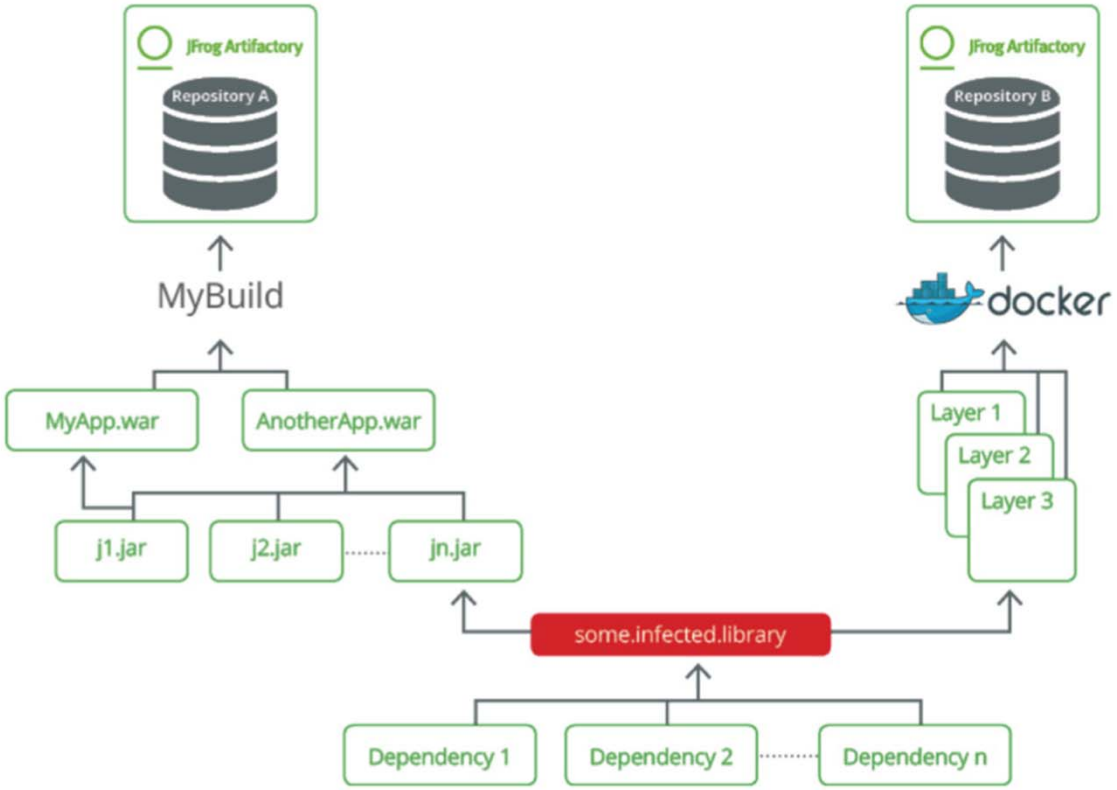
- 自动流水线第三方包构建
- 生成安全报告人工审核
- 保证依赖制品来源可追溯
- 根据场景定制安全标准



数据指标	数据类型	门限值	权重比
qa.code.qulity.alert_status	string	OK	15%
qa.code.qulity.coverage	float	> 80%	5%
qa.code.qulity.duplicated_lines_density	float	< 0.1	5%
qa.code.qulity.comment_lines_density	float	> 0.1	10%
qa.code.qulity.new_coverage	float	> 80%	5%
qa.code.qulity.security_rating	string	A	10%
qa.code.qulity.reliability_rating	string	A	5%
qa.ut.passrate	float	> 0.8	10%
qa.xray.high	int	=0	25%
qa.xray.medium	int	<5	10%
qa.xray.lower	int	<10	10%

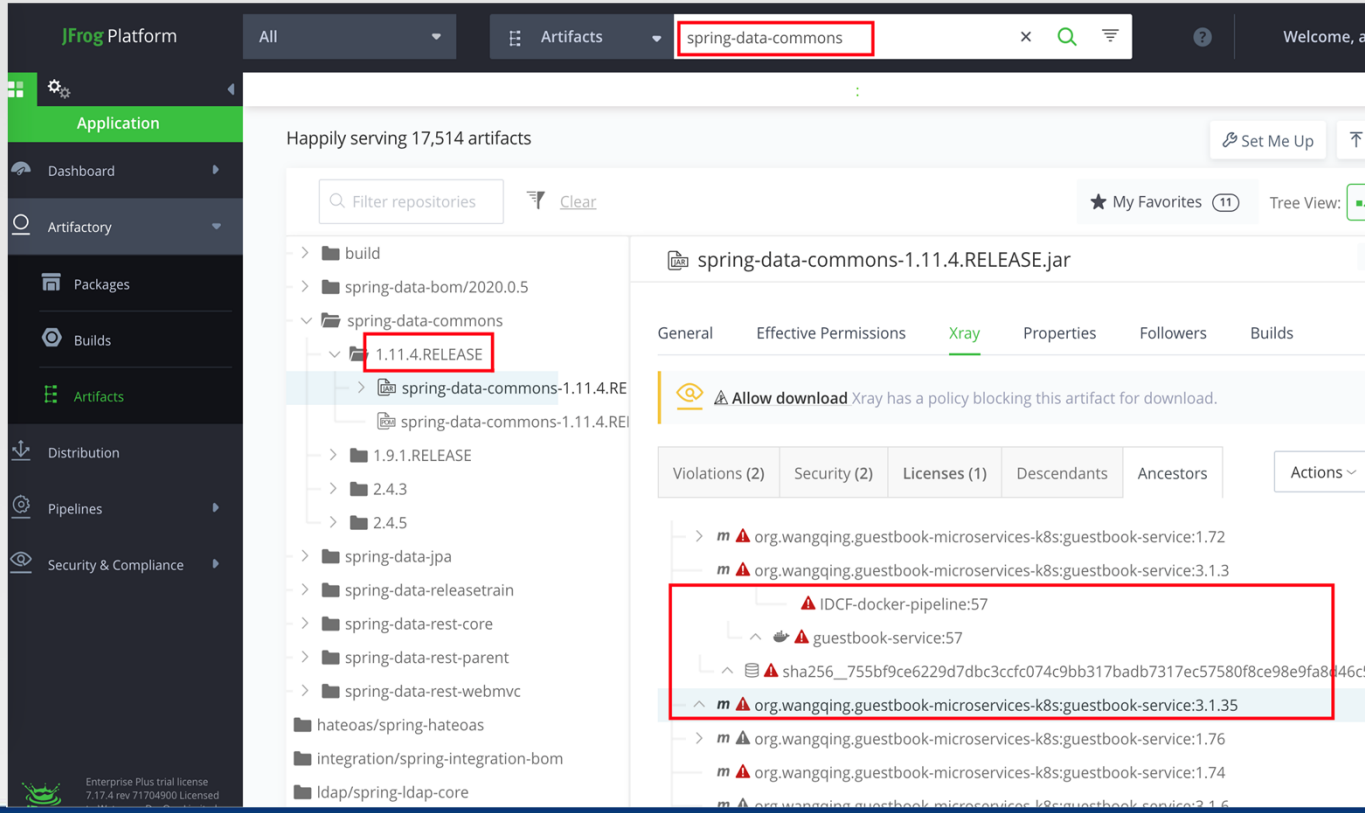
依赖关系分析

- 存量监控
- 影响性分析
- 依赖关系



影响定位

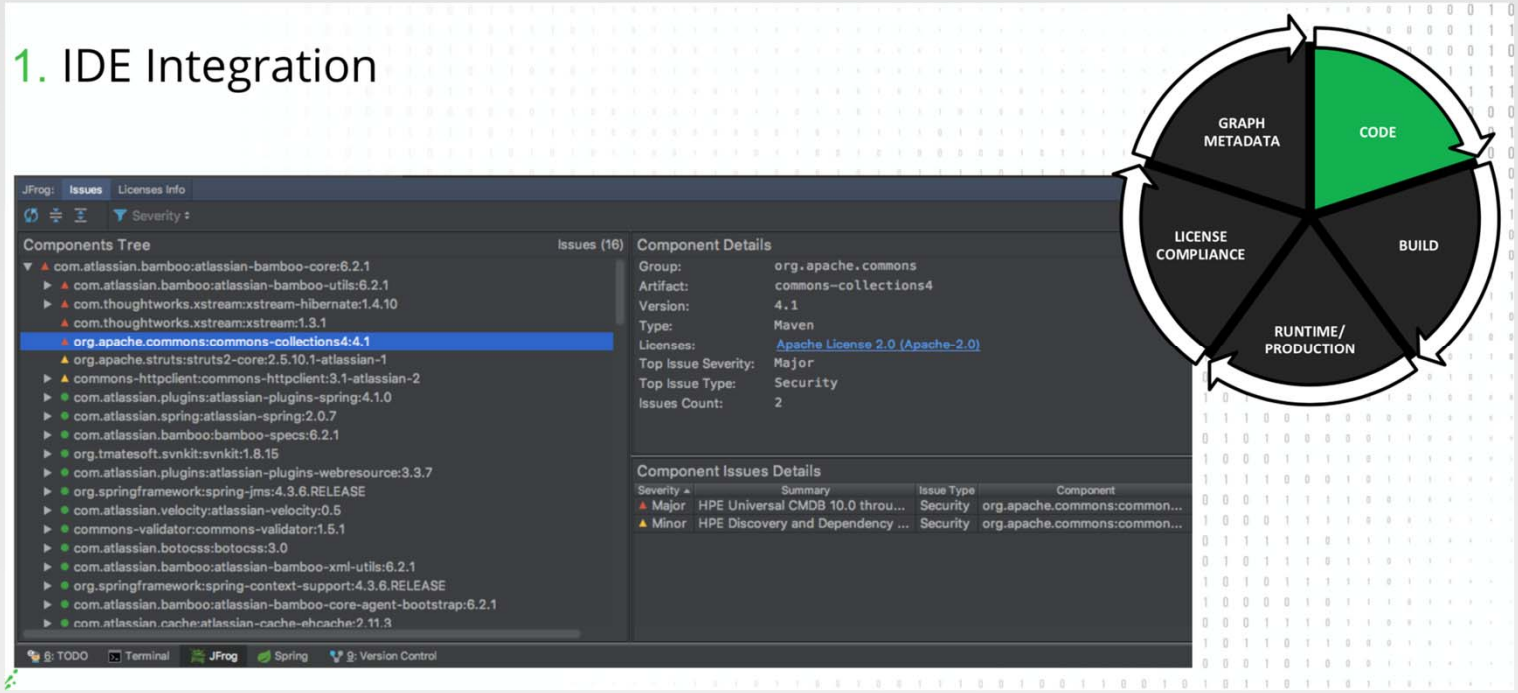
正反向依赖分析
风险阻断



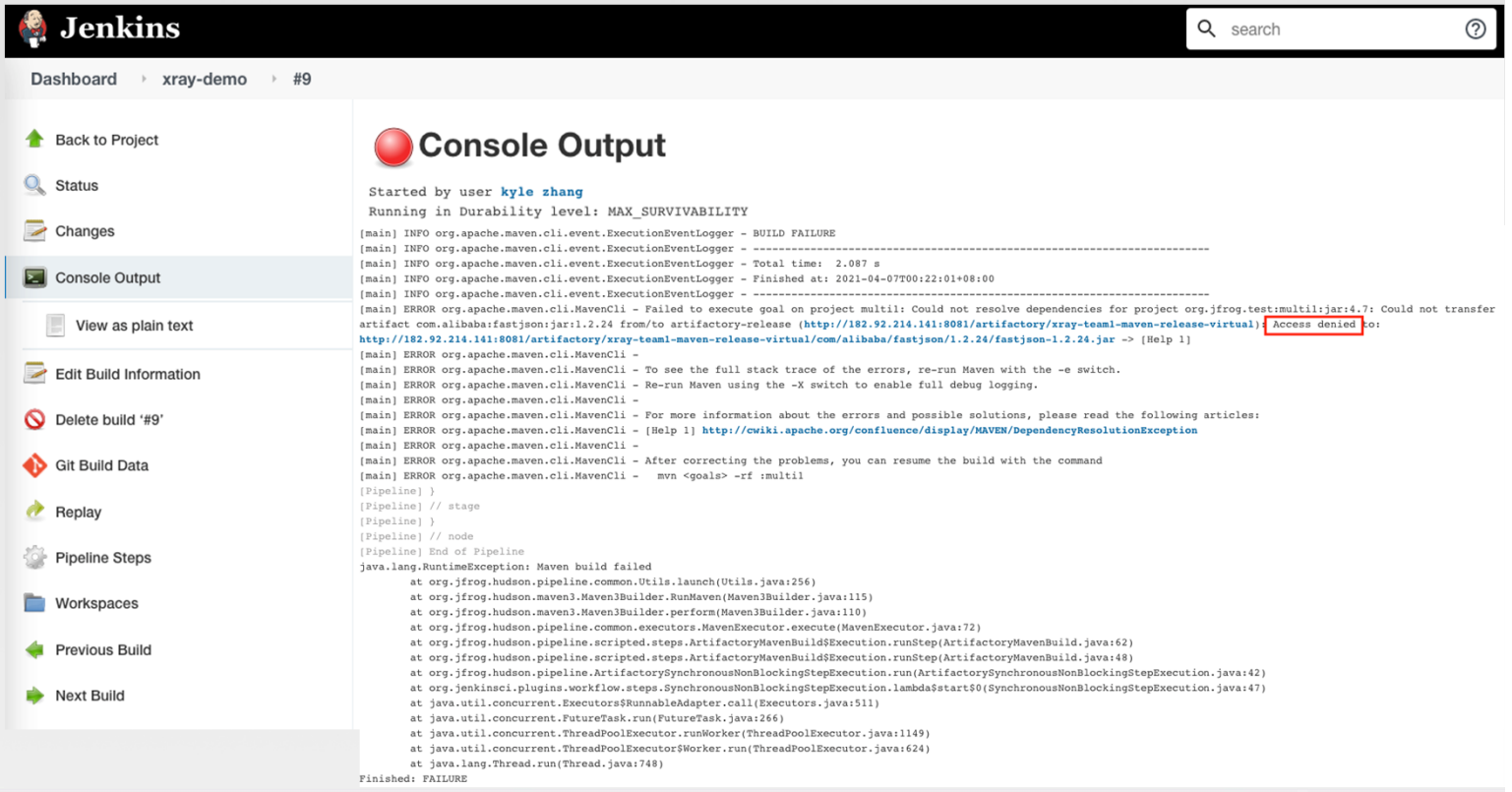
组件 & 漏洞数据源

技术栈	组件信息源	漏洞信息源
Java	Https://jcenter.bintray.com	NVD:https://nvd.nist.gov CNVD:https://www.cnvd.org.cn/
Javascrip(Nodejs)	https://registry.npmjs.org	NVD:https://nvd.nist.gov CNVD:https://www.cnvd.org.cn/
Net(Nuget)	https://www.nuget.org	NVD:https://nvd.nist.gov CNVD:https://www.cnvd.org.cn/
Debian	Http://archive.ubuntu.com.ubuntu	Debian:https://security-tracker.debian.org/tracker Ubuntu:https://launchpad.net/ubuntu-cve-tracker
Rpm	http://mirror.centos.org/centos	https://www.redhat.com/security/data/oval/
Python(pypi)	https://pypi.python.org/pypi	NVD:https://nvd.nist.gov CNVD:https://www.cnvd.org.cn/
Ruby(gems)	https://rubygems.org/	NVD:https://nvd.nist.gov CNVD:https://www.cnvd.org.cn/

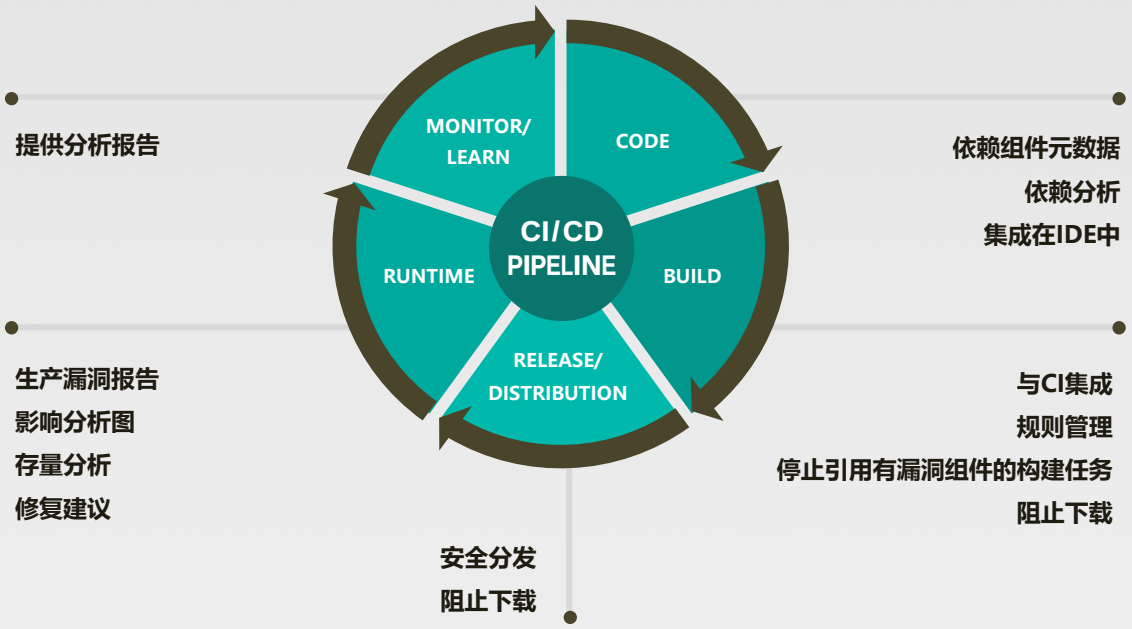
安全左移


1. IDE Integration
JFrog: Issues Licenses Info
Severity
Components Tree
Issues (16)
com.atlassian.bamboo:atlassian-bamboo-core:6.2.1
com.atlassian.bamboo:atlassian-bamboo-utils:6.2.1
com.thoughtworks.xstream:xstream-hibernate:1.4.10
com.thoughtworks.xstream:xstream:1.3.1
org.apache.commons:commons-collections4:4.1
org.apache.struts:struts2-core:2.5.10.1-atlassian-1
commons-httpclient:commons-httpclient:3.1-atlassian-2
com.atlassian.plugins:atlassian-plugins-spring:4.1.0
com.atlassian.spring:atlassian-spring:2.0.7
com.atlassian.bamboo:bamboo-specs:6.2.1
org.tmatesoft.svnkit:svnkit:1.8.15
com.atlassian.plugins:atlassian-plugins-webresource:3.3.7
org.springframework:spring-jms:4.3.6.RELEASE
com.atlassian.velocity:atlassian-velocity:0.5
commons-validator:commons-validator:1.5.1
com.atlassian.botocss:botocss:3.0
com.atlassian.bamboo:atlassian-bamboo-xml-utils:6.2.1
org.springframework:spring-context-support:4.3.6.RELEASE
com.atlassian.bamboo:atlassian-bamboo-core-agent-bootstrap:6.2.1
Component Details
Group: org.apache.commons
Artifact: commons-collections4
Version: 4.1
Type: Maven
Licenses: Apache License 2.0 (Apache-2.0)
Top Issue Severity: Major
Top Issue Type: Security
Issues Count: 2
Component Issues Details
Severity Summary Issue Type Component
Major HPE Universal CMDB 10.0 throu... Security org.apache.commons:common...
Minor HPE Discovery and Dependency ... Security org.apache.commons:common...
6: TODO Terminal JFrog Spring 9: Version Control
GRAPH METADATA
CODE
LICENSE COMPLIANCE
BUILD
RUNTIME/ PRODUCTION

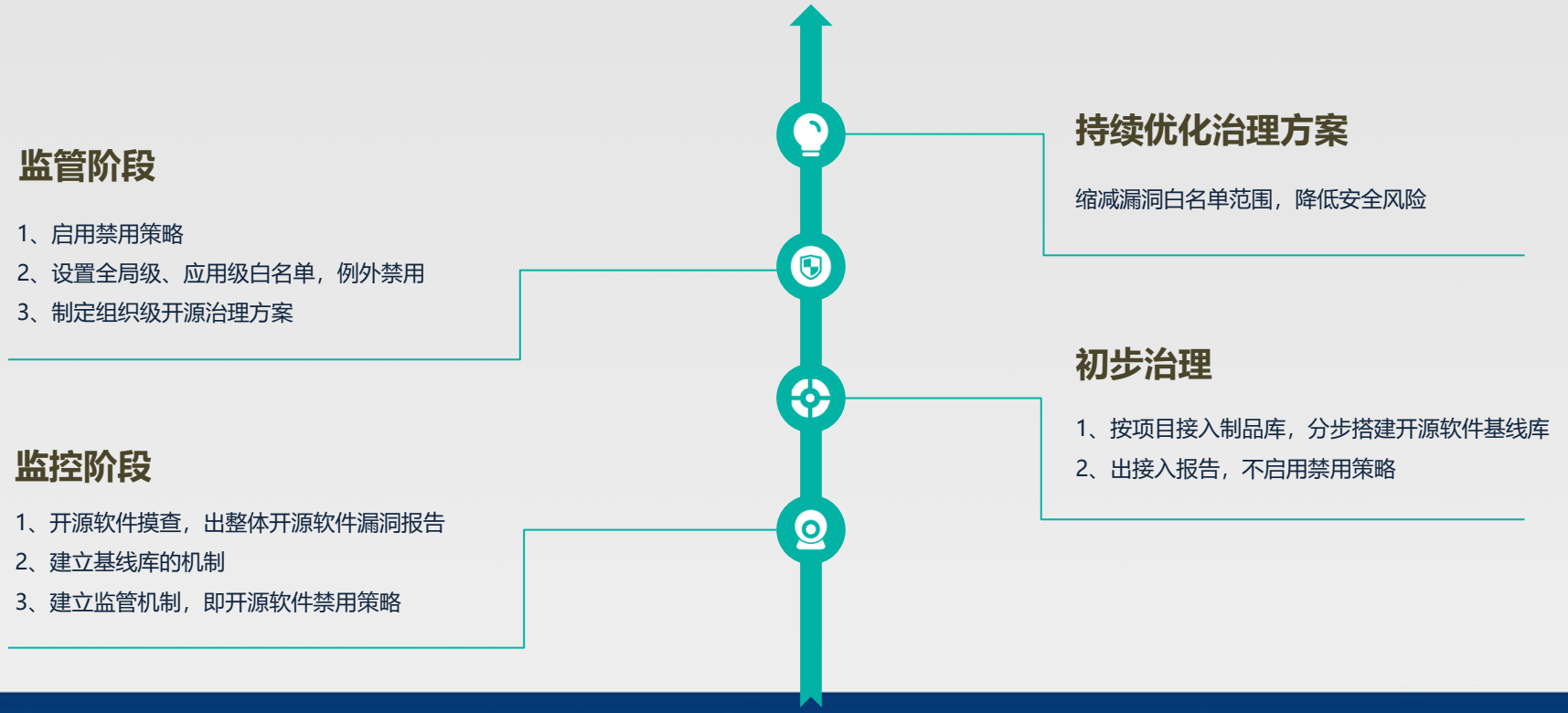
安全门禁


Jenkins
search
Dashboard
xray-demo
#9
Back to Project
Status
Changes
Console Output
View as plain text
Edit Build Information
Delete build '#9'
Git Build Data
Replay
Pipeline Steps
Workspaces
Previous Build
Next Build
Console Output
Started by user kyle zhang
Running in Durability level: MAX_SURVIVABILITY
[main] INFO org.apache.maven.cli.event.ExecutionEventLogger - BUILD FAILURE
[main] INFO org.apache.maven.cli.event.ExecutionEventLogger - --
[main] INFO org.apache.maven.cli.event.ExecutionEventLogger - Total time: 2.087 s
[main] INFO org.apache.maven.cli.event.ExecutionEventLogger - Finished at: 2021-04-07T00:22:01+08:00
[main] INFO org.apache.maven.cli.event.ExecutionEventLogger - --
[main] ERROR org.apache.maven.cli.MavenCli - Failed to execute goal on project multi1: Could not resolve dependencies for project org.jfrog.test:multi1:jar:4.7: Could not transfer artifact com.alibaba:fastjson:jar:1.2.24 from/to artifactory-release (http://182.92.214.141:8081/artifactory/xray-teaml-maven-release-virtual): Access denied to: http://182.92.214.141:8081/artifactory/xray-teaml-maven-release-virtual/com/alibaba/fastjson/1.2.24/fastjson-1.2.24.jar -> [Help 1]
[main] ERROR org.apache.maven.cli.MavenCli -
[main] ERROR org.apache.maven.cli.MavenCli - To see the full stack trace of the errors, re-run Maven with the -e switch.
[main] ERROR org.apache.maven.cli.MavenCli - Re-run Maven using the -X switch to enable full debug logging.
[main] ERROR org.apache.maven.cli.MavenCli -
[main] ERROR org.apache.maven.cli.MavenCli - For more information about the errors and possible solutions, please read the following articles:
[main] ERROR org.apache.maven.cli.MavenCli - [Help 1] http://cwiki.apache.org/confluence/display/MAVEN/DependencyResolutionException
[main] ERROR org.apache.maven.cli.MavenCli -
[main] ERROR org.apache.maven.cli.MavenCli - After correcting the problems, you can resume the build with the command
[main] ERROR org.apache.maven.cli.MavenCli - mvn <goals> -rf :multi1
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline
java.lang.RuntimeException: Maven build failed
at org.jfrog.hudson.pipeline.common.Utils.launch(Utils.java:256)
at org.jfrog.hudson.maven3.Maven3Builder.RunMaven(Maven3Builder.java:115)
at org.jfrog.hudson.maven3.Maven3Builder.perform(Maven3Builder.java:110)
at org.jfrog.hudson.pipeline.common.executors.MavenExecutor.execute(MavenExecutor.java:72)
at org.jfrog.hudson.pipeline.scripted.steps.ArtifactoryMavenBuild$Execution.runStep(ArtifactoryMavenBuild.java:62)
at org.jfrog.hudson.pipeline.scripted.steps.ArtifactoryMavenBuild$Execution.runStep(ArtifactoryMavenBuild.java:48)
at org.jfrog.hudson.pipeline.ArtifactorySynchronousNonBlockingStepExecution.run(ArtifactorySynchronousNonBlockingStepExecution.java:42)
at org.jenkinsci.plugins.workflow.steps.SynchronousNonBlockingStepExecution.lambda$start$0(SynchronousNonBlockingStepExecution.java:47)
at java.util.concurrent.Executors$RunnableAdapter.call(Executors.java:511)
at java.util.concurrent.FutureTask.run(FutureTask.java:266)
at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1149)
at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:624)
at java.lang.Thread.run(Thread.java:748)
Finished: FAILURE

JFrog Xray - 全生命周期保护



解决方案案例 - 开源治理方案及路线图



监管阶段

1、启用禁用策略
2、设置全局级、应用级白名单，例外禁用
3、制定组织级开源治理方案

监控阶段

1、开源软件摸查，出整体开源软件漏洞报告
2、建立基线库的机制
3、建立监管机制，即开源软件禁用策略

持续优化治理方案

缩减漏洞白名单范围，降低安全风险

初步治理

1、按项目接入制品库，分步搭建开源软件基线库
2、出接入报告，不启用禁用策略

管控盲区 – License 管控

为什么要有开源协议?

1. 保护原作者的知识成果

防止被恶意利用。开源协议中一般都包含有免责声明，可以防止原作者承担相应风险和后果。比如你开源了一个破　解Windows秘钥的软件，而使用者却用来进行商业资料窃取，那么你是不需要为此承担责任的。

2. 保护使用者的权利

使用者可以知晓经授权和未经授权的操作。防止你使用未添加协议（可能未授权）的代码，而使原作者起诉你。

License管控的必要性

知识产权风险 | 违约风险 | 开源许可证兼容性风险 | 安全风险

随着开源软件的不断发展，社区里出现了各式各样的 License许可证，如果你使用了不合适的许可证软件，会为公司带来法律上的纠纷，同时，如果因为开源组件 License 选用不当，导致在交付的时候需要进行开源组件的替换，那随之带来的开发工作量也非常巨大。所以选择合适的许可证应该在第一时间进行。

开源软件提倡公开、自由与创新等开源精神，为推动软件产业的 发展起到了积极作用。但是，个人或企业在使用或引入开源软件的过 程中，将不可避免地面临知识产权上的风险。如个人或企业在使用或 引入开源软件，因为不了解知识产权风险而引起相关法律或商业争 议，将可能给个人或企业在经济或声誉等方面带来巨大的损失。

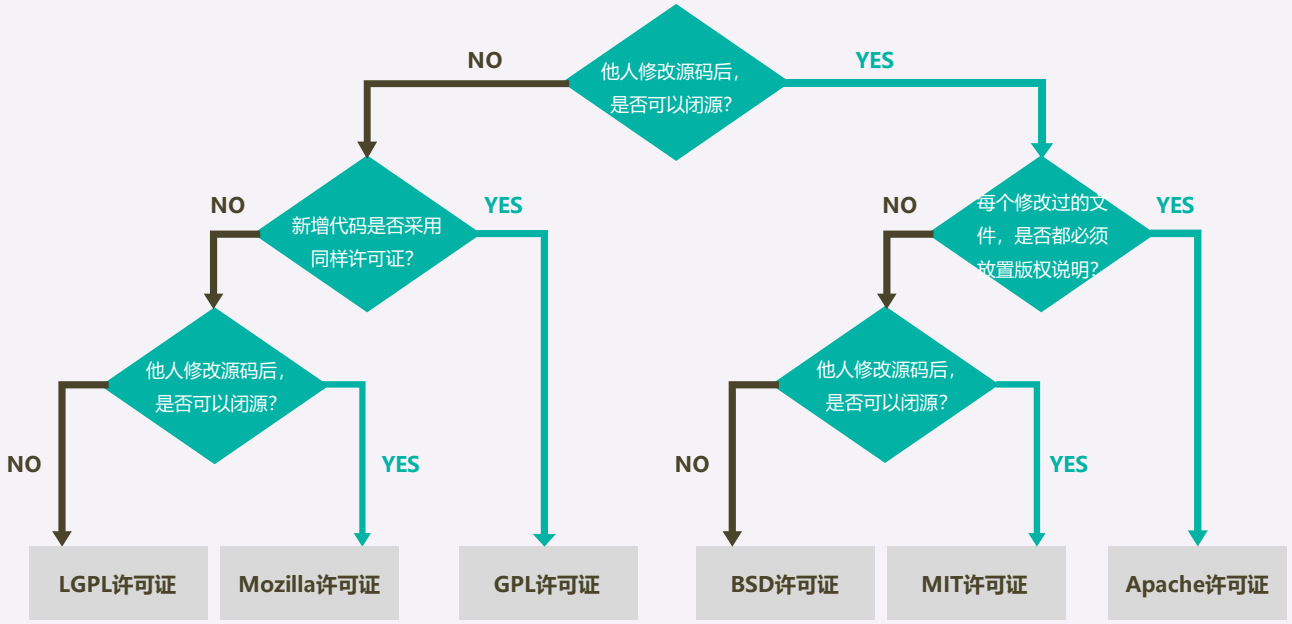
OSI开源协议查询的网站

https://opensource.org/licenses/alphabetical

常见的6种开源协议

- LGPL
- Mozilla
- GPL
- BSD
- MIT
- Apache

开源不等于免费，开源也不等于没有约束



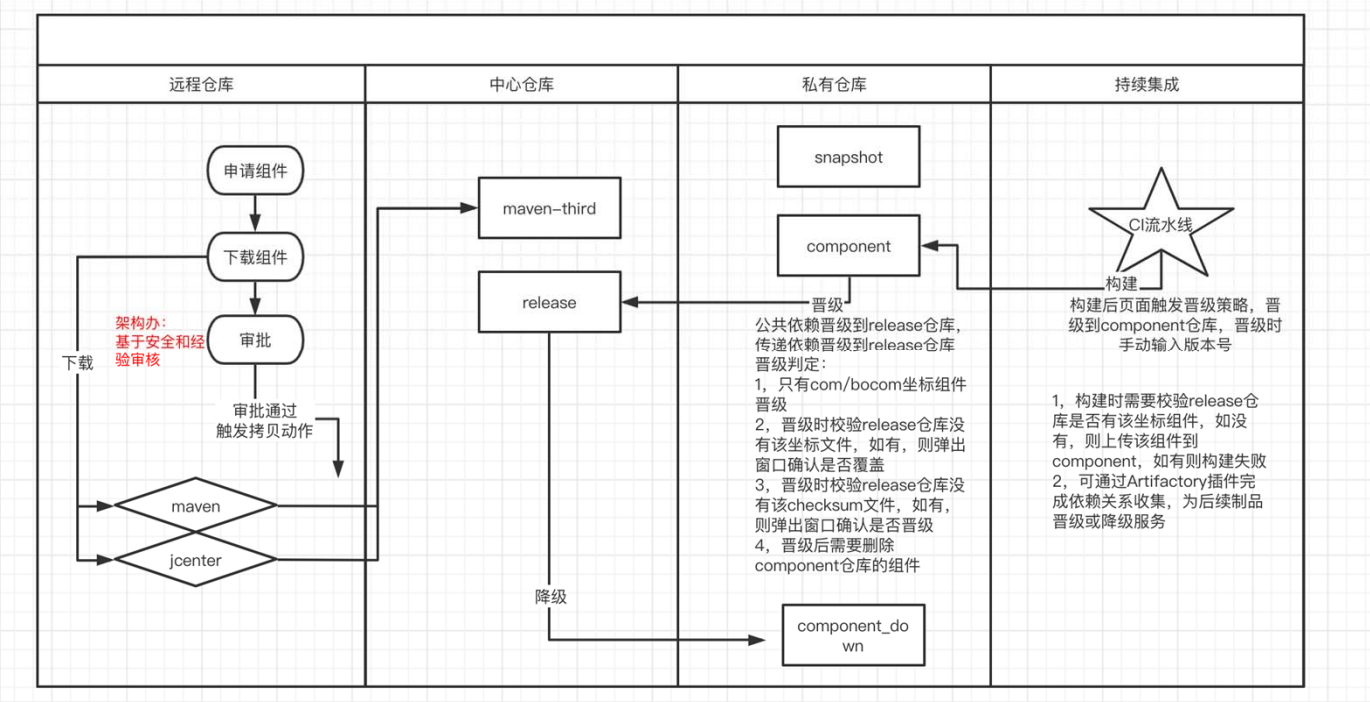
内容

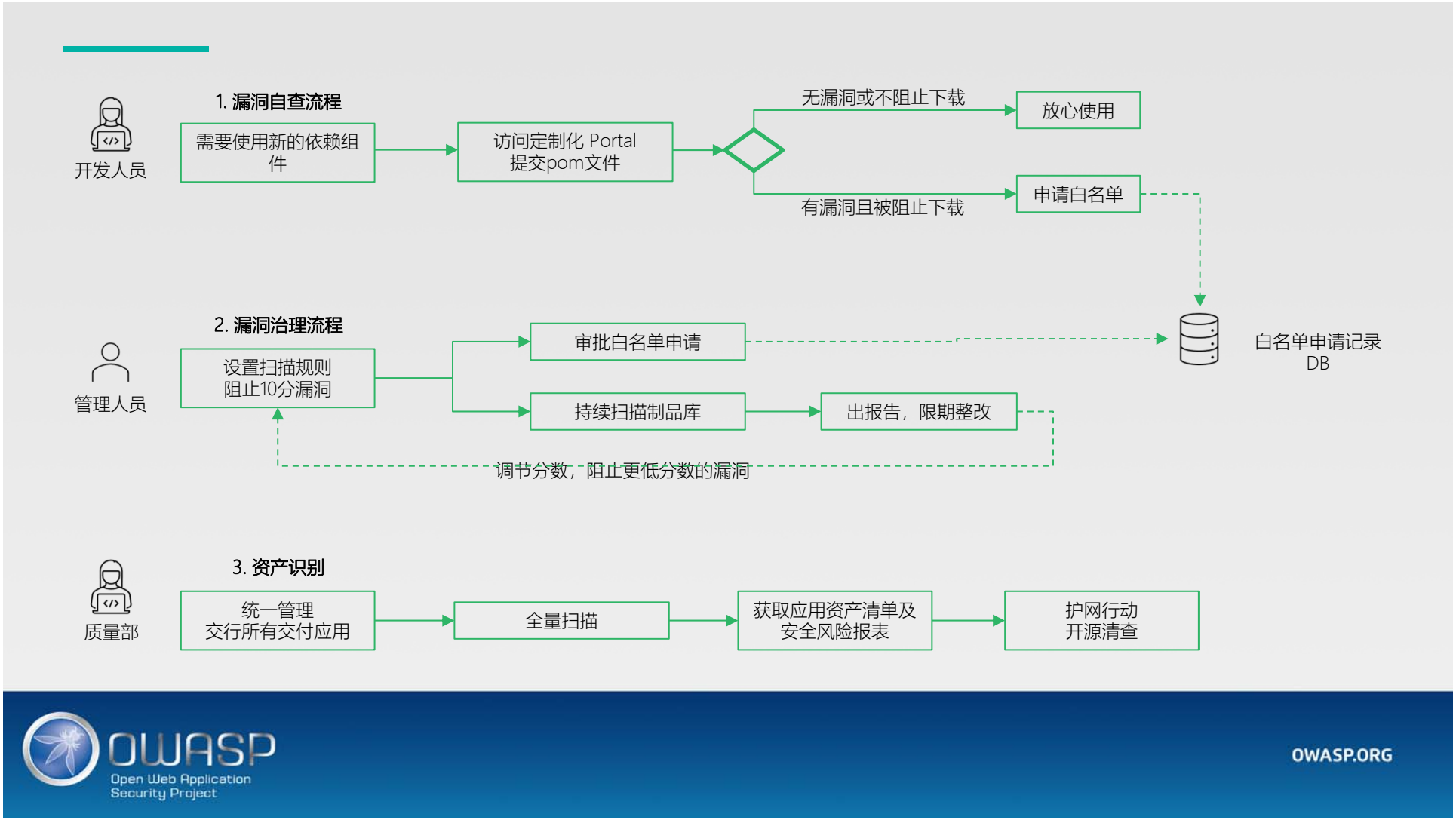
1. 企业开源软件治理的挑战
2. “4步工作法”应对开源风险
3. 开源治理最佳实践
4. 问答

开源治理解决方案


Open source governance solutions
统一引入管理
统一管理平台
统一审核流程
风险识别
开源许可证风险
安全风险
风险记录沟通
风险记录
沟通机制
风险处置
处置指导
漏洞修复

某银行开源治理方案


远程仓库
中心仓库
私有仓库
持续集成
申请组件
下载组件
审批
架构办：
基于安全和经验审核
下载
审批通过
触发拷贝动作
maven
jcenter
maven-third
release
降级
snapshot
component
component_down
晋级
公共依赖晋级到release仓库，
传递依赖晋级到release仓库
晋级判定：
1，只有com/bocom坐标组件晋级
2，晋级时校验release仓库没有该坐标文件，如有，则弹出窗口确认是否覆盖
3，晋级时校验release仓库没有该checksum文件，如有，则弹出窗口确认是否晋级
4，晋级后需要删除component仓库的组件
CI流水线
构建
构建后页面触发晋级策略，晋级到component仓库，晋级时手动输入版本号
1，构建时需要校验release仓库是否有该坐标组件，如没有，则上传该组件到component，如有则构建失败
2，可通过Artifactory插件完成依赖关系收集，为后续制品晋级或降级服务


1. 漏洞自查流程
开发人员
需要使用新的依赖组件
访问定制化 Portal
提交pom文件
无漏洞或不阻止下载
放心使用
有漏洞且被阻止下载
申请白名单
2. 漏洞治理流程
管理人员
设置扫描规则
阻止10分漏洞
审批白名单申请
持续扫描制品库
出报告，限期整改
调节分数，阻止更低分数的漏洞
白名单申请记录
DB
3. 资产识别
质量部
统一管理
交行所有交付应用
全量扫描
获取应用资产清单及
安全风险报表
护网行动
开源清查

JFrog ARTIFACTORY 主要价值

企业的制品唯一可信源

全语言制品统一管理 高可用高并发

代替nexus、harbor、svn等

提供全语言制品统一管理

减少仓库维护成本180 人天/年

《DevOps成熟度》可达制品库成熟度4+级标准

DevOps元数据 制品质量度量

记录软件包关联的Jira ID，测试结果，漏洞扫描，部署结果等信息

作为质量门禁，贯穿整个CICD

提升软件质量，用户依赖度提升

多地多环境协同

多环境多地去重同步，推拉式同步，多点分发，基于https协议，分片分发。

自动校验checksum

开源软件漏洞扫描

通过实时识别高危漏洞包，并阻止受污染的包上线。

减少排查漏洞的成本 100 人天/年

减少漏洞带来的企业损失不可估量

1. 使用 Artifactory

统一入口 & 控制源头

代理 maven central，spring 等大的第三方源站

参考制品库规范规划仓库

2. 使用 Xray

持续扫描全生命周期

IDE引入时扫描，CI构建时扫描

库上存储时扫描，审批、分发、部署时扫描

3. 使用 Xray

正反向依赖分析，快速止损

正反向依赖分析

快速定位止损

4. 应用 Xray

开源治理最佳实践

关注开发体验，安全左移

逐步治理，良性循环

Q & A

交流 DevSecOps

（个人微信）