QCon
全球软件开发大会

InfoQ 极客传媒

面向未来的DevSecOps：Kodem 如何用AI重塑应用安全

刘永强

技术总监

目录

InfoQ 极客传媒

北京

QCon 全球软件开发大会

会议时间：4月10-12日

- 大模型赋能 AIOps
- 越挫越勇的大前端
- 云上业务架构演进
- 多模态大模型及应用
- 海外 AI 应用创新实践

北京

AiCon 全球人工智能开发与应用大会

会议时间：6月27-28日

- 端侧智能
- AI Agent
- 多模态大模型
- 金融+大模型

上海

QCon 全球软件开发大会

会议时间：10月23-25日

- AI Agent
- 大模型训练推理
- 端侧 AI
- 搜推深度融合
- 数智金融

4月 — 5月 — 6月 — 8月 — 10月 — 12月

上海

AiCon 全球人工智能开发与应用大会

会议时间：5月23-24日

- 通用大模型
- AI Agent
- 垂直领域应用
- 模型可解释性

深圳

AiCon 全球人工智能开发与应用大会

会议时间：8月22-23日

- 模型效率与部署
- 多模态大模型
- 大模型安全
- 智能硬件

北京

AiCon 全球人工智能开发与应用大会

会议时间：12月19-20日

- 通用大模型
- 智能硬件
- LMOps
- 具身智能

极客邦科技 2025 年会议规划

促进软件开发及相关领域知识与创新的传播

参会咨询

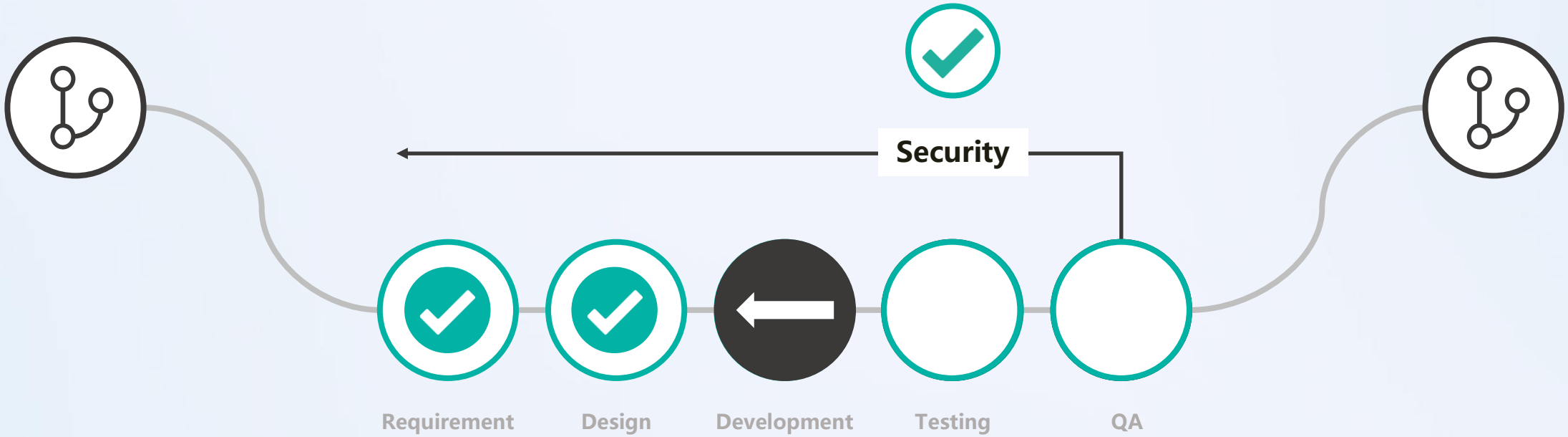
查看会议

01

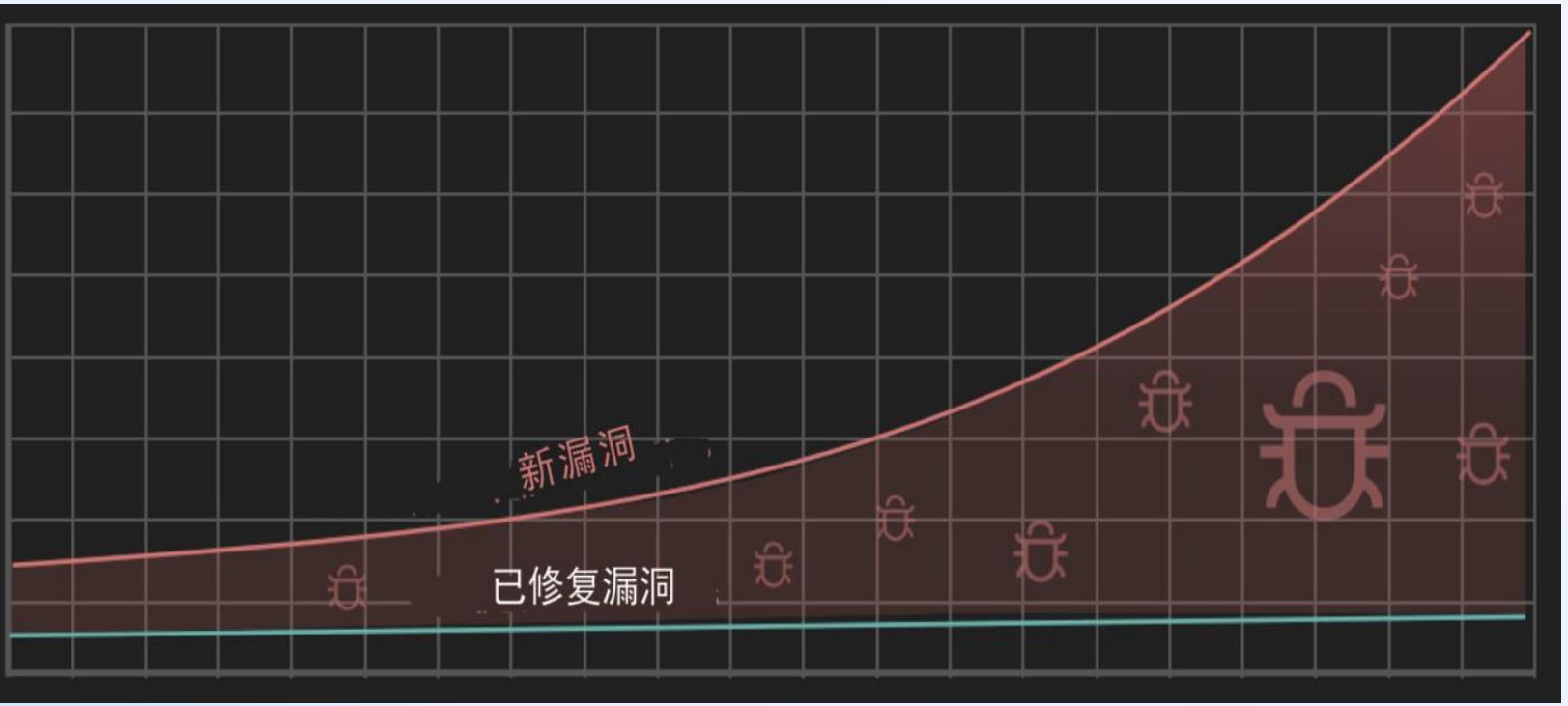
AI 重构 DevSecOps

趋势与角色

传统安全治理原则 — 安全左移



应用安全差距正在扩大



随着代码规模扩大，漏洞积压激增。
但开发人员的修复能效提升缓慢。

供应链漏洞总体趋势

年份	新CVE(开源组件类)	增长率	主要来源
2019	≈ 5 200	—	GitHub、Maven
2020	≈ 6 800	+31 %	NPM、PyPI
2021	≈ 8 500	+25 %	Log4j 等事件推动
2022	≈ 9 700	+14 %	PyPI 激增
2023	≈ 11 100	+14 %	供应链攻击增多
2024	≈ 13 600	+23 %	AI 组件漏洞增加
2025	≈ 15 000+	+10 – 12 %	AI 依赖和容器镜像

供应链漏洞总体趋势

漏洞类型	占比	示例
RCE / 代码执行	21 %	Apache Log4j、Fastjson 等
供应链投毒	18 %	npm event-stream、PyPI fake packages
信息泄露与配置错误	16 %	Spring 环境变量泄露
包版本污染	10 %	Python module 替换攻击
AI 代码库滥用漏洞	≈ 5 %	AI 模型加载、pickle 反序列化等

漏洞修复时延（Time to Remediate, TTR）

- 2019 年平均 TTR: 120 天
- 2025 年缩短至 80 天

安全工作让开发人员负担更重

250 工程师 x **500K** 平均年薪 x **5%** 安全投入

¥6.35M

附加的安全问题成本

传统的 AppSec 方法

清点资产	分诊	修复补救	治理
代码、开源、容器和应用程序接口的零散漏洞列表和清单	人工汇总 - 根据通用评分（如 CVSS）确定优先级。与工程部反复来回沟通	大量的修复工作与冲刺阶段的核心功能竞争。 没有“如何修复”或“这是否是破坏性改动”的指导	时间点式报告； 干扰或减缓 SDLC

打破传统的 AppSec 方法 -- 打破孤岛，简化工作流程。

清点资产

代码、开源、容器和应用程序接口的零散漏洞列表和清单

分诊

人工汇总 - 根据通用评分（如 CVSS）确定优先级。
与工程部反复来回沟通

修复补救

大量的修复工作与冲刺阶段的核心功能竞争。
没有“如何修复”或“这是否是破坏性改动”的指导

治理

时间点式报告；
干扰或减缓 SDLC

清点资产

一份针对代码 (SAST)、开放源代码 (SCA)、容器和应用程序接口的统一清单和漏洞列表

分诊

根据运行时执行情况、可到达性、可利用性和实际攻击面进行自动分流

修复补救

针对代码、直接和传递依赖关系的自助式“最佳修复位置”和“破坏性变更”指导

治理

执行从源代码到生产的持续策略，涵盖容器、操作系统和内存保护

02

“安全左移”和团队现实的差距和现实挑战

打破关于应用安全的一些误区

任何开源库集成都会引入超过70个额外的子依赖项。

简单性误区

应用安全仅靠静态扫描覆盖源码是不够的。应用运行时的依赖与配置层风险往往无人监控。

健全性误区

脱离上下文盲目信任准确性是一种谬误。超过90%的警报属于误报，纯粹是在制造噪音。

准确性误区

安全工具从不会“受到开发人员的喜爱“。工程人员欣赏的是准确性、深入的研究与专业的精神。

协作性误区

“安全左移“--现实挑战

1. 安全工具无缝集成在 CI/CD 管道中
2. 开发具备一定安全意识、自主修复漏洞
3. 安全检测实时反馈、开发快速修复
4. 安全与开发目标统一（质量 + 安全）
5. 各部门协同 DevSecOps 文化已形成

理想 VS 现实

1. 工具引入复杂、难以稳定运行，误报偏高
2. 安全培训不足，安全知识薄弱
3. 检测延迟、漏洞数量巨大、修复能力有限
4. 优先级冲突：业务上线速度 > 安全合规
5. 安全与开发仍是“两条线”，沟通壁垒严重

03

从“工具集合”走向“平台化的安全智能”

全栈分析 一打破信息孤岛

Kodem

鉴渊

SCA | SAST | Container Scanner | Secret Scanning | API Security | ADR

应用安全代理框架 **(Agentic Framework)**

发现	分诊	修复	报告	治理
代码 / 服务	实际风险评估，漏斗	计划	合规性	代码管理 策略
容器 / APIs	攻击者视角	简化	SBOM	CI 集成策略
漏洞		验证	按需定制	运行时 策略
违规 / 仓库		跟踪		

上下文可观察与修复引擎**(CORE)**

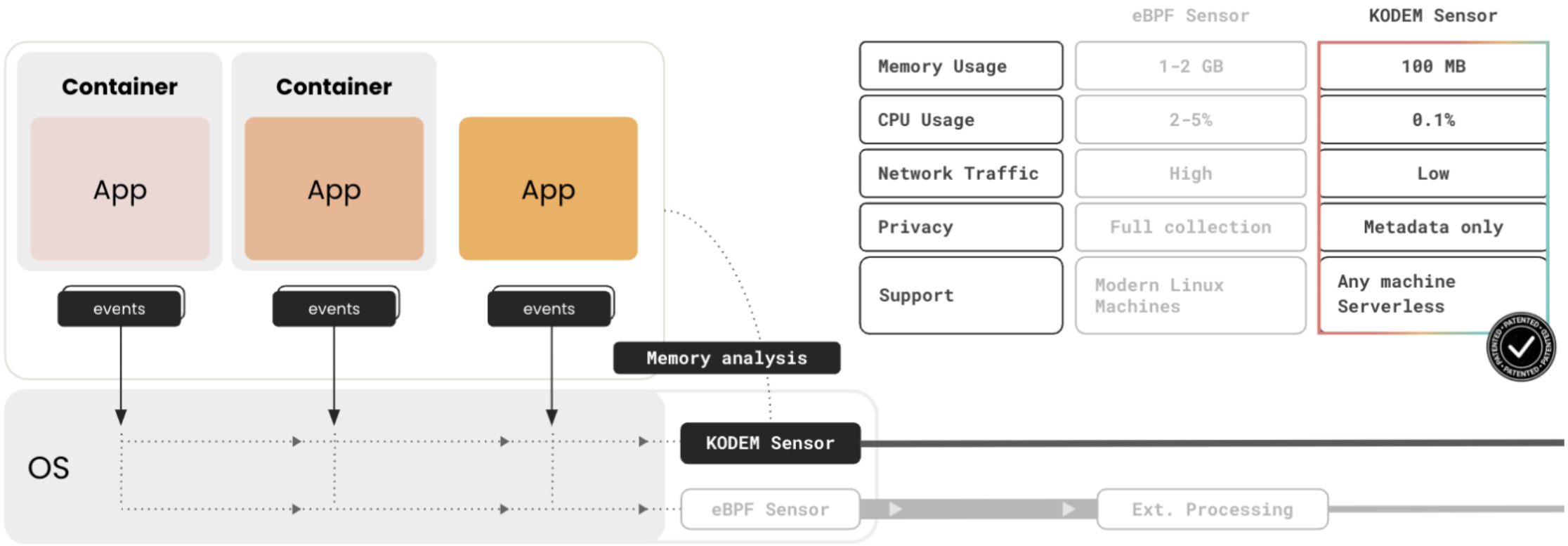
SCM 集成 | 镜像仓库 集成 | 运行时 集成

上下文可观察性 以及 补救修复引擎

方法步骤

1. 全面的依赖关系图分析，涵盖直接依赖关系和传递依赖关系。
2. **内存分析确认是否加载了有漏洞的代码/库。**
3. **全面覆盖应用层，包括操作系统依赖性和第三方依赖组件。**
4. 代码库与容器镜像之间的自动映射。
5. 为发现的每个风险制定补救计划。

运行时分析代理



	eBPF Sensor	KODEM Sensor
Memory Usage	1-2 GB	100 MB
CPU Usage	2-5%	0.1%
Network Traffic	High	Low
Privacy	Full collection	Metadata only
Support	Modern Linux Machines	Any machine Serverless

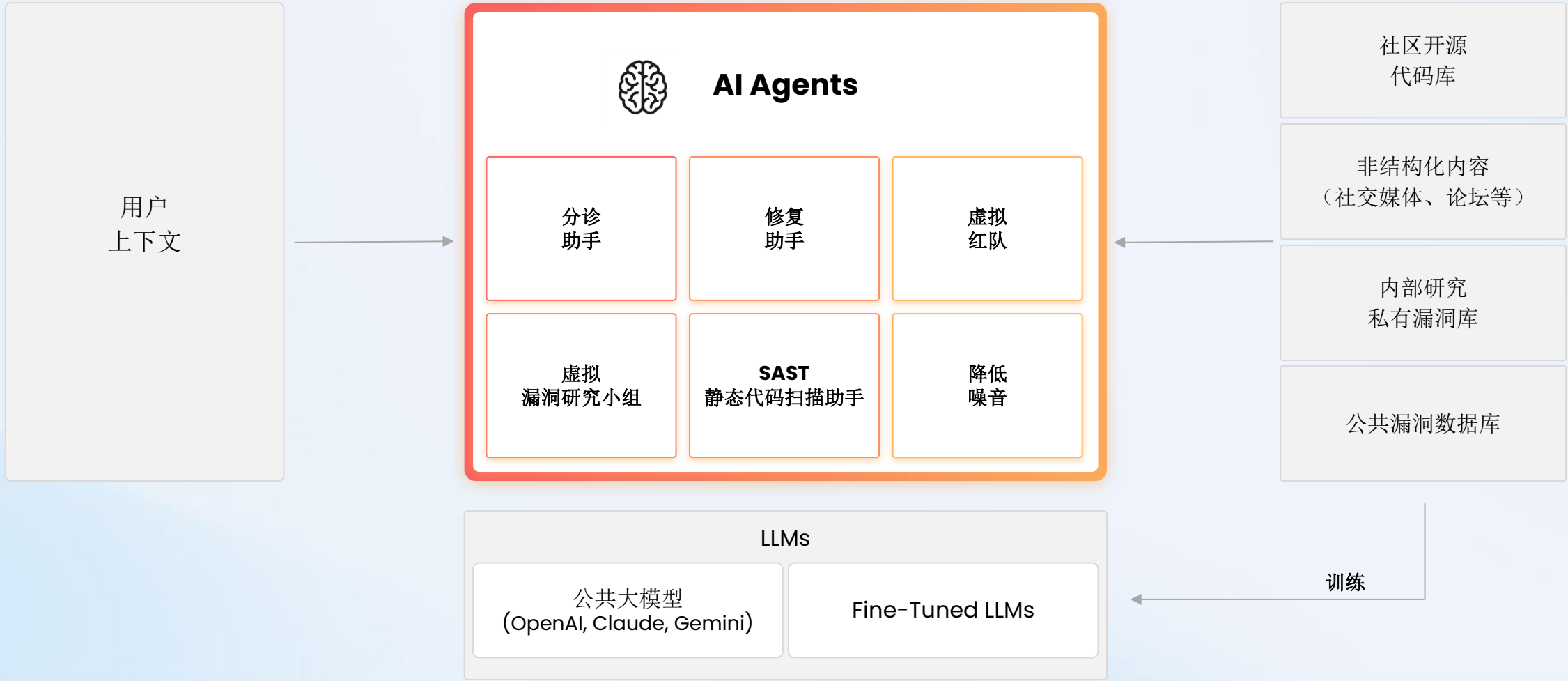
04

将 AI 集成到 DevSecOps 的最佳实践

践行“安全平铺“

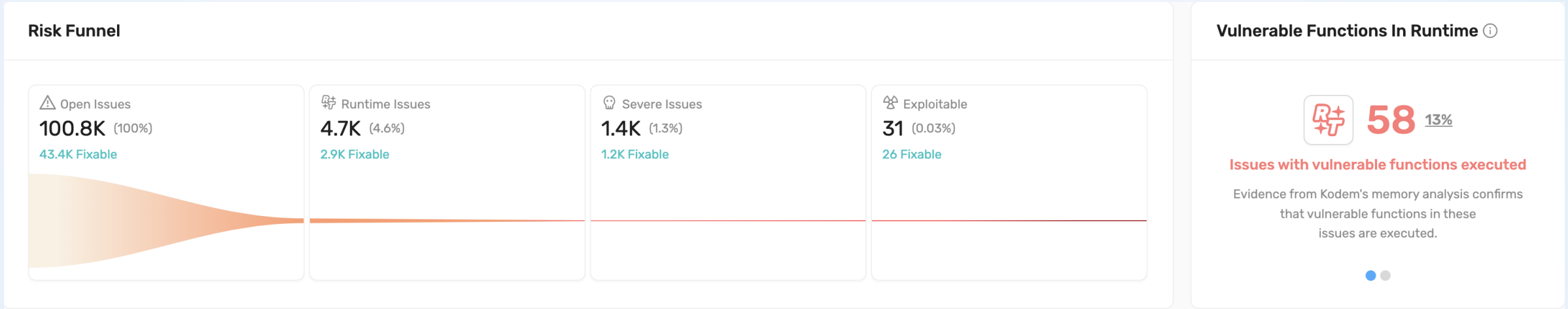
Kodem AI Agents -- 为安全治理赋能

以 AI 驱动的上下文化分析，实现对复杂攻击链的精准防御。

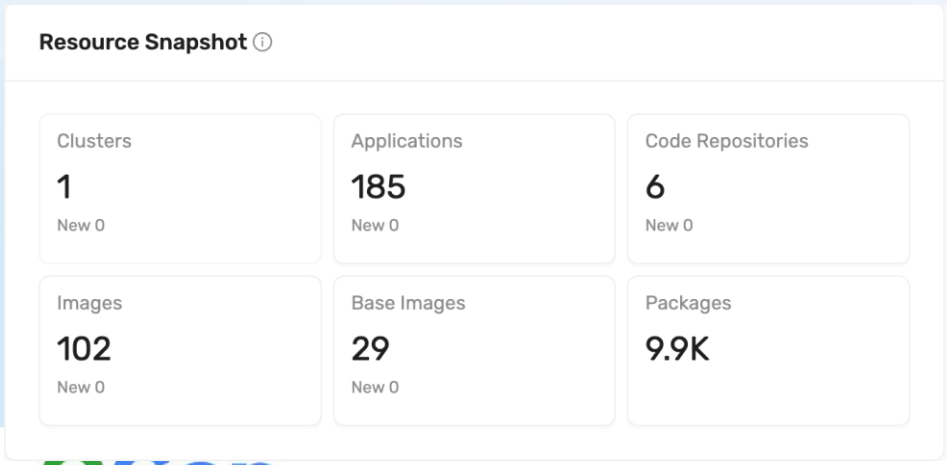


基于上下文的漏洞降噪

AI 自动分诊问题 – 整体降噪至4.6%, 高威胁降噪至 0.03%



覆盖所有应用资源



根据严重性、工作量、影响和可利用性评估漏洞

运行时--函数级分析

确定易受影响的功能执行情况（**function level**）

Code Injection

CVE-2022-22965 on spring-beans

Create Ticket | Dismiss | Label | Select Issue

Overview | Runtime Evidence

Key Evidence

Vulnerable Functions Executed

Executed Vulnerable Function:

- org/springframework/beans/CachedIntrospectionResults: <init>

Detected On Image:

docker.io/qienda/vuln-sca-test-backend:latest

Last time runtime evidence was observed in your environment: Oct 21, 2025 12:14 PM

Image: docker.io/qienda/vuln-sca-test-backend:latest

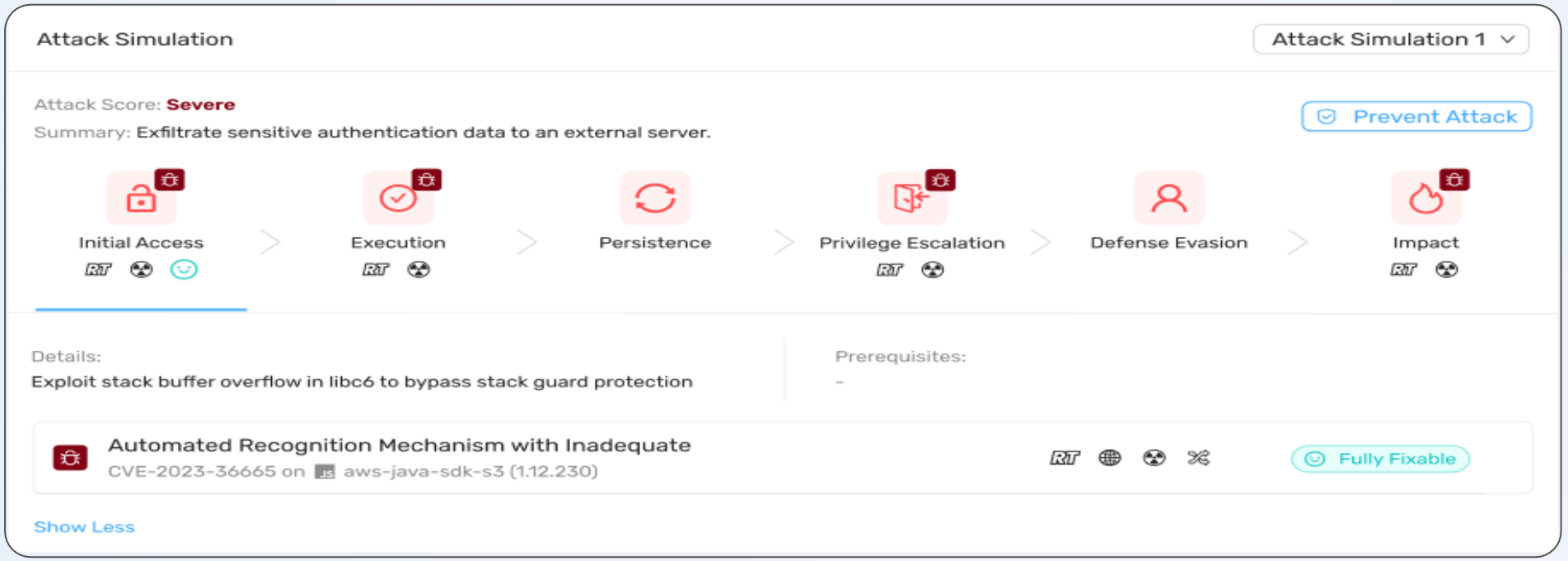
Observed In App: sca-backend-new | Environment: dev | Namespace: kodem-demo

Runtime Evidence


```
  "versions": [
      "5.1.14.RELEASE"
  ],
  "language": "java",
  "issueInRuntime": true,
  "lastObserved": "2025-10-21T04:14:03Z",
  "firstObserved": "2025-10-18T23:15:39Z",
  "functionsInRuntime": [
      "org/springframework/beans/factory/support/AbstractBeanDefinition: setDestroyMethodName",
      "org/springframework/beans/factory/support/AbstractBeanDefinition: isSingleton",
      "org/springframework/beans/factory/support/AbstractBeanDefinition: setAbstract",
      "org/springframework/beans/factory/support/AbstractBeanDefinition: <init>"
  ],
  "vulnerableFunctionsInRuntime": {
      "org/springframework/beans/CachedIntrospectionResults: <init>": [
          "CVE-2022-22965"
      ]
  },
  "executedClasses": [
      "org/springframework/beans/factory/config",
      "org/springframework/beans/factory/parsing",
      "org/springframework/beans/factory",
      "org/springframework/beans/factory/annotation",
      "org/springframework/beans/factory/support"
  ],
  "executedVulnerableClasses": {
      "org/springframework/beans/CachedIntrospectionResults": [
          "CVE-2022-22965"
      ]
  }
}
```

攻击连分析方法论

AppSec 防御 ≠ 简单检测漏洞



分析探测攻击场景和意图，而不只是一堆漏洞

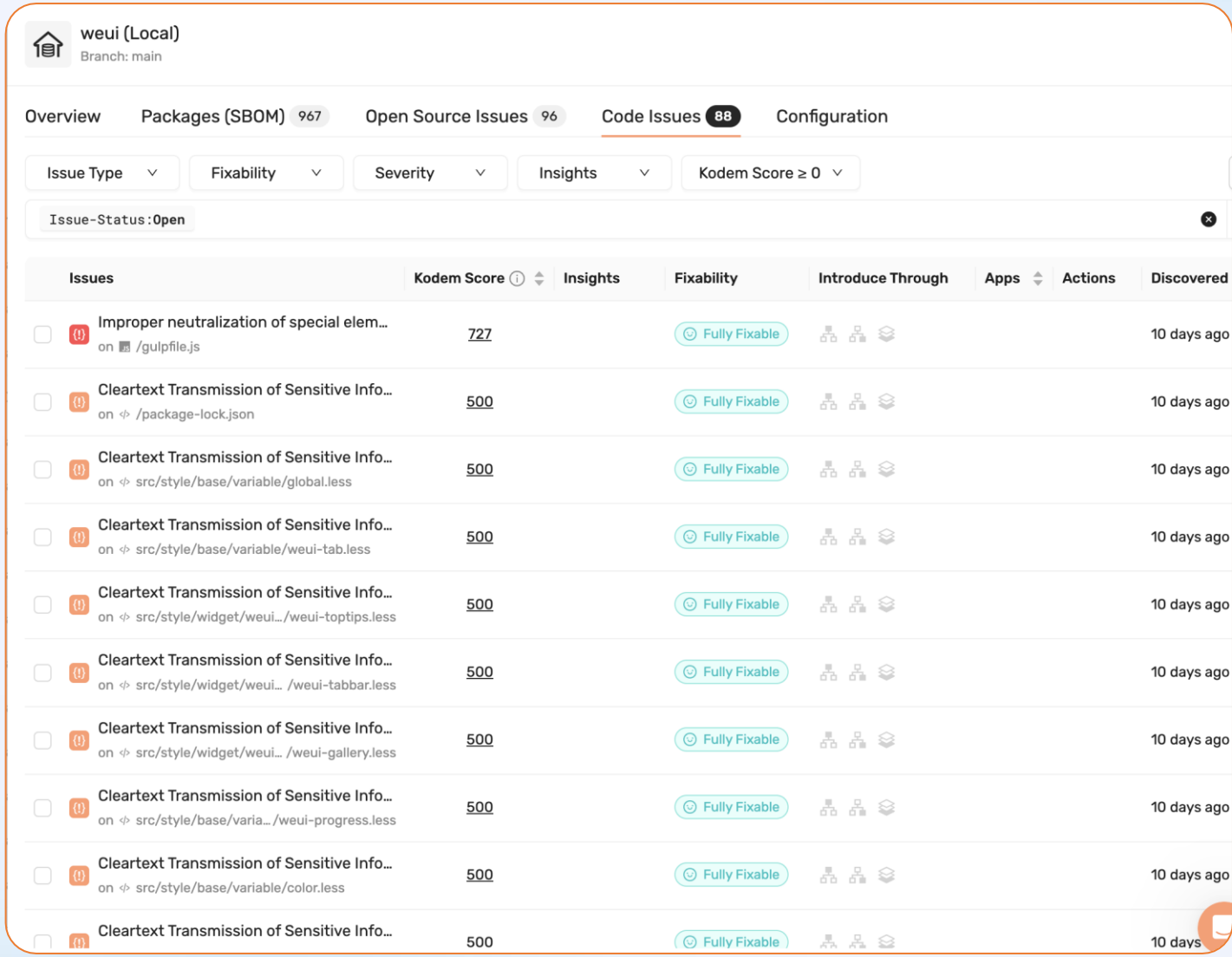
专注实际的攻击面，并提供攻击者视角，重现红队战役。

SAST 源代码扫描

范围：

生成代码库中以下信息：

- 代码仓库信息
- 代码漏洞 Code Weakness
- 代码敏感信息 Exposed Secret

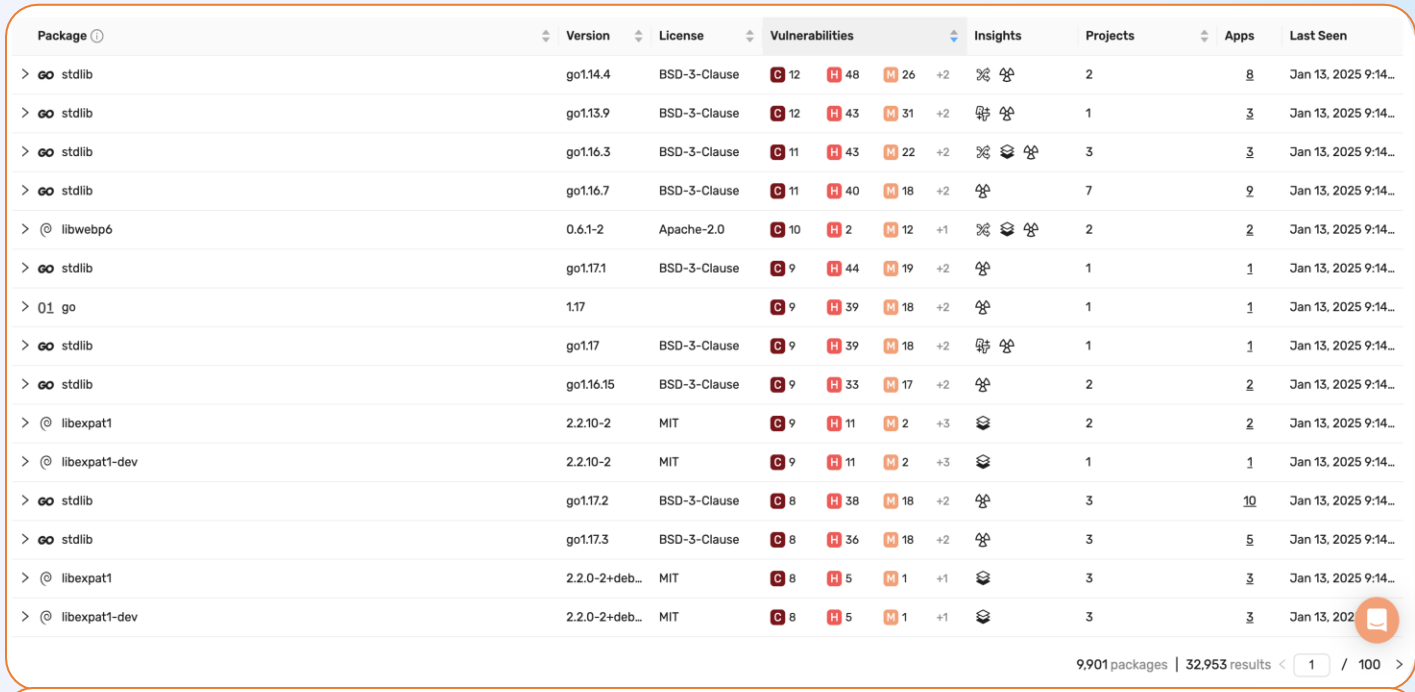


SCA 组件扫描

范围：

生成代码库中以下信息：

- 代码仓库信息
- 代码仓库SBOM 清单，
- 代码仓库三方漏洞清单
- 代码仓库三方License清单



Package	Version	License	Vulnerabilities	Insights	Projects	Apps	Last Seen
stdlib	go1.14.4	BSD-3-Clause	C 12 H 48 M 26 +2		2	8	Jan 13, 2025 9:14...
stdlib	go1.13.9	BSD-3-Clause	C 12 H 43 M 31 +2		1	3	Jan 13, 2025 9:14...
stdlib	go1.16.3	BSD-3-Clause	C 11 H 43 M 22 +2		3	3	Jan 13, 2025 9:14...
stdlib	go1.16.7	BSD-3-Clause	C 11 H 40 M 18 +2		7	9	Jan 13, 2025 9:14...
libwebp6	0.6.1-2	Apache-2.0	C 10 H 2 M 12 +1		2	2	Jan 13, 2025 9:14...
stdlib	go1.17.1	BSD-3-Clause	C 9 H 44 M 19 +2		1	1	Jan 13, 2025 9:14...
go	1.17		C 9 H 39 M 18 +2		1	1	Jan 13, 2025 9:14...
stdlib	go1.17	BSD-3-Clause	C 9 H 39 M 18 +2		1	1	Jan 13, 2025 9:14...
stdlib	go1.16.15	BSD-3-Clause	C 9 H 33 M 17 +2		2	2	Jan 13, 2025 9:14...
libexpat1	2.2.10-2	MIT	C 9 H 11 M 2 +3		2	2	Jan 13, 2025 9:14...
libexpat1-dev	2.2.10-2	MIT	C 9 H 11 M 2 +3		1	1	Jan 13, 2025 9:14...
stdlib	go1.17.2	BSD-3-Clause	C 8 H 38 M 18 +2		3	10	Jan 13, 2025 9:14...
stdlib	go1.17.3	BSD-3-Clause	C 8 H 36 M 18 +2		3	5	Jan 13, 2025 9:14...
libexpat1	2.2.0-2+deb...	MIT	C 8 H 5 M 1 +1		3	3	Jan 13, 2025 9:14...
libexpat1-dev	2.2.0-2+deb...	MIT	C 8 H 5 M 1 +1		3	3	Jan 13, 202

9,901 packages | 32,953 results 1 / 100

Introduced Through

Explore the resources below to see how the package is introduced to your environment. All listed resources have the same issue in the same project.

Resources

qienda/vuln-sca-test-backend
Tag: latest

Package Location:

- /root/.m2/repository/org/springframework/spring-beans/5.1.14.RELEASE/spring-beans-5.1.14.RELEASE.jar

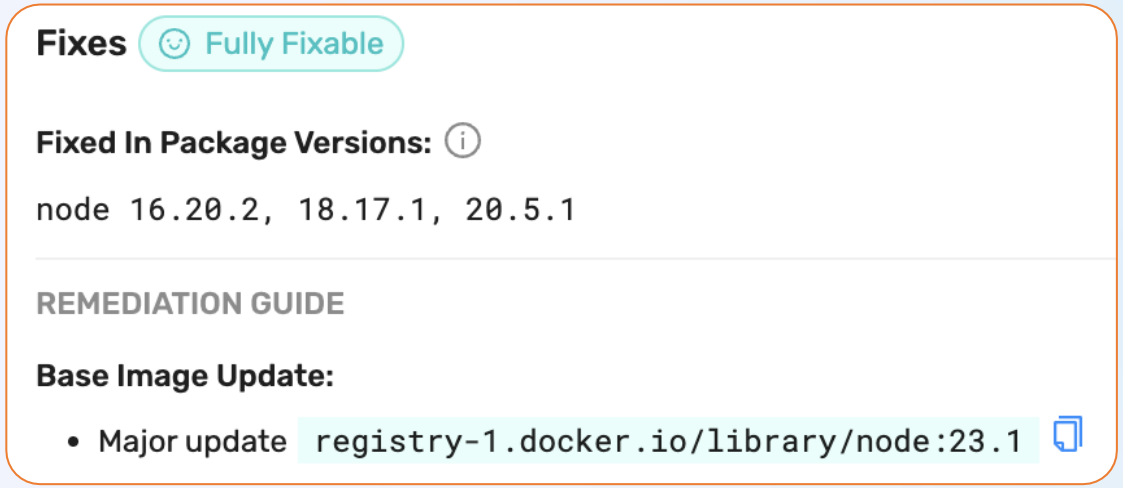
Docker History Commands

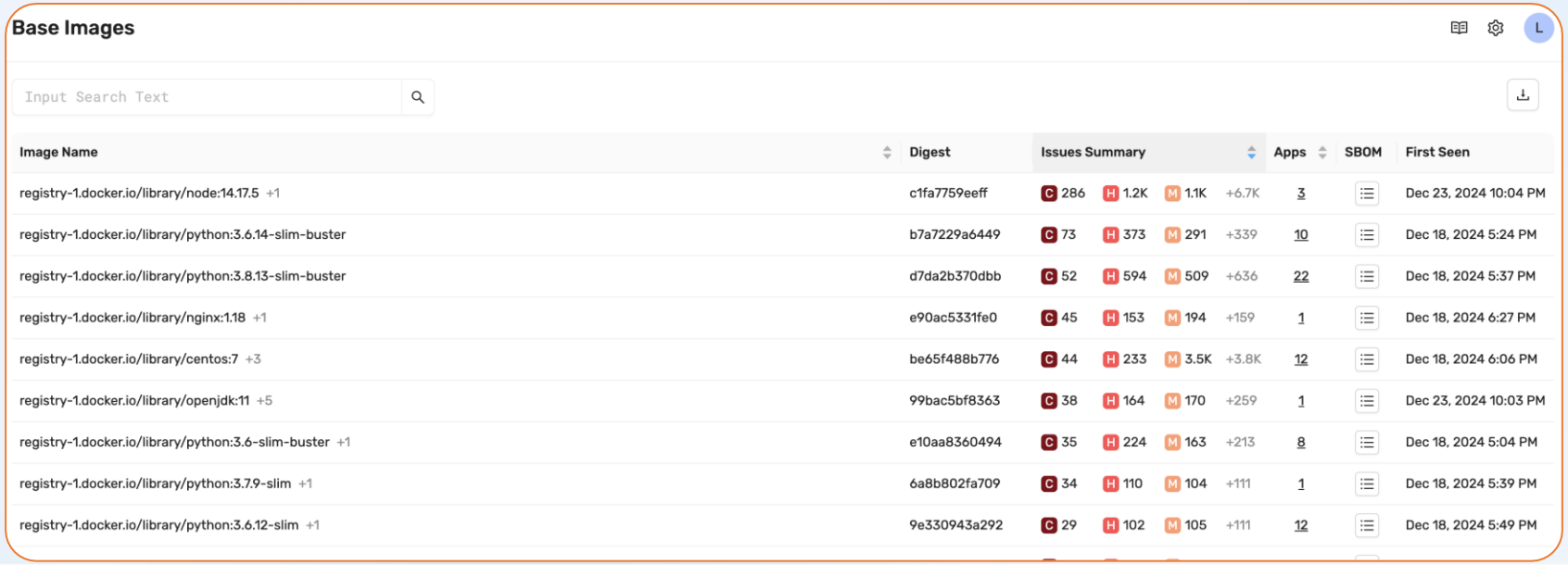
```
# History layer number: 16
RUN /bin/sh -c mvn clean install
```

基础镜像分析

范围：

- 确定问题是否来自基础镜像，并识别基础镜像层次结构
- 为公共基础图像修复提供修复建议





AI 驱动的 DevSecOps 智能安全中枢

分级助手
自动化地对关键漏洞进行优先级排序

漏洞降噪
过滤无关告警以提升分析专注度

修复助手
提供可执行且与上下文相关的修复操作

虚拟红队
模拟潜在攻击场景以识别系统弱点

Kodem Kai

AI 驱动的 DevSecOps 智能安全中枢

智能风险识别

- 基于大模型深入理解代码语义，自动识别漏洞与逻辑缺陷。
- 无需人工介入即可检测潜在的依赖风险与供应链隐患。
- 在开发阶段即时提示问题，避免漏洞流入生产环境。

主动预警

自动优先级与分流

- 综合评估漏洞的严重程度、影响范围与可利用性。
- 自动执行风险分流，帮助团队聚焦高价值修复。
- 动态调整策略，保证安全与效率的平衡。

精准

智能修复与建议

- 生成可直接使用的修复建议代码段。
- 结合上下文解释风险来源与修复原理。
- 支持多种语言与框架，快速适配开发环境。

自动修复

安全平铺与持续学习

- 将安全能力嵌入 CI/CD，全流程可追踪。
- Kai 持续学习历史修复案例，不断优化检测精度。
- 在不同环境中自适应安全策略，避免重复误报。

平铺

InfoQ 极客传媒

北京

QCon 全球软件开发大会

会议时间：4月10-12日

- 大模型赋能 AIOps
- 越挫越勇的大前端
- 云上业务架构演进
- 多模态大模型及应用
- 海外 AI 应用创新实践

北京

AiCon 全球人工智能开发与应用大会

会议时间：6月27-28日

- 端侧智能
- AI Agent
- 多模态大模型
- 金融+大模型

上海

QCon 全球软件开发大会

会议时间：10月23-25日

- AI Agent
- 大模型训练推理
- 端侧 AI
- 搜推深度融合
- 数智金融

4月 — 5月 — 6月 — 8月 — 10月 — 12月

上海

AiCon 全球人工智能开发与应用大会

会议时间：5月23-24日

- 通用大模型
- AI Agent
- 垂直领域应用
- 模型可解释性

深圳

AiCon 全球人工智能开发与应用大会

会议时间：8月22-23日

- 模型效率与部署
- 多模态大模型
- 大模型安全
- 智能硬件

北京

AiCon 全球人工智能开发与应用大会

会议时间：12月19-20日

- 通用大模型
- 智能硬件
- LMOps
- 具身智能

极客邦科技 2025 年会议规划

促进软件开发及相关领域知识与创新的传播

参会咨询

查看会议

QCon
全球软件开发大会

InfoQ 极客传媒

THANKS

大模型正在重新定义软件

Large Language Model Is Redefining The Software