

# Использование инструментов композиционного анализа

29.10.2025



# План вебинара

- 0 Введение и предпосылки
- 1 Композиционный анализ
- 2 ГОСТ 56939-2024
- 3 Демо: применение в IDE, конвейере и проведение регулярного анализа



# О компании

Создали **первое российское решение** для обеспечения безопасной работы с open source

1

**5 лет** реализуем свой продукт на российском рынке

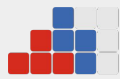
2

**> 10 лет** анализируем исходные коды на качество и безопасность

3

**> 50 экспертов**, чья работа была отмечена ИСП РАН и ФСТЭК России

Участники ассоциаций



АРПП  
Отечественный софт

Мы в едином реестре ПО [№13008](#)





# Нам доверяют



Более 60 компаний из разных отраслей уже внедрили CodeScoring  
Более 40 000 программистов, чья работа стала безопаснее

Финтех

Сырьевые  
и энергетические  
компании

Телеком и медиа

Ритейл

ИТ/ИБ-компании  
и интеграторы

OZON



самолет

билайн

Ростелеком



НПЦ КСБ



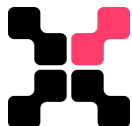
ДИКСИ



# Проверка безопасности и качества вашего ПО



**CodeScoring.OSA** ваш фаервол от вредоносного и уязвимого Open Source



**CodeScoring.SCA** ваш помощник безопасности и лицензионной чистоты Open Source на всех этапах разработки



**CodeScoring.TOI** ваш помощник в анализе качества собственного кода в контексте команд разработки



**CodeScoring.Secrets** эффективная работа с поиском секретов в коде с применением машинного обучения

*CodeScoring это модульная платформа:  
установка on-premise, интеграция  
с ASPM, task-трекерами, почтой,  
айдентити провайдерами, сервисами  
безопасности, и др.*



The background is a solid pink color with a series of thin, dark pink diagonal lines running from the top-left towards the bottom-right, creating a subtle geometric pattern.

# Open Source



# Чаще всего ваш продукт СОСТОИТ ИЗ:

## Сторонний код

~80% заимствованных компонентов из открытых источников

Известен

Идентифицируем

Исследуем

## Собственный код

~20% собственного кода

Неизвестен

Нужно сканировать

Сложно исследовать



# Немного статистики про сторонний код

> 250 млн

проектов с открытым  
исходным кодом, а также:  
8 млн. готовых пакетов;  
100 млн. их версий

~90 млн

разработчиков хотя бы раз  
поучаствовали в разработке, а  
регулярно участвует ~6 млн.

protestware

саботаж и закладки  
Пакеты: es5-ext, node-ipc,  
colors, faker, и ещё немного.  
И Палестина.

x3

увеличилась скачиваемость  
открытых компонентов  
с 2023 на 2024

x13

выросло количество известных  
атак на цепочки поставки:  
Dependency Confusion,  
Typosquatting, Namesquatting,  
Brandjacking, Malicious Code  
Injection и др.

> 1000/мес

вредоносных пакетов  
выявляется экспертами  
по безопасности каждый месяц:  
кража параметров окружения,  
бэкдоры, шифровальщики и др.



# Open Source уязвим



## Уязвимости

Обнаруживаются каждый день.  
На многие уязвимости есть  
эксплоит на GitHub.  
Время подготовки атаки  
значительно меньше.  
Обостряется проблема  
закладок и вредоносного кода.

## Лицензии

Их отсутствие,  
лицензионная (не)  
совместимость, риск смены  
лицензии на более строгую,  
риск введения экспортных  
ограничений.

## Качество

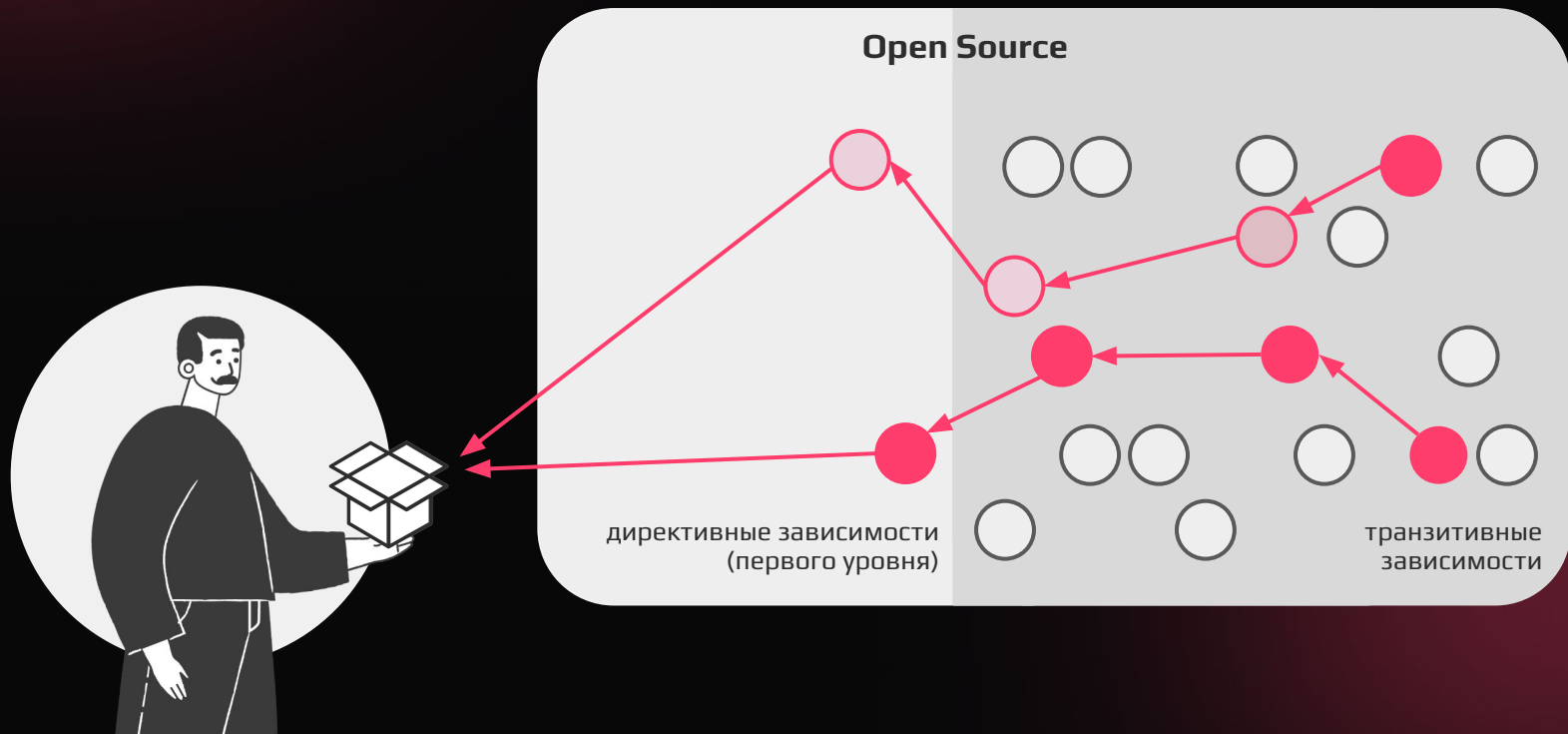
Ошибки, отсутствие тестов,  
заглушки  
функциональности  
(до 30%) и зачастую  
брошенные проекты.

Но есть решения.



# Проблема глубже чем кажется

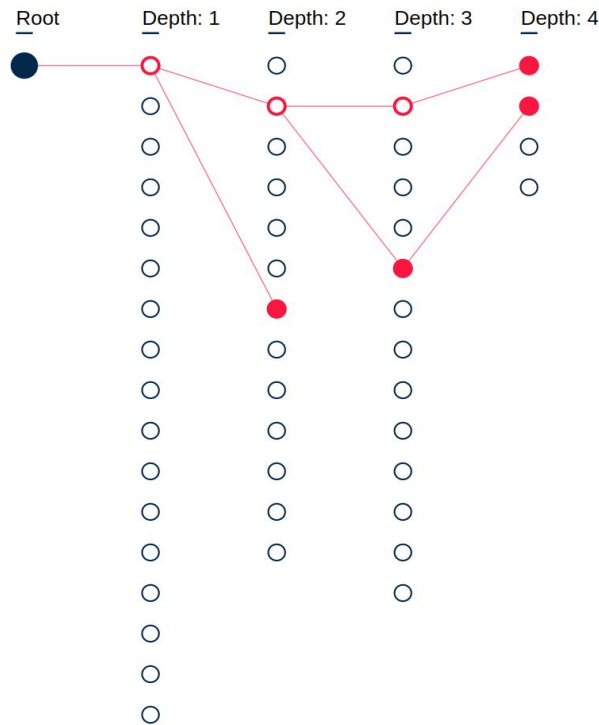
Проверка заимствованных компонентов и понимание состава программного продукта — ключевой аспект безопасной разработки.





# Транзитивные зависимости (уязвимости)

Транзитивная зависимость —  
зависимость ваших зависимостей.



CodeScoring dependency graph



# Как разработка разрабывает

Разработчики очень по-разному могут работать со сторонним кодом

- Зависимости не фиксируются
- Фиксируются, включением в код 😂
- Фиксируются, но в \*.txt-файле (иногда не все)
- Фиксируются, без разделения окружений сборки
- Фиксируются, но без версий
- Фиксируются, но без фиксированной версии
- Фиксируются, но без транзитивности
- Фиксируются, но без хэш-сумм
- Фиксируются, но не соотносятся с релизами продуктов
- Фиксируются, но не сохраняются (не кэшируются)
- и т. д.





The background is a solid pink color with a series of thin, dark pink diagonal lines running from the top-left towards the bottom-right, creating a sense of motion or depth.

# Кейсы



# Червь Shai-Hulud vs NPM

- установка пакета
- запуск локального поиска секретов TruffleHog
- публикация найденных секретов на гитхаб
- обнаружение ваших npm-пакетов
- дописывание себя в пакеты и выпуск новых версий

Затронуто более 500 проектов в npm.



Кадр из Дюны



# Уязвимости «из новостей»



# log4j

**CVE-2021-44228, CVE-2021-45046.**

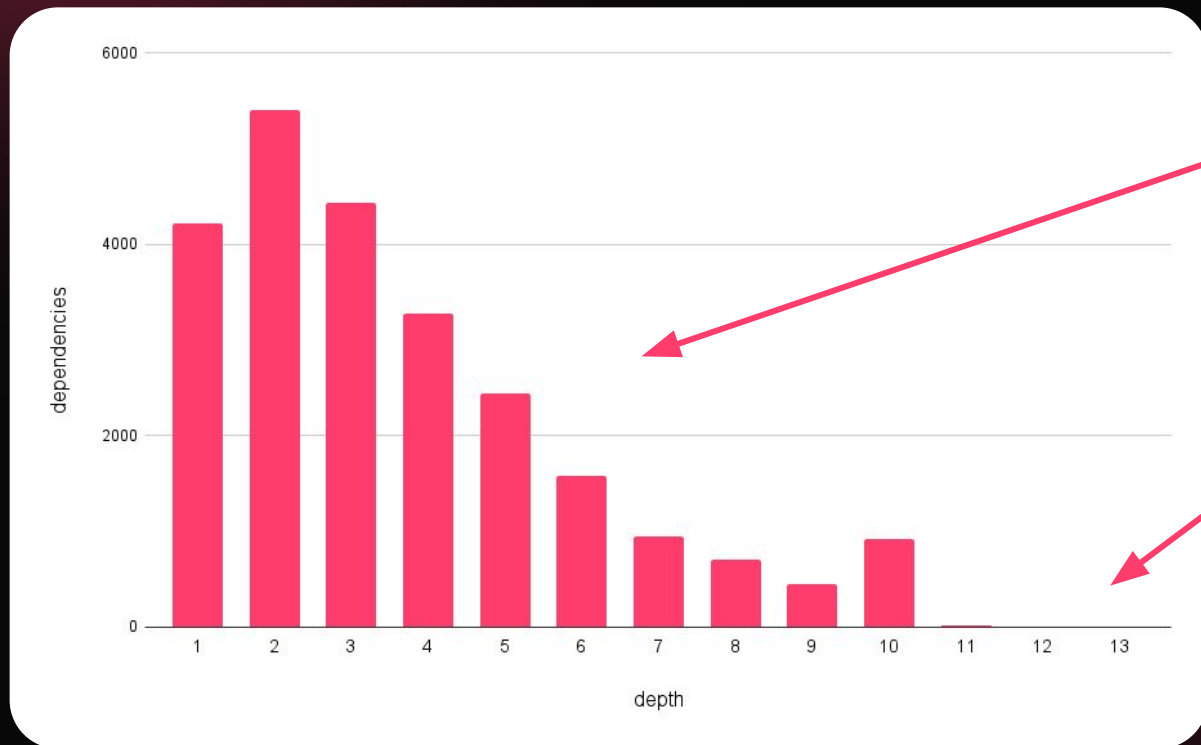
Выполнение произвольного кода на сервере  
(Arbitrary Code Execution, ACE).

**Log4shell затронула 8% всей экосистемы Java-пакетов.**

**Возможно защититься при помощи Application Firewall.**



# О глубинах залегания транзитивных проблем (log4shell, затронула 8% mvn-экосистемы)



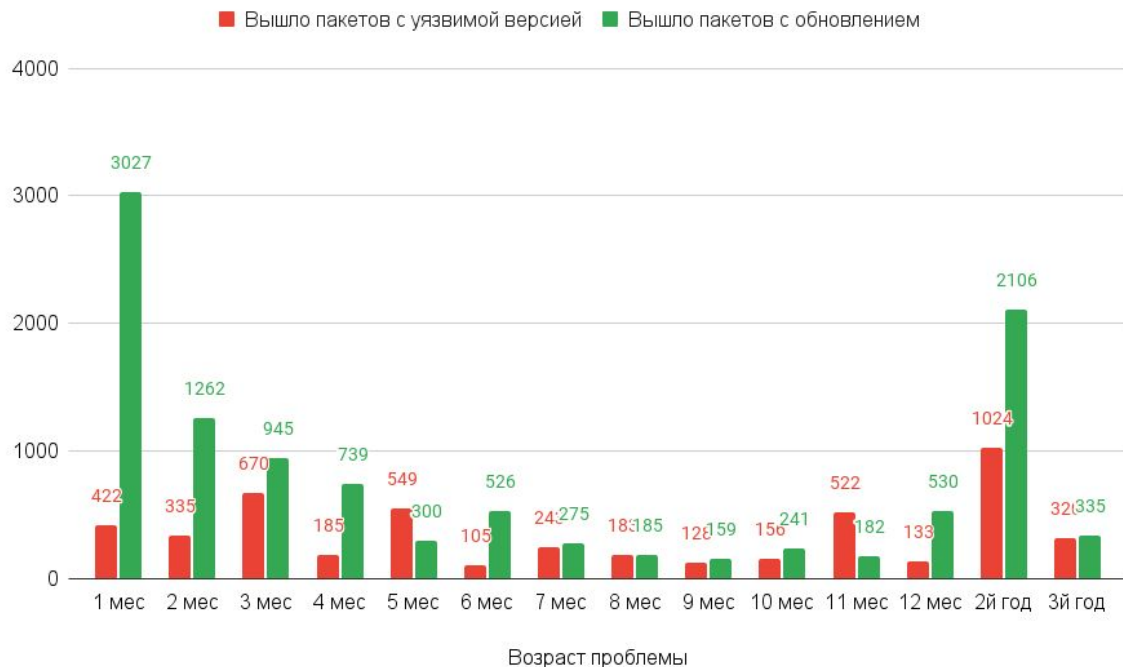
Эксплуатация  
возможна  
и здесь

Тут тоже  
есть!



# И проблема со временем не уходит

Переход сообщества на безопасный log4j-core  
Окружения: compile, runtime, provided





# Уязвимости не из новостей



# pillow

CVE-2022-22817, CVE-2022-22816, CVE-2022-22815, CVE-2022-30595, и пр.: раскрытие конфиденциальной информации, запись за границами буфера в памяти, выполнение произвольного кода, отказ в обслуживании.

**Затронуло 4% всей экосистемы Python-пакетов (по оценкам аналитиков команды CodeScoring). Невозможно защититься при помощи Application Firewall.**



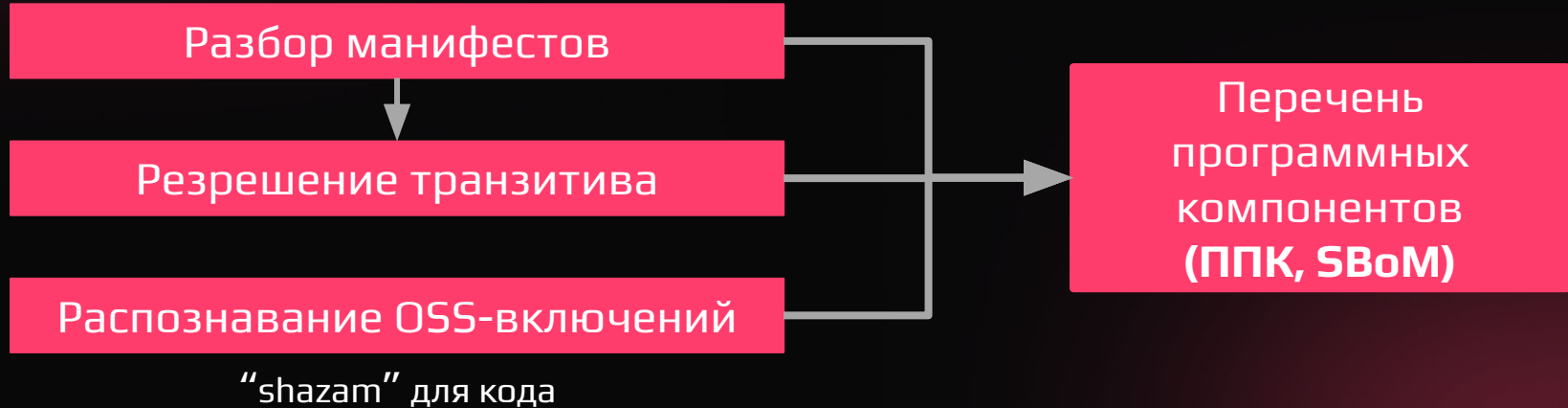
The background is a solid pink color with a series of thin, dark pink diagonal lines running from the top-left towards the bottom-right, creating a subtle geometric pattern.

# Стандартизация



# Инвентаризация ПО!

Найти манифесты, разобрать их, идентифицировать компоненты.  
Нужно не забыть про «включения» Open Source в ваш код.





# Перечень программных компонентов (ППК)



Машиночитаемый документ, содержащий в себе структурированную информацию о сторонних компонентах программного обеспечения и отношениях между ними.

SBoM, Software bill of materials.



# Основные форматы SBOM

## SPDX

The Software Package Data Exchange

- ❑ Становление: 2010 г.
- ❑ Linux Foundation
- ❑ Стандарт  
ISO/IEC 5926:2021
- ❑ Изначально ориентирован  
на работу с лицензиями,  
безопасность появилась  
позже
- ❑ [spdx.dev](https://spdx.dev)

## CycloneDX

- ❑ Становление: 2017 г.
- ❑ OWASP, ECMA (TC54),  
TK362 ФСТЭК России
- ❑ Стандарт: ECMA-424
- ❑ Есть развитие: SaaSOM,  
SBOM, ML-BOM, OBOM, ...
- ❑ Учтены огрехи  
предшественников,  
поддержка в сообществе  
и среди инструментов
- ❑ [cyclonedx.org](https://cyclonedx.org)



# Перечень программных компонентов (SBoM)

Metadata	Supplier	Authors	Component		
	Manufacturer	Tools	Lifecycles		
Components	Supplier	Identity	Pedigree	Provenance	Evidence
	Component Type	Licenses	Hashes	Release Notes	Relationships
Services	Provider	Data Classification	Trust Zone		
	Endpoints	Data Flow	Relationships		
Dependencies	Components	Services			
Compositions	Completeness of:				
	Components	Services	Dependencies		
Vulnerabilities	Details	Source	Exploitability	Targets Affected	
	Advisories	Risk Ratings	Evidence	Version Ranges	
Formulation	Declared	Formulas	Tasks	Components	
	Observed	Workflows	Steps	Services	
Annotations	Per Person	Per Organization	Per Tool		
	Details	Timestamp	Signature		
Extensions	Properties	Per Organization	Per Team		
	Formal Taxonomy	Per Industry	...		

Содержит необходимую информацию о составе ваших продуктов.

Объектная модель одного из стандартов обмена данными: CycloneDX.

Ещё есть SPDX, etc.



# Открытые инструменты SCA

[cli] Агенты-анализаторы:

- Trivy /[trivy.dev](https://trivy.dev)
- cdxgen /[github.com/CycloneDX/cdxgen](https://github.com/CycloneDX/cdxgen)
- dep-scan /[github.com/owasp-dep-scan/dep-scan](https://github.com/owasp-dep-scan/dep-scan)
- osv-scanner /[github.com/google/osv-scanner](https://github.com/google/osv-scanner)
- Syft/[github.com/anchore/syft](https://github.com/anchore/syft)
- Иные: OWASP tool Center /[cyclonedx.org/tool-center](https://cyclonedx.org/tool-center)

[web] Платформы с пользовательским интерфейсом:

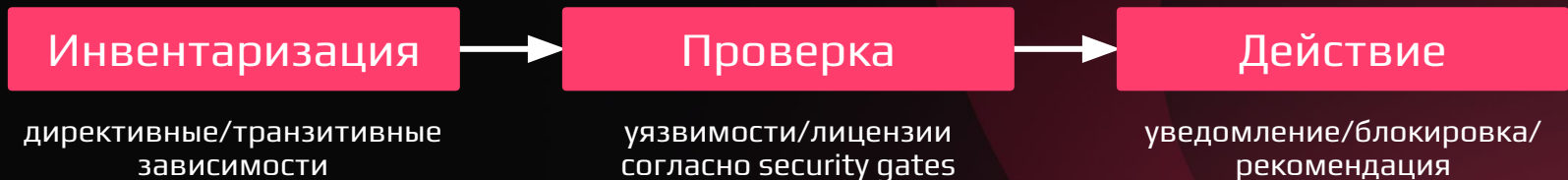
- Dependency Track /[dependencytrack.org](https://dependencytrack.org)



# Композиционный анализ

Цель согласно ГОСТ Р 56939-2024:

*«создание условий для снижения рисков наследования уязвимостей и недеklarированных возможностей при использовании заимствованного кода в коде ПО разработчика»*





# Композиционный анализ

Определение согласно проекту ГОСТа Композиционный анализ:

*«анализ, основанный на инвентаризации сторонних компонентов ПО, определении особенностей их использования, составлении перечня известных уязвимостей и/или иных недостатков компонентов»*



# Композиционный анализ /КАПО

Требования согласно **ГОСТ Р 56939-2024**:

- *разработка регламента композиционного анализа*
- *формирование перечня зависимостей ПО (ППК/SBOM)*
- *контролировать и актуализировать ППК*
- *проводить анализ компонентов находящихся на поверхности атаки*
- *применять корректирующие воздействия*



# КАПО. Разработка регламента

Артефакты реализации /ГОСТ Р 56939-2024:

- *обязанности сотрудников и их роли*
- *правила отслеживания уязвимостей в сторонних компонентах*
- *правила проведения анализа на наличие известных уязвимостей*
- *правила принятия компенсирующих и защитных мер*
- *периодичность проведения композиционного анализа*



# КАПО. Построение и актуализация ППК/SBOM

Артефакты реализации /ГОСТ Р 56939-2024:

- *процедура контроля и актуализации ППК*
- *описание инструментов контроля ППК*
- *ведение журналов результатов анализа*



# КАПО. Анализ компонентов на поверхности атаки

Определение /ГОСТ Р 56939-2024:

**поверхность атаки:** Множество подпрограмм (функций, модулей) программного обеспечения, обрабатывающих данные, поступающие посредством интерфейсов, напрямую или косвенно подверженных риску атаки.

[адаптировано из ГОСТ Р 56498-2015 п. 3.1.7]



# КАПО. Корректирующие воздействия

Артефакты реализации /ГОСТ Р 56939-2024:

Для закрытого ПО:

- *результаты анализа применимости уязвимости*
- *результаты обращения к разработчику*
- *результаты после применения обновлений*

Для открытого ПО:

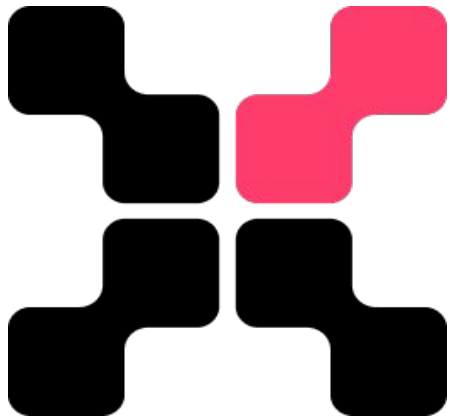
- *результаты анализа применимости уязвимости*
- *результаты действий по попыткам обновлений компонентов или применение собственных механизмов исправлений*



The background is a solid pink color with a series of thin, dark pink diagonal lines running from the top-left towards the bottom-right, creating a subtle geometric pattern.

# CodeScoring.SCA



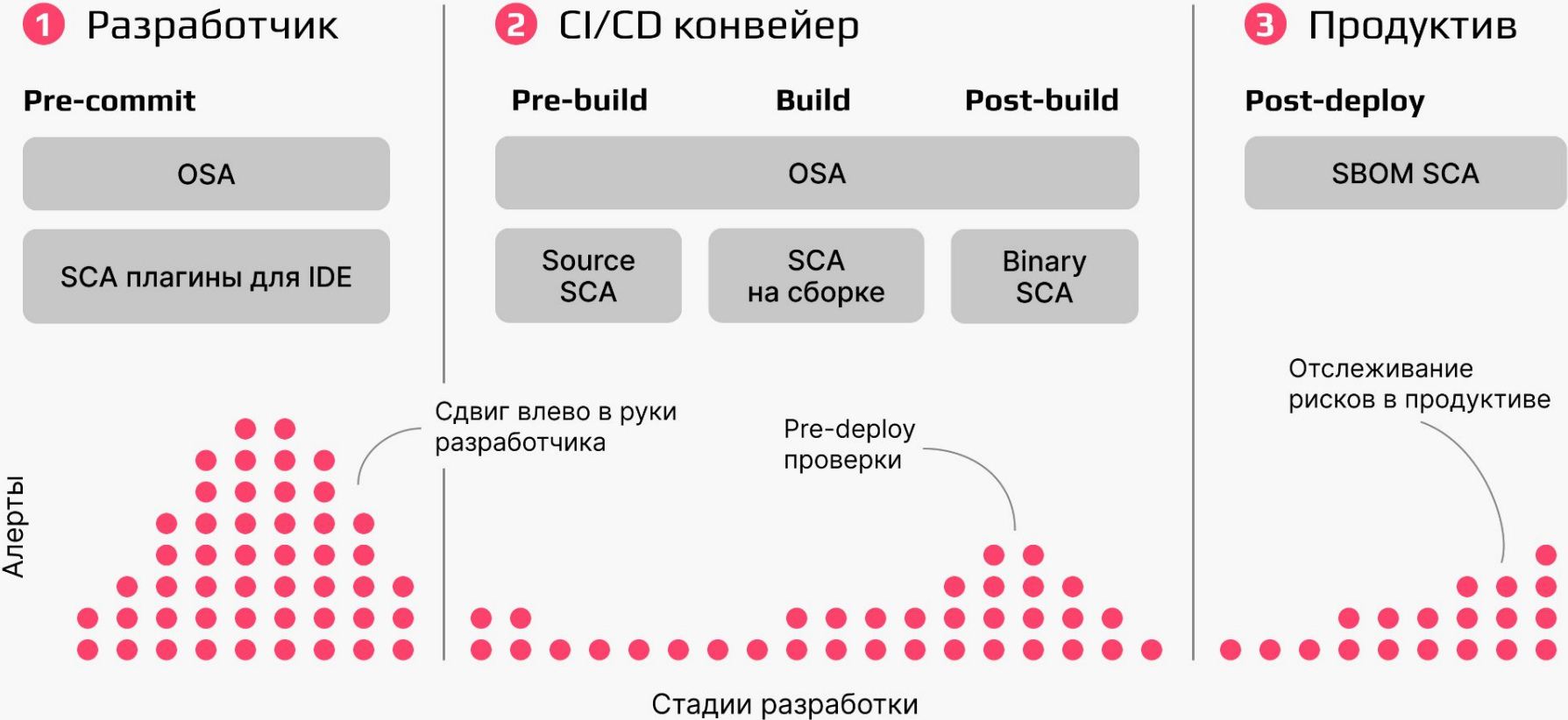


# CodeScoring.SCA

Композиционный анализ



# Безопасность цепочки поставки как процесс

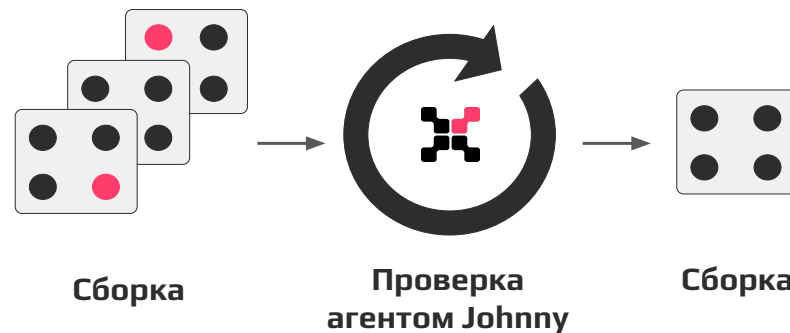




# Безопасность сборок в CI/CD



- Проверка сторонних компонентов на этапе сборки **универсальным агентом**
- Возможность **блокировки сборки** на основании политик безопасности
- Сохранение **перечня программных компонентов\*** (SBOM) и отчетов
- **Уведомления и рекомендации** по исправлению выявленных проблем



\* поддерживается стандарт CycloneDX, SPDX, а также расширенные требования ФСТЭК России к составу SBOM



# Интеграция в инструменты сборки



Jenkins

+оркестрация



GitLab



TeamCity



Bamboo



GitFlic

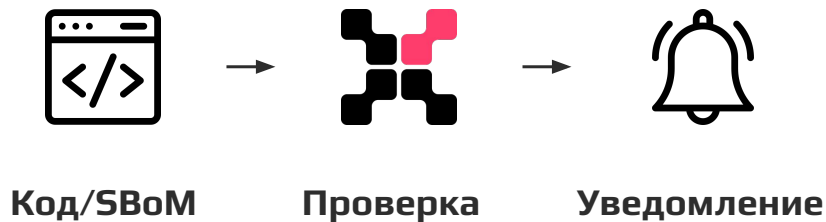
Агент Johnny интегрируется в любые инструменты сборки как **бинарная утилита** или **Docker-образ**



# Непрерывный мониторинг



- Сканирование **по расписанию** SBoM и git-репозиториев кода
- Уведомление о **новых рисках** в менеджер задач, почту или ASPM/ASOC/SIEM-системы
- Ведение **полной истории сканирования** и отчетов





# Интеграция с платформами разработки



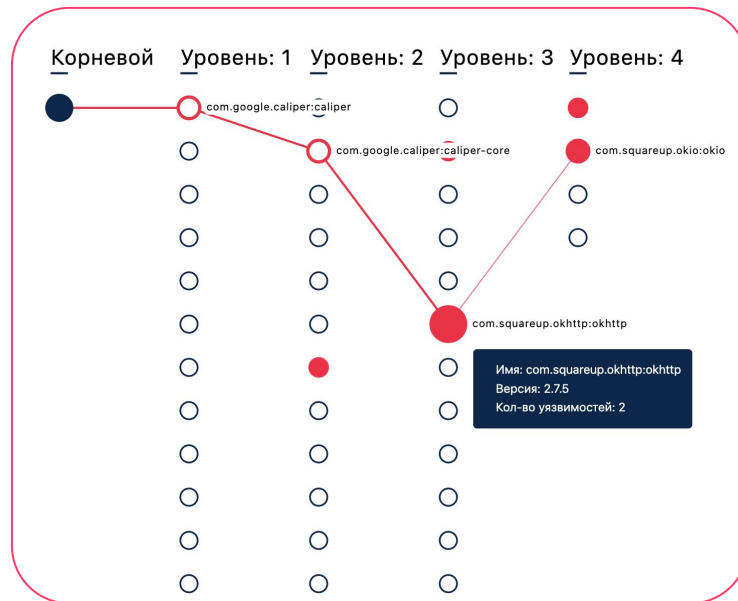
**CodeScoring SCA** поддерживает основные  
платформы, использующие **git**



# Методы инвентаризации



- **Автообнаружение** зависимостей
  - разбор манифестов (конфигураций)
  - разрешение транзитивных зависимостей в облаке и окружении сборки
  - определение OSS-включений (“shazam” для пакетов)
  - анализ сборки для языков C и C++
- Визуализация **графа зависимостей**
- **Отчеты** по всей организации и отдельным проектам

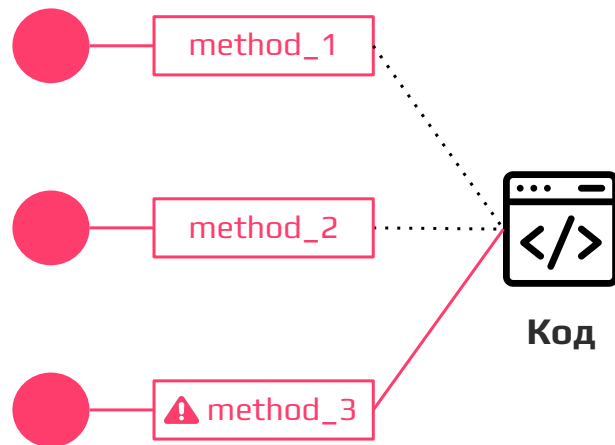




# Анализ **достижимости уязвимостей**



- Построение **графа вызовов** в проекте
- Определение путей к **уязвимым методам**
- Настройка **политики безопасности** на достижимость: блокировка/разрешение по факту использования
- **Снижение ручного анализа** найденных уязвимостей в **~10 раз\***





# База знаний про Open Source



- **Собственная база знаний** с метаданными компонентов
- Модерирование данных из **20+** источников знаний об угрозах и собственный фид **protestware**
- Единственная на рынке нативная интеграция с **Kaspersky OSS Threats Data Feed**
- **Дедупликация** уязвимостей из всех источников
- Автоматическое обновление

## CVE-2022-30123

Possible shell escape sequence injection vulnerability in Rack

CVE.ORG	<a href="#">CVE-2022-30123</a>
NVD	<a href="#">CVE-2022-30123</a>
GITHUB	<a href="#">GHSA-wq4h-7r42-5hrr</a>
OSV	<a href="#">CVE-2022-30123</a>
БДУ	<a href="#">BDU:2022-04201</a>
DEBIAN	<a href="#">CVE-2022-30123</a>
UBUNTU	<a href="#">CVE-2022-30123</a>
REDHAT	<a href="#">CVE-2022-30123</a>
CWE	<a href="#">CWE-150: Improper Neutralization of Escape, Meta, or Control Sequences</a> <a href="#">CWE-179: Incorrect Behavior Order: Early Validation</a>
Есть эксплойт	Нет
Дата публикации	06.12.2022
Дата обновления	09.12.2023



# Контекст угроз



- Определение **malware & protestware**
- **Классификация: CVSS 2 / CVSS 3**
- Информация об **эксплойтах** и патчах
- Рекомендации по **исправлению** уязвимостей

Уязвимость	Зависимость	Связь	Окружение	Проект	CVSS 2	CVSS3
CVE-2020-1747	pyyaml@5.1	Прямая	runtime	pypa/gh-action-pip-audit	10.0 HIGH	9.8 CRITICAL
CVE-2020-14343	pyyaml@5.1	Прямая	runtime	pypa/gh-action-pip-audit	10.0 HIGH	9.8 CRITICAL
CVE-2020-14343	pyyaml@5.3.1	Прямая	test.	zyfra/lebonite	10.0 HIGH	9.8 CRITICAL
CVE-2020-14343	pyyaml@5.1.2	Прямая	runtime	CycloneDX/cdxgen	10.0 HIGH	9.8 CRITICAL
CVE-2020-1747	pyyaml@5.1.2	Прямая	runtime	CycloneDX/cdxgen	10.0 HIGH	9.8 CRITICAL
CVE-2022-30123	rack@2.0.7	Транзитивная	unknown	CycloneDX/cdxgen	10.0 HIGH	10.0 CRITICAL
CVE-2023-40175	puma@4.2.1	Прямая	unknown	CycloneDX/cdxgen	10.0 HIGH	9.8 CRITICAL
CVE-2020-1747	pyyaml@5.2	Прямая	develop	CycloneDX/cdxgen	10.0 HIGH	9.8 CRITICAL
CVE-2020-14343	pyyaml@5.2	Прямая	develop	CycloneDX/cdxgen	10.0 HIGH	9.8 CRITICAL
CVE-2020-36177	wolfssl@4.4.0	Прямая	runtime	CycloneDX/cdxgen	10.0 HIGH	9.8 CRITICAL
CVE-2020-14343	pyyaml@5.3.1	Прямая	runtime	CycloneDX/cdxgen	10.0 HIGH	9.8 CRITICAL
CVE-2024-39705	nlk@3.8.1	Прямая	.freeze	CycloneDX/cdxgen	10.0 HIGH	9.8 CRITICAL
CVE-2022-45907	torch@1.11.0	Прямая	.freeze	CycloneDX/cdxgen	10.0 HIGH	9.8 CRITICAL
CVE-2023-45853	zlib@1.2.13	Прямая	runtime	aquasecurity/trivy	10.0 HIGH	9.8 CRITICAL
CVE-2022-37434	zlib@1.2.12	Транзитивная	runtime	aquasecurity/trivy	10.0 HIGH	9.8 CRITICAL



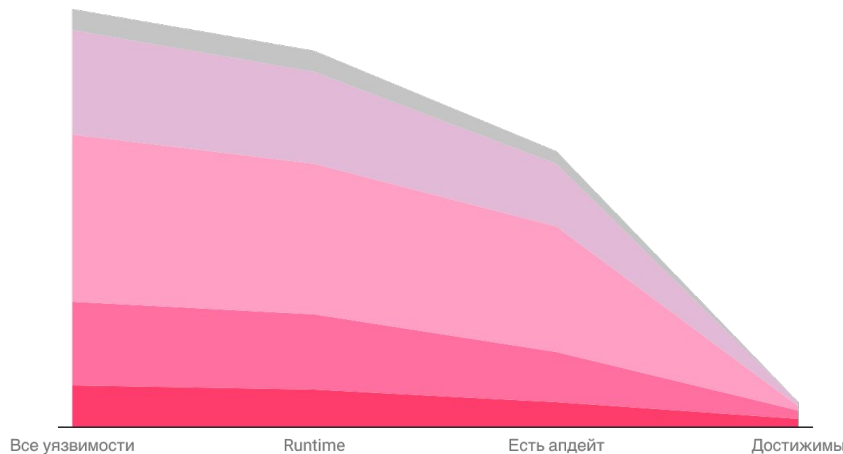
# Контекст угроз



- Анализ достижимости
- Определение этапа разработки
- Классификация: **CVSS 2 / CVSS 3**
- Информация об **эксплоитах** и патчах
- Рекомендации по **исправлению** уязвимостей

Доля уязвимостей на разных уровнях фильтрации

■ Critical ■ High ■ Medium ■ Low ■ None

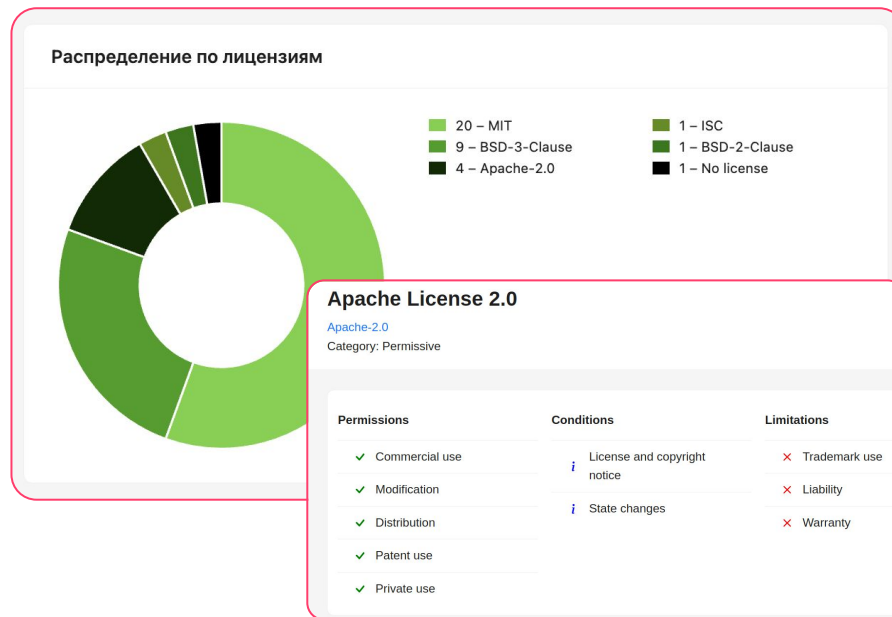




# Проверка лицензий



- **Крупнейшая** база лицензий в инструменте композиционного анализа ~**2500** шт.
- Проверка **лицензионной совместимости** компонентов
- Описание лицензий с перечнем ключевых условий
- **Отчеты** по лицензионному составу и нарушениям





# Политики безопасности

- Применение на разных этапах разработки и уровнях организации
- > **40** параметров с логическим объединением и вложенностью
- Возможность **блокировки** сборки или компонента
- **Уведомления:** почта, менеджер задач, ASPM, вебхуки
- Управление **игнорами**

The screenshot displays a configuration form for a security policy. The fields and their values are as follows:

- \* Название:** Критические уязвимости в директивных зависимостях
- Группы:** Группа 1
- Подразделения:** Отдел разработки веб-приложений
- Проекты:** matplotlib/matplotlib
- \* Этапы:** dev, prod
- Компоненты OSA:** (empty)
- Репозитории:** (empty)
- \* Уровень:** Критично
- Блокер:** ☒
- Активно:** ☒



# Гибкие сценарии политик

Найдена критическая уязвимость или пакет из стоп-листа

Заблокировать сборку, поставить задачу на третью линию

Выпущен пакет после конкретной даты (или слишком молодая версия)

Поместить в карантин и поставить задачу на SAST-ревью

Нарушена лицензионная чистота (найдена неподходящая лицензия)

Поставить задачу на замену библиотеки и уведомить юристов

Нарушено сочетание условий:  
critical in directive dependencies +  
has\_exploit + has\_fixed\_version

Поставить задачу напрямую на разработчиков, экономя время AppSec'ов.



# Поддерживаемые технологии

## Контейнерные образы



## Языки программирования



## Системные зависимости





# Интеграция в **жизненный цикл**

Менеджеры репозиториев



Платформы разработки



CI/CD конвейер



Таск-менеджеры



ASPM/ASOC-системы



Среды разработки



Инфраструктура

Почтовый сервер / LDAP / OIDC / API / вебхуки



The background is a solid pink color with a series of thin, dark pink diagonal lines that create a sense of motion or depth, radiating from the top left towards the bottom right.

**Демо**