CODE SCORING

Благими намерениями вымощена дорога в РБПО

бережливое конструирование политик безопасности разработки

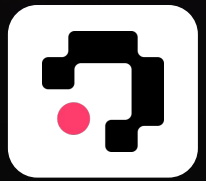
Алексей Смирнов
Основатель CodeScoring

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.TQI 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.Save* современное хранилище артефактов разработки с акцентом на безопасность и скорость

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

Построение ППК (SBOM) в целях сертификации

написали об этом

CodeScoring.SCA не только помогает бороться с техническим долгом и уязвимостями в рамках РБПО, но и поддерживает Требования ФСТЭК России к формату формирования перечня программных компонентов

Классические статистики

195,7 млн

версий пакетов всего

11,1 млн

пакетов всего

+28,0 млн

новых версий за 2025

+1,98 млн

новых пакетов за 2025

* Данные основаны на собственных результатах исследований

Пугающие статистики

+14 k

новых уязвимостей в 2025 году, включая React4Shell и возросшую эксплуатацию нулевого дня

+456 k

вредоносных версий пакетов зарегистрировал CodeScoring за 2025. Рост от 2024 года >1000%

* Данные основаны на собственных результатах исследований

Сколько сторонних уязвимостей в проектах?

Экосистема	Зависимостей	Директивных	Транзитивных	Уязвимостей, **когда нет Sec**	Уязвимостей, **когда Sec есть**
JavaScript	488	42	446	**50**	**10**
Java	90	54	36	**60**	**6**
C#	58	23	35	**4**	**1**
Python	39	16	23	**17**	**2**

* Данные основаны на собственных результатах исследований 2024 г.

«Благими намерениями вымощена дорога в ад» — крылатое выражение, означающее, что добрые помыслы, не подкрепленные делами или неправильно реализованные, приводят к пагубным результатам. Фраза подчеркивает, что из-за недальновидности, лени или неверных средств попытки сделать как лучше оборачиваются противоположным эффектом, часто являясь предостережением о вреде неосмысленных добрых порывов.

Примеры

основ таких намерений из жизни

Недальновидность и лень

1. «проверять тогда, когда попросит заказчик»
2. «в открытом коде 1000 глаз, они всё найдут»
3. «сторонний код слабо влияет на безопасность»
4. «где я столько людей найду?»
5. «опять они новые требования придумали»

приводят к анализу стороннего кода у себя,
только когда об этом написали в новостях
или не заплатили денег или ...

Неверные средства

1. «искать» в составе собранного продукта или на хостинге кода
2. клонировать кодовую базу и искать при помощи grep’а
3. верить, что всё можно найти руками, глазами и скриптом
4. запретить сторонние компоненты (всё в блок — пишите письма)
5. писать свой анализатор, потому что «а чего там такого?»

только подкрепляют желание страдать

Или обращаться к практикам

Композиционный анализ — не выдумка, а устоявшаяся практика автоматизации проверки сторонних компонентов на риски

~(c) ГОСТ Р 56939-2024

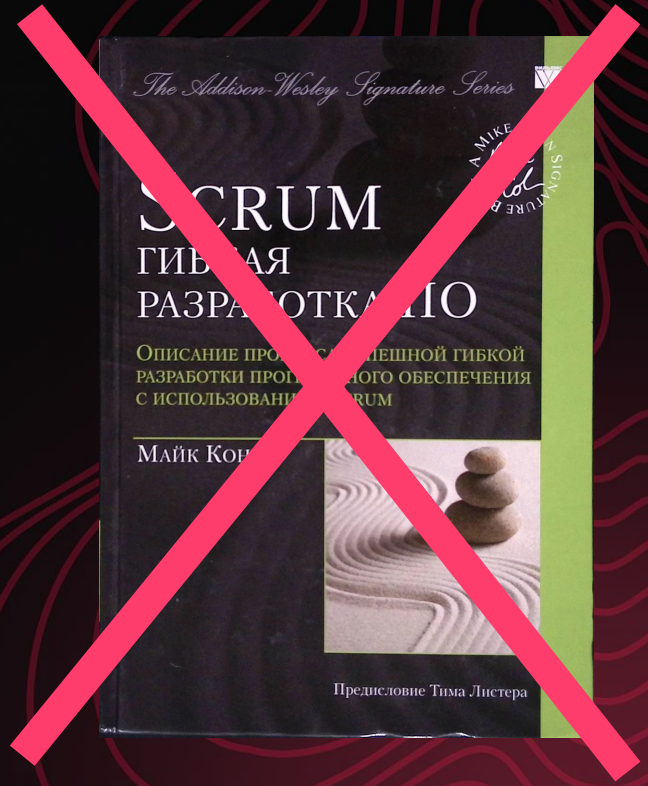
БРБПО

«новый» подход

Если у вас нет «чувства процесса» — придется много читать

Если у вас нет «чувства процесса» — придется много читать

иии, это не поможет



Чувства можно и нужно прокачивать практикой

Бережливое РБПО — когда все силы направлены на сокращение потерь в производстве, улучшение безопасности (качества) продукта и повышение эффективности процессов

Принципы бережливого производства

Материалы Kaiten

«Принцип производства: сами варим, сами мажем...» (с) Кирпичи

Клиент вашей РБПО — не Регулятор!

* просто напомним

Бережливые

политики безопасной разработки

Объем страданий определяет качество настройки политик безопасности

Критерии политик

CODE SCORING

Пакет

- PURL
- Название
- Версия
- Дата релиза (возраст)
- Автор(ы)
- Технология (язык)
- Тип пакета (сист. / разраб)
- Транзитивность Как нашли?
- Окружение сборки
- Признак вредоноса

Уязвимость

- Vuln ID
- CWE
- CVSS ⅔ Score
- CVSS ⅔ Severity
- Impact
- CVSS ⅔ части вектора
- Наличие эксплойта
- Наличие безоп. версии
- Наличие патча
- Дата публикац. / обновления

Лицензия

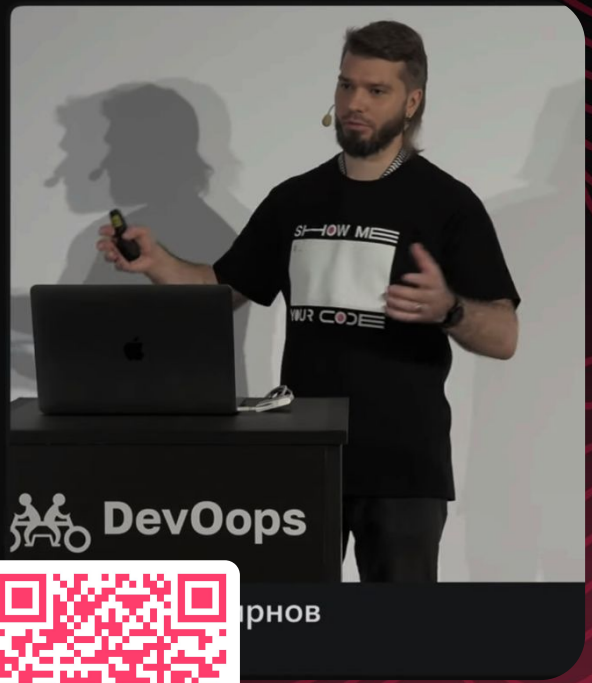
- Тип лицензии
- Категория лицензии
- Совместимость лицензии
- Экспортные ограничения
- Иные ограничения

Критериев политик, которые можно комбинировать, более 50 шт.

Примеры политик

Найдена критическая уязвимость или пакет из стоп-листа	Блокировать сборку, поставить задачу на третью линию, предложить без. версию
Выпущен пакет после конкретной даты (или слишком молодая версия)	Поместить в карантин и поставить задачу на SAST-ревью, предложить без. версию
Подтверждена достижимость уязвимости?	Предлагать митигацию разработчику: безопасная версия или путь фильтрации
Выполняется сочетание условий: critical in directive dependencies + has_exploit + has_fixed_version	Поставить задачу напрямую на разработчиков, экономя время AppSec'ов.

Прорастить или насадить? Практическое руководство по выращиванию политик безопасной разработки



vkvideo.ru/video-149053226_456239587

Одна из целей БРБПО — минимизация потерь

Потери: таксономия

1. Перепроизводство

Создание большего количества результатов анализа для триажа, чем нужно в данный момент, или до того, как появится реальная потребность.

Из-за этого РБПО-служба копит излишки, которым нужны дополнительное внимание: хранение и учет => тревога.

Пример: политика «чистой комнаты» — блокировка всех уязвимостей.

Потери: таксономия

1. Перепроизводство
2. Ожидания

Возникают, когда эксперты и инструменты ожидают следующего этапа производства.

Может происходить из-за простоев на разных этапах — например, ожидание, что разработчик сам разберет все уязвимости самостоятельно и без поддержки или что инструмент запустится по расписанию через сутки.

Пример: политика отправляющая результаты фильтрации на нерелевантных специалистов или лишний раз блокирующая сборку.

Потери: таксономия

1. Перепроизводство
2. Ожидания
3. Излишняя транспортировка

Возникает, когда задача на исправление уязвимости ожидает вливания в продуктивный код или пройдет тесты обратной совместимости или ждет других исправлений.

Требует дополнительных затрат времени, энергии и средств, но не всегда добавляет ценности. Чем больше этапов транспортировки, тем выше вероятность срыва сроков и потери качества.

Пример: слишком «широкая» политика, когда важное смешивается с неважным.

Потери: таксономия

1. Перепроизводство
2. Ожидания
3. Излишняя транспортировка
4. Излишние запасы

Ручной разбор всех-всех уязвимостей увеличенным штатом экспертов.

В итоге вложения сил (средств) съедают полезное внимание у других практик безопасной разработки.

Пример: отсутствие инструмента автоматизации и политик как механизма регуляции объема работ.

Потери: таксономия

1. Перепроизводство
2. Ожидания
3. Излишняя транспортировка
4. Излишние запасы
5. Лишние движения

Лишние движения происходят, когда эксперты делают дополнительные шаги, пересогласования, ревалидацию или другие действия, которые не помогают выполнить задачу.

Это может происходить из-за неудобной комбинации инструментов, а также из-за непродуманной организации процесса.

Пример: в политике не комбинируются критерии срабатывания, что приводит к лишним работам.

Потери: таксономия

1. Перепроизводство
2. Ожидания
3. Излишняя транспортировка
4. Излишние запасы
5. Лишние движения
6. Дефекты

Ошибочные результаты анализа. Из-за них приходится переделывать работу и сдвигать сроки.

Чтобы исправить недочеты, бизнесу нужно выделить еще больше времени и денег, что сказывается на качестве результата.

Пример: результаты политики не наводят на мысль, когда что-то не так с точностью анализа («всегда по нулям» или скан соседнего проекта).

Потери: таксономия

1. Перепроизводство
2. Ожидания
3. Излишняя транспортировка
4. Излишние запасы
5. Лишние движения
6. Дефекты
7. Недоиспользование потенциала

Связано с недостаточным обучением, отсутствием вовлеченности или неправильной организацией работы.

Сотрудники тратят время на задачи, которые они могли бы выполнить быстрее или эффективнее, если бы использовали более подходящие методы или средства.

Пример: политики не настраиваются в обучающем режиме («режим сужения ворот безопасности»).

Безопасность цепочки поставки как процесс

1 Разработчик

Pre-commit

- OSA
- SCA плагины для IDE

2 CI/CD конвейер

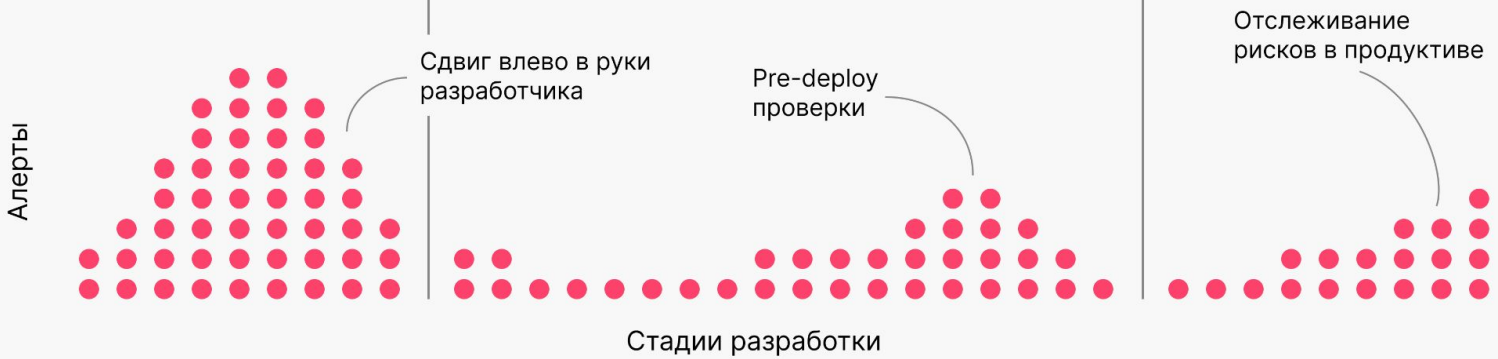
Pre-build | **Build** | **Post-build**

- OSA
- Source SCA
- SCA на сборке
- Binary SCA

3 Продуктив

Post-deploy

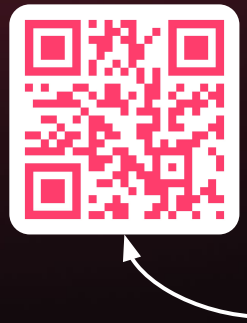
- SBOM SCA



отследить

Этот и другие материалы войдут в открытый курс «Безопасное использование Open Source»

Спасибо! Ваши вопросы?



Контакты:

hello@codescoring.ru

codescoring.ru — сайт

@codescoring — новости

Образование:

vkvideo.ru/@codescoring

youtube.com/@codescoring

История! Про спецсредство

Везем рекламный ролл-ап на конференцию:

- Консъерж: «Вы давно занимаетесь охотой?»
- Аэропорт. Рамка: «У вас спецсредство?»
- Аэропорт. Посадочная зона: «Молодой человек, что несете?»
- Аэропорт. КПП: «Ой, а давайте мы подробно изучим ваши вещи»