SBOM в SIEM для реагирования на инциденты и защиты облачной инфраструктуры

Алексей Вишняков

Кандидат физико-математических наук
Старший инженер DevSecOps, Yandex Cloud

@sweetvishnya

Whoami

Yandex Cloud

OFF ONE 2024

- Старший инженер DevSecOps, Yandex Cloud
- Внедряю процессы и инструменты безопасной разработки (SSDLC) в Yandex Cloud
- Закончил бакалавриат и магистратуру ВМК МГУ
- Кандидат физико-математических наук по методам автоматизированного поиска уязвимостей в программах
- Ранее 8 лет в ИСП РАН исследовал компьютерную безопасность, динамический анализ, символьное выполнение, фаззинг и др.
- Мейнтейнер поисковика гаджетов ROPgadget github.com/JonathanSalwan/ROPgadget и анализатора аварийных завершений после фаззинга CASR github.com/ispras/casr

Мотивация

- Yandex Cloud одновременно обслуживает множество сервисов, которые регулярно обновляются
- Важно контролировать состояние прода и понимать:

  Какие версии сервисов крутятся на проде?

  Где и кем были собраны пакеты / docker-образы?

  Из каких компонентов состоят? Подвержены ли они уязвимостям?

- Внесённая ошибка в общий код (например, в библиотеку логирования) может привести к нарушению безопасности целой группы сервисов — как понять, каким сервисам необходимо передеплоиться?
- Вышло очередное критичное RCE в библиотеке (например, Log4Shell) — как быстро найти на проде затронутые сервисы?

Цели

- Алерты в SIEM-системе

  Контроль целостности файлов (FIM) и docker-образов

  Файлы не должны попадать на прод в обход CI/CD-системы

- Политики деплоя

  Запрет на деплой из личных веток и отладочных пайплайнов

- Сканирование уязвимостей

  Регулярное сканирование зависимостей (SCA) сервисов, налитых в проде

- Автоматизированное расследование инцидентов ИБ

  Поиск сервисов на проде, собранных из уязвимого общего кода

  Обнаружение хостов, на которых запущены сервисы с уязвимой сторонней библиотекой

Work in Progress

- Обогащённые SBOM формируются для сервисов на Go и Java
- Интегрируем хранилище SBOM в SIEM-систему
- Текущие алерты позволяют убедиться, что сервис был собран в CI

  Однако поиск соответствующего SBOM осуществляется вручную

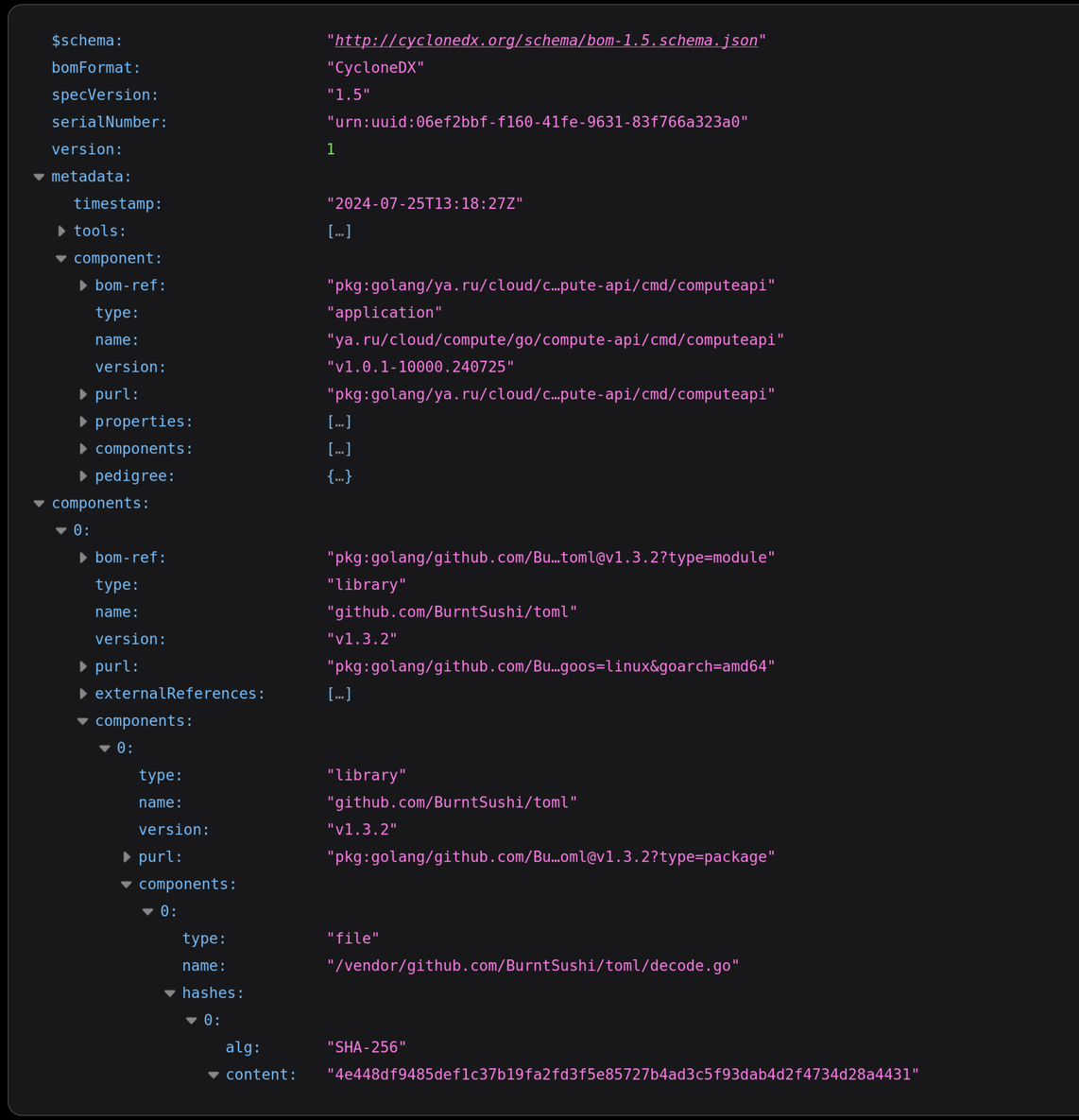
- Композиционный анализ (SCA) осуществляется во время сборки

  Хочется регулярно проверять зависимости сервисов, уже запущенных на проде, и порождать алерты

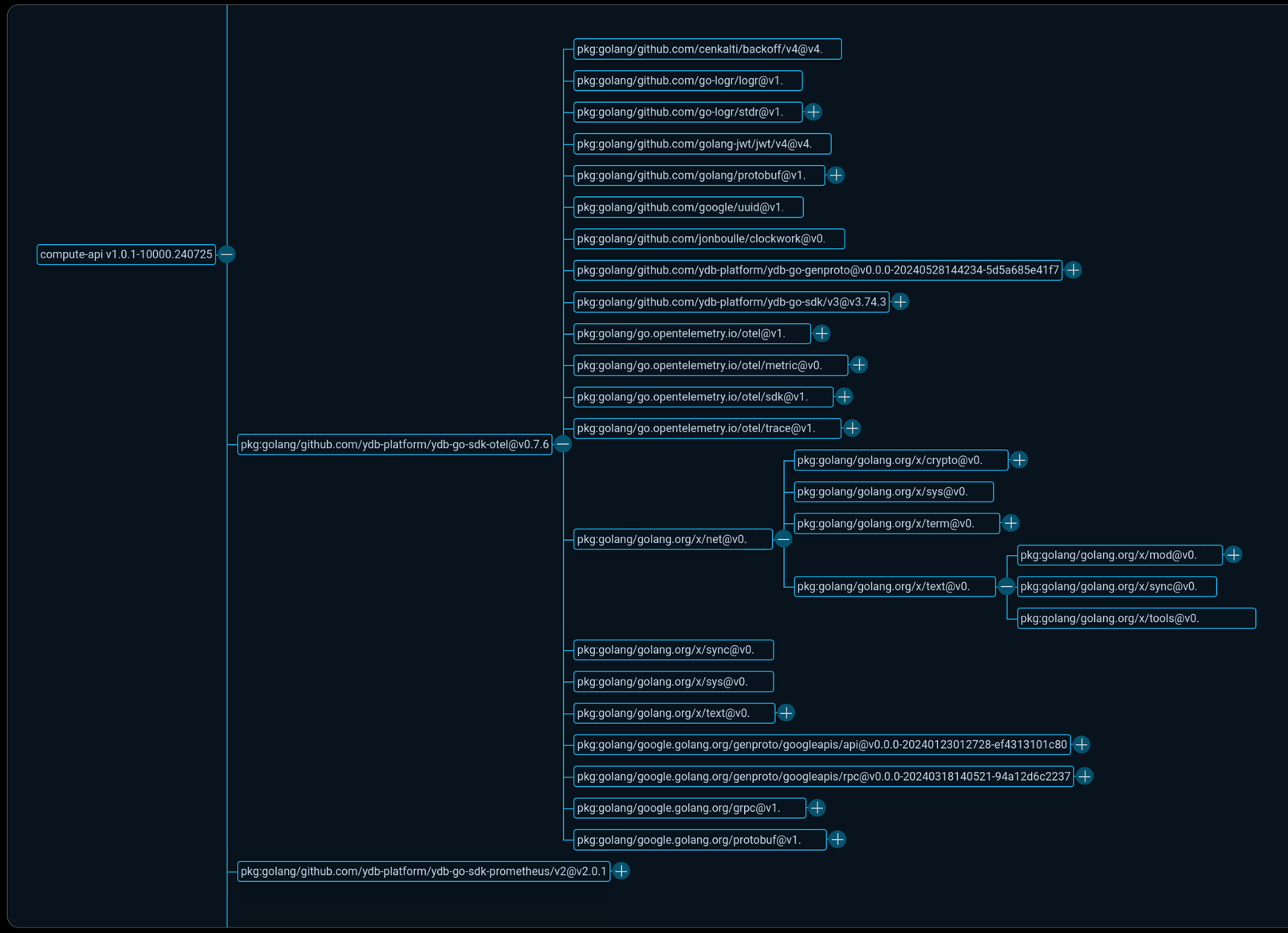
- Планируем для отправки событий с хешами файлов на хосте перейти с osquery (user-mode) на Tetragon (eBPF/LSM)

CycloneDX SBOM

- Software Bill of Materials (SBOM) содержит описание всех внутренних и сторонних зависимостей
- Базовая сущность в формате CycloneDX — компонент (библиотека, deb-пакет, docker-образ, файл и др.)
- `cyclonedx-gomod app -json -files -paths -licenses -packages -std -output bom.json -main ./compute-api/cmd/computeapi`
- Для Maven и Gradle есть плагины, которые генерируют SBOM во время mvn package и ./gradlew :service:app:cyclonedxBom



Dependency Track



Обогащение SBOM

Полученный SBOM обогащается недостающими метаданными о сборке и артефактах:

- Информация про VCS (хеш коммита, ссылка, автор, ветка, тег, сабмодули)
- Параметры сборки (ссылка, id, номер, имя, кем запущено, проект, сборочный агент, время)
- Deb-пакеты (имя, версия, описание, хеши файлов)
- Docker-контейнеры (имя, тег, digest, image id, время создания, хеши файлов)
- Хеши файлов с исходным кодом

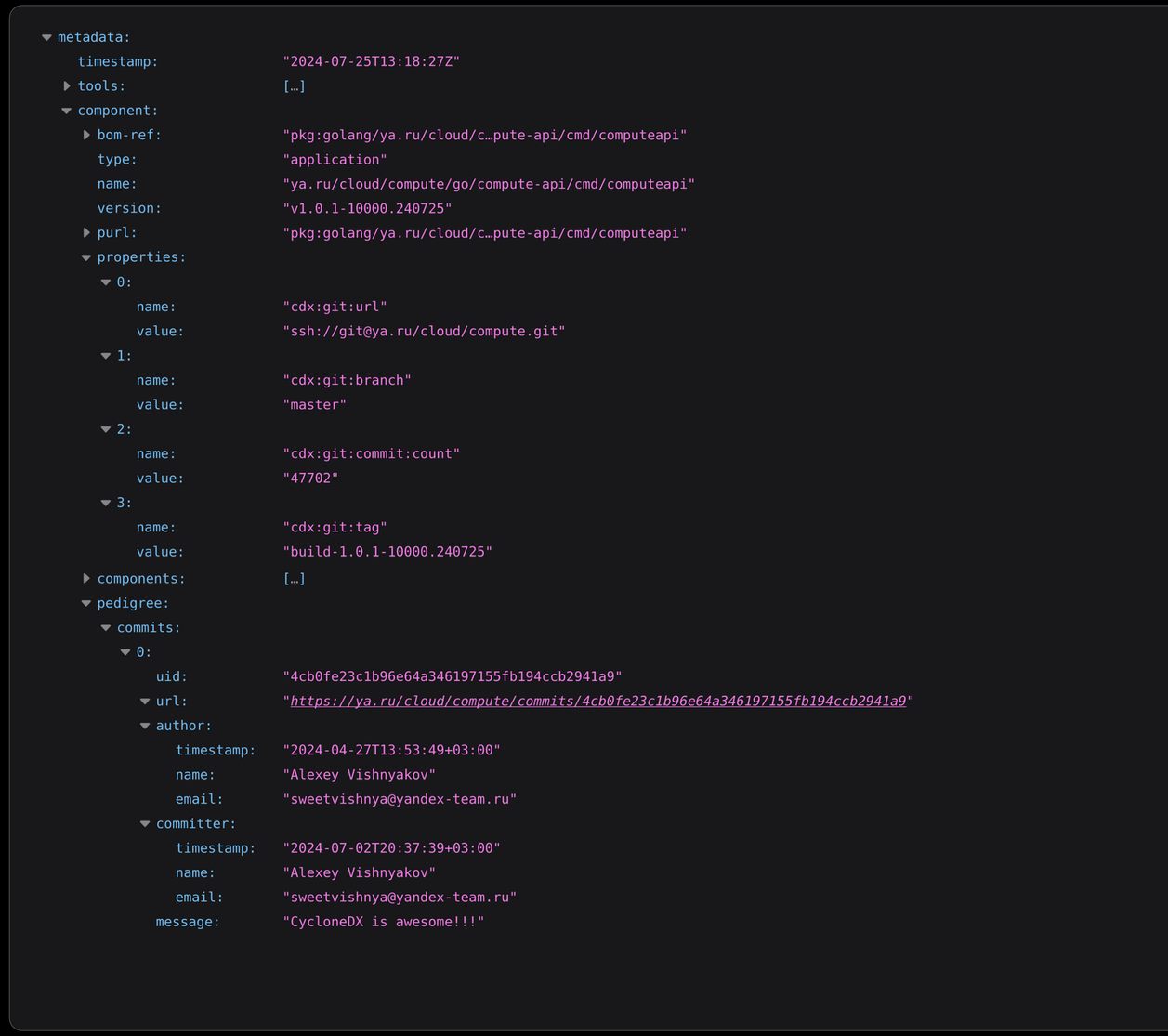
VCS Info

Информацию про коммит сохраняем
в pedigree, а остальное в properties:

```
git show --quiet --pretty=format:…
git config --get remote.origin.url
git branch --show-current
git rev-list --count HEAD
git describe --tags --exact-match
git submodule status --recursive
```

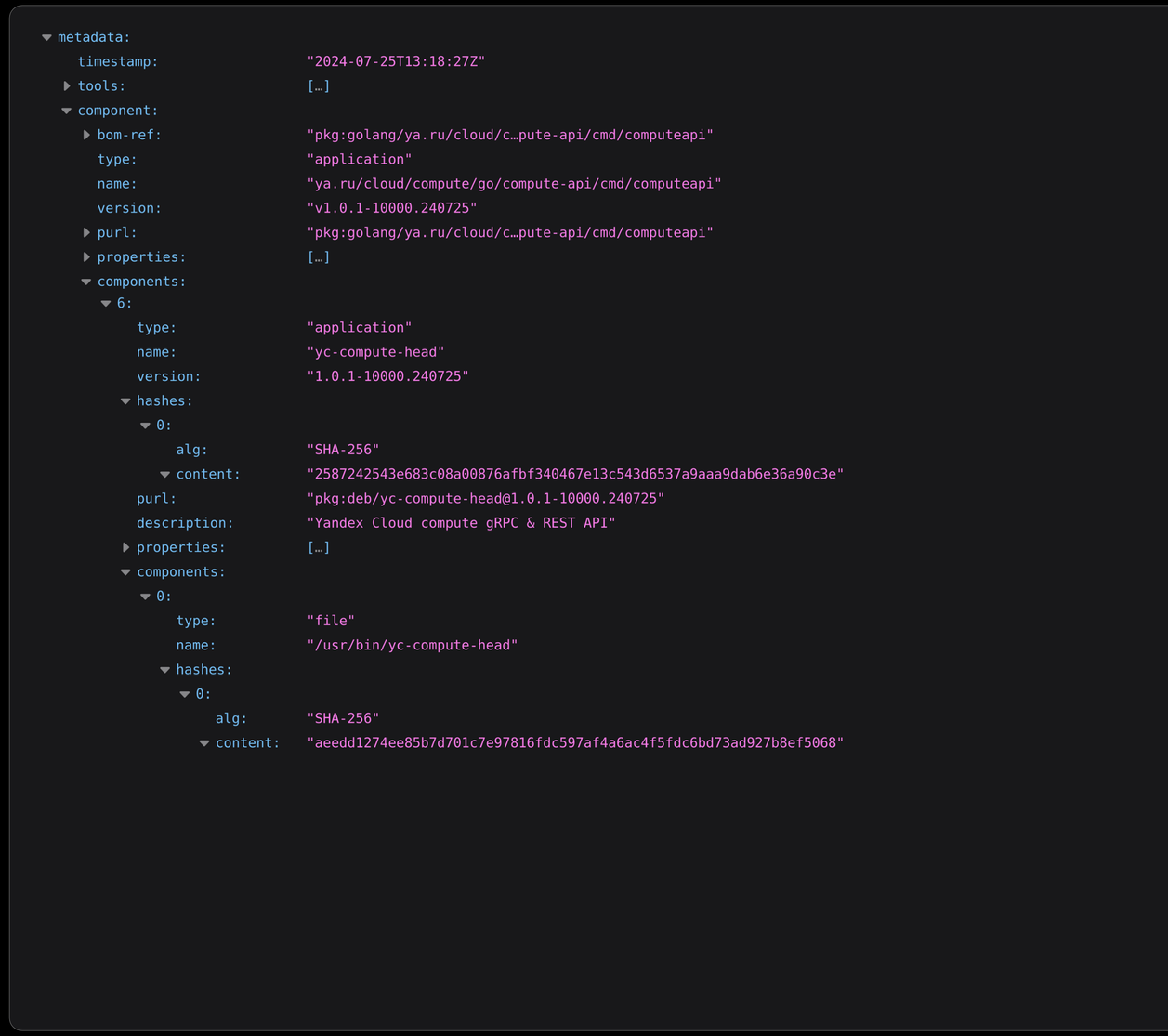
Ищем файлы bom.json, .deb и др.
только в изменённых файлах:

```
git status --ignored=matching -s
```



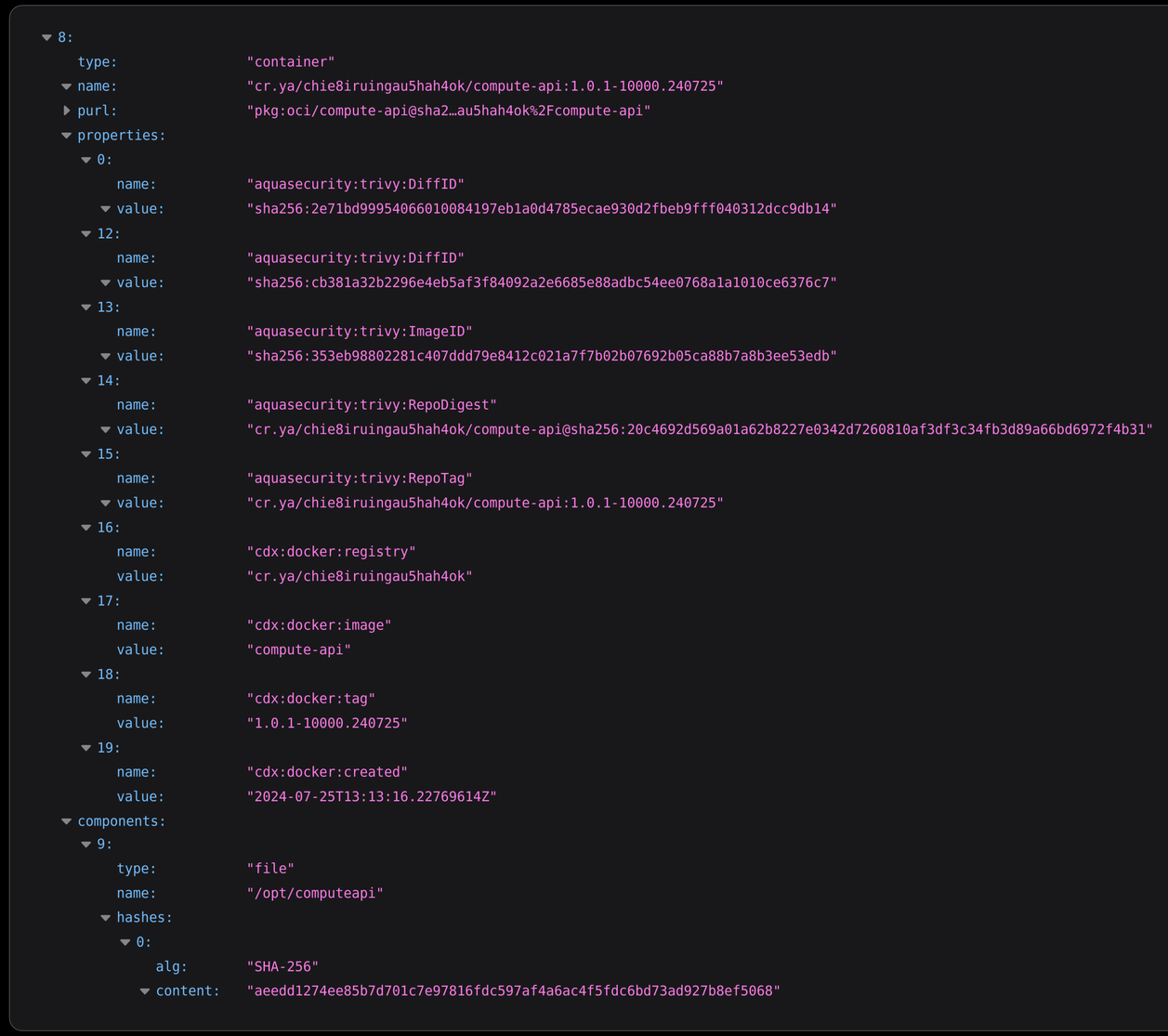
Deb-пакеты

- Получаем информацию про пакет: `dpkg –I`
- Считаем хеши для содержимого пакета: `dpkg –x`
- Помним, что в пакете могут быть вложенные deb-пакеты и архивы



Docker-образы

- Dive тяжеловесен, чтобы приносить его во все сборки
- Проанализируем докер своими руками
- Включим сохранение событий `docker events --format json`
- Запустим на каждом pushed-образе `docker inspect`
- Считаем хеши для файлов в слоях: `docker save`
- При этом пропускаем pulled layers



Хеши файлов с исходным кодом

- В cyclonedx-gomod пришлось поддержать сохранение полных путей до файлов с исходным кодом: github.com/CycloneDX/cyclonedx-gomod/pull/412
- Maven/Gradle плагины CycloneDX не считают хеши для исходного кода, поэтому:

  Смотрим содержимое jar-пакетов: `jar tf`

  Заменяем в пути .class на .java

  Считаем хеш для исходного файла с таким же суффиксом пути

Tetragon FIM

- Сейчас используется osquery, который считает хеши файлов в user-mode
- Tetragon выполняет код в ядре (eBPF)
- Подробнее про Tetragon: Андрей Федотов, расширение Tetragon для решения задач облачной безопасности, PHD 2, 2024
- Андрей поддержал LSM-события в Tetragon: github.com/cilium/tetragon/pull/2566
- Linux IMA позволяет эффективно считать хеши в ядре
- Таким образом, при различных операциях с файлами (LSM-событиях) мы можем эффективно считать их хеши в ядре (IMA) и отправлять в SIEM
- Объединив эти события с SBOM, можно написать FIM-алерт, который проверяет, что файлы сервиса собирались в CI

SCA для сервисов на проде

1

Получаем SBOM для запущенных на проде сервисов: хеши артефактов в SBOM должны совпадать с хешами из событий Tetragon

2

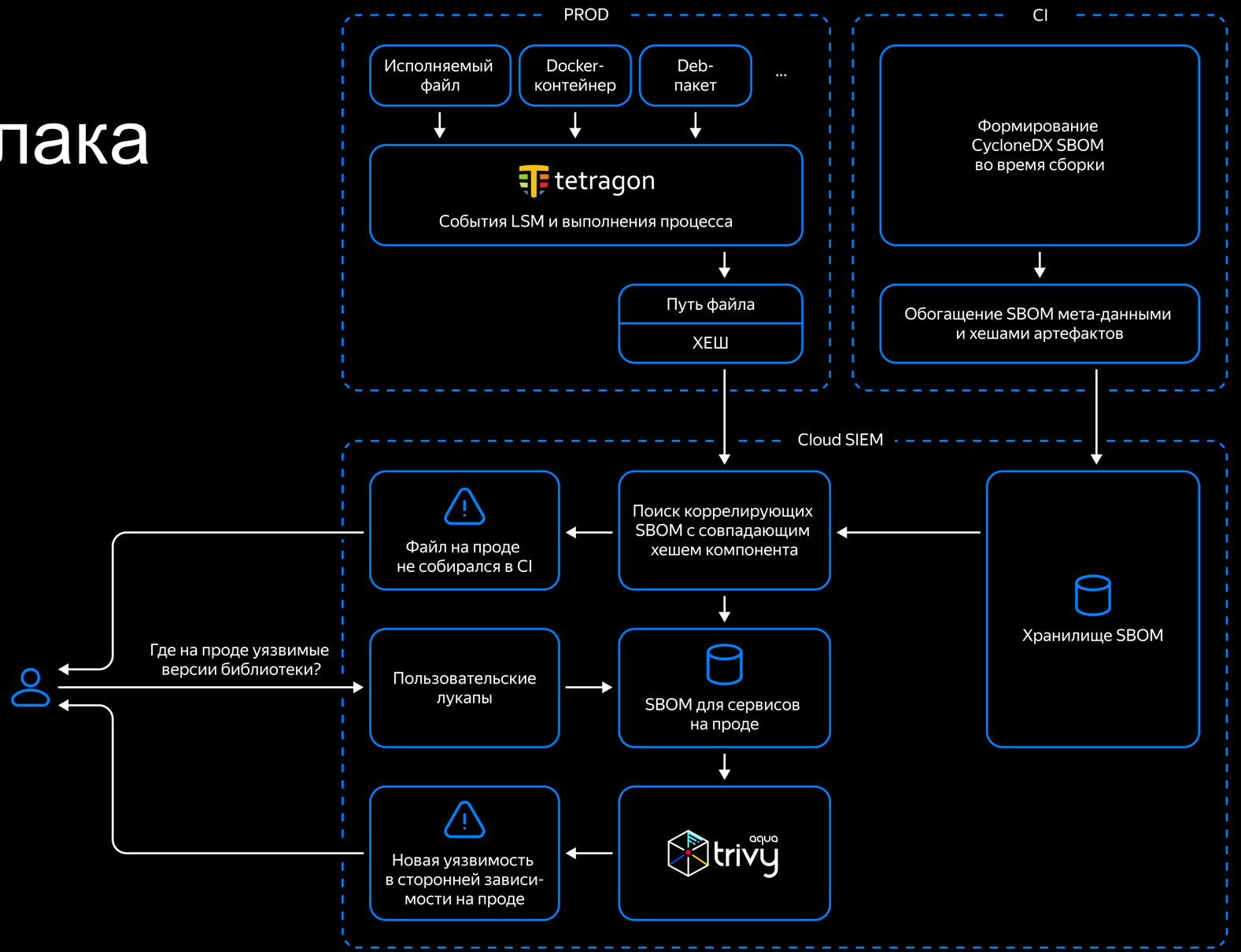
Регулярно запускаем Trivy на полученных SBOM

3

Порождаем алерты для новых обнаруженных уязвимостей в сторонних зависимостях

SBOM в SIEM для защиты облака

- Обогащённые SBOM формируются в сборках и отправляются в хранилище внутри SIEM
- Tetragon отправляет в SIEM события с хешами файлов на проде
- SIEM находит SBOM с совпадающими хешами артефактов или порождает алерт об их отсутствии
- Trivy регулярно проверяет SBOM сервисов на проде на наличие новых уязвимостей
- Пользовательские лукапы позволяют получить SBOM, удовлетворяющие заданным свойствам



Вызовы внедрения

- SIEM-система очень чувствительна к размеру событий
  - Необходимо удалить избыточную информацию из SBOM
  - Оставляем только SHA-256 хеш-суммы, удаляем версии для файлов и др.
  - Минимизируем JSON: `json.dump(bom, file, ensure_ascii=False, separators=(',', ':'))`
  - Размер даже уменьшенного обогащённого SBOM ~1МБ
  - Это слишком большой размер события
  - Будем складывать SBOM в отдельное хранилище с быстрым поиском по хешу компонента
- Большое разнообразие сборочных пайплайнов у сервисов
  - Пишем универсальные скрипты
  - Терпим и внедряем их во всевозможные виды сборок
- Не все сервисы универсально версионируются
  - Приводим версию к SemVer 2 перед формированием SBOM


NO FF ONE 2024
Yandex Cloud
Q&A
Алексей Вишняков
Кандидат физико-математических наук
Старший инженер DevSecOps, Yandex Cloud
@sweetvishnya