



Yandex @ Cloud

SBOM в SIEM для реагирования на инциденты и защиты облачной инфраструктуры

Алексей Вишняков

Кандидат физико-математических наук
Старший инженер DevSecOps, Yandex Cloud

Whoami

Yandex @ Cloud



- 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`

```
$schema: "http://cyclonedx.org/schema/bom-1.5.schema.json"
bomFormat: "CycloneDX"
specVersion: "1.5"
serialNumber: "urn:uuid:06ef2bbf-f160-41fe-9631-83f766a323a0"
version: 1
▼ metadata:
  timestamp: "2024-07-25T13:18:27Z"
  ► tools: [...]
  ▼ component:
    ► bom-ref: "pkg:golang/ya.ru/cloud/c...pute-api/cmd/computeapi"
    type: "application"
    name: "ya.ru/cloud/compute/go/compute-api/cmd/computeapi"
    version: "v1.0.1-10000.240725"
    ► purl: "pkg:golang/ya.ru/cloud/c...pute-api/cmd/computeapi"
    ► properties: [...]
    ► components: [...]
    ► pedigree: {}
  ▼ components:
    ▼ 0:
      ► bom-ref: "pkg:golang/github.com/Bu...toml@v1.3.2?type=module"
      type: "library"
      name: "github.com/BurntSushi/toml"
      version: "v1.3.2"
      ► purl: "pkg:golang/github.com/Bu...goos=linux&goarch=amd64"
      ► externalReferences: [...]
      ▼ components:
        ▼ 0:
          type: "library"
          name: "github.com/BurntSushi/toml"
          version: "v1.3.2"
          ► purl: "pkg:golang/github.com/Bu...oml@v1.3.2?type=package"
          ▼ components:
            ▼ 0:
              type: "file"
              name: "/vendor/github.com/BurntSushi/toml/decode.go"
              ▼ hashes:
                ▼ 0:
                  alg: "SHA-256"
                  ▼ content: "4e448df9485def1c37b19fa2fd3f5e85727b4ad3c5f93dab4d2f4734d28a4431"
```


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
```

```
▼ metadata:
  timestamp: "2024-07-25T13:18:27Z"
  ▶ tools: [...]
  ▼ component:
    ▶ bom-ref: "pkg:golang/ya.ru/cloud/c...pute-api/cmd/computeapi"
    type: "application"
    name: "ya.ru/cloud/compute/go/compute-api/cmd/computeapi"
    version: "v1.0.1-10000.240725"
    ▶ purl: "pkg:golang/ya.ru/cloud/c...pute-api/cmd/computeapi"
  ▼ properties:
    ▼ 0:
      name: "cdx:git:url"
      value: "ssh://git@ya.ru/cloud/compute.git"
    ▼ 1:
      name: "cdx:git:branch"
      value: "master"
    ▼ 2:
      name: "cdx:git:commit:count"
      value: "47702"
    ▼ 3:
      name: "cdx:git:tag"
      value: "build-1.0.1-10000.240725"
  ▶ components: [...]
  ▼ pedigree:
    ▼ commits:
      ▼ 0:
        uid: "4cb0fe23c1b96e64a346197155fb194ccb2941a9"
        ▼ url: "https://ya.ru/cloud/compute/commits/4cb0fe23c1b96e64a346197155fb194ccb2941a9"
        ▼ author:
          timestamp: "2024-04-27T13:53:49+03:00"
          name: "Alexey Vishnyakov"
          email: "sweetvishnya@yandex-team.ru"
        ▼ committer:
          timestamp: "2024-07-02T20:37:39+03:00"
          name: "Alexey Vishnyakov"
          email: "sweetvishnya@yandex-team.ru"
          message: "CycloneDX is awesome!!!"
```

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

```
▼ metadata:
  timestamp: "2024-07-25T13:18:27Z"
  ▶ tools: [...]
  ▼ component:
    ▶ bom-ref: "pkg:golang/ya.ru/cloud/c...pute-api/cmd/computeapi"
    type: "application"
    name: "ya.ru/cloud/compute/go/compute-api/cmd/computeapi"
    version: "v1.0.1-10000.240725"
    ▶ purl: "pkg:golang/ya.ru/cloud/c...pute-api/cmd/computeapi"
    ▶ properties: [...]
  ▼ components:
    ▼ 6:
      type: "application"
      name: "yc-compute-head"
      version: "1.0.1-10000.240725"
      ▼ hashes:
        ▼ 0:
          alg: "SHA-256"
          ▼ content: "2587242543e683c08a00876afbf340467e13c543d6537a9aaa9dab6e36a90c3e"
          purl: "pkg:deb/yc-compute-head@1.0.1-10000.240725"
          description: "Yandex Cloud compute gRPC & REST API"
          ▶ properties: [...]
          ▼ components:
            ▼ 0:
              type: "file"
              name: "/usr/bin/yc-compute-head"
              ▼ hashes:
                ▼ 0:
                  alg: "SHA-256"
                  ▼ content: "aeedd1274ee85b7d701c7e97816fdc597af4a6ac4f5fdc6bd73ad927b8ef5068"
```

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`

```
▼ 8:
  type: "container"
  ▼ name: "cr.ya/chie8iruingau5hah4ok/compute-api:1.0.1-10000.240725"
  ▶ purl: "pkg:oci/compute-api@sha2...au5hah4ok%2Fcompute-api"
  ▼ properties:
    ▼ 0:
      name: "aquasecurity:trivy:DiffID"
      ▼ value: "sha256:2e71bd99954066010084197eb1a0d4785ecae930d2fbeb9fff040312dcc9db14"
    ▼ 12:
      name: "aquasecurity:trivy:DiffID"
      ▼ value: "sha256:cb381a32b2296e4eb5af3f84092a2e6685e88adb54ee0768a1a1010ce6376c7"
    ▼ 13:
      name: "aquasecurity:trivy:ImageID"
      ▼ value: "sha256:353eb98802281c407ddd79e8412c021a7f7b02b07692b05ca88b7a8b3ee53edb"
    ▼ 14:
      name: "aquasecurity:trivy:RepoDigest"
      ▼ value: "cr.ya/chie8iruingau5hah4ok/compute-api@sha256:20c4692d569a01a62b8227e0342d7260810af3df3c34fb3d89a66bd6972f4b31"
    ▼ 15:
      name: "aquasecurity:trivy:RepoTag"
      ▼ value: "cr.ya/chie8iruingau5hah4ok/compute-api:1.0.1-10000.240725"
    ▼ 16:
      name: "cdx:docker:registry"
      value: "cr.ya/chie8iruingau5hah4ok"
    ▼ 17:
      name: "cdx:docker:image"
      value: "compute-api"
    ▼ 18:
      name: "cdx:docker:tag"
      value: "1.0.1-10000.240725"
    ▼ 19:
      name: "cdx:docker:created"
      value: "2024-07-25T13:13:16.22769614Z"
  ▼ components:
    ▼ 9:
      type: "file"
      name: "/opt/computeapi"
      ▼ hashes:
        ▼ 0:
          alg: "SHA-256"
          ▼ content: "aeedd1274ee85b7d701c7e97816fdc597af4a6ac4f5fdc6bd73ad927b8ef5068"
```


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

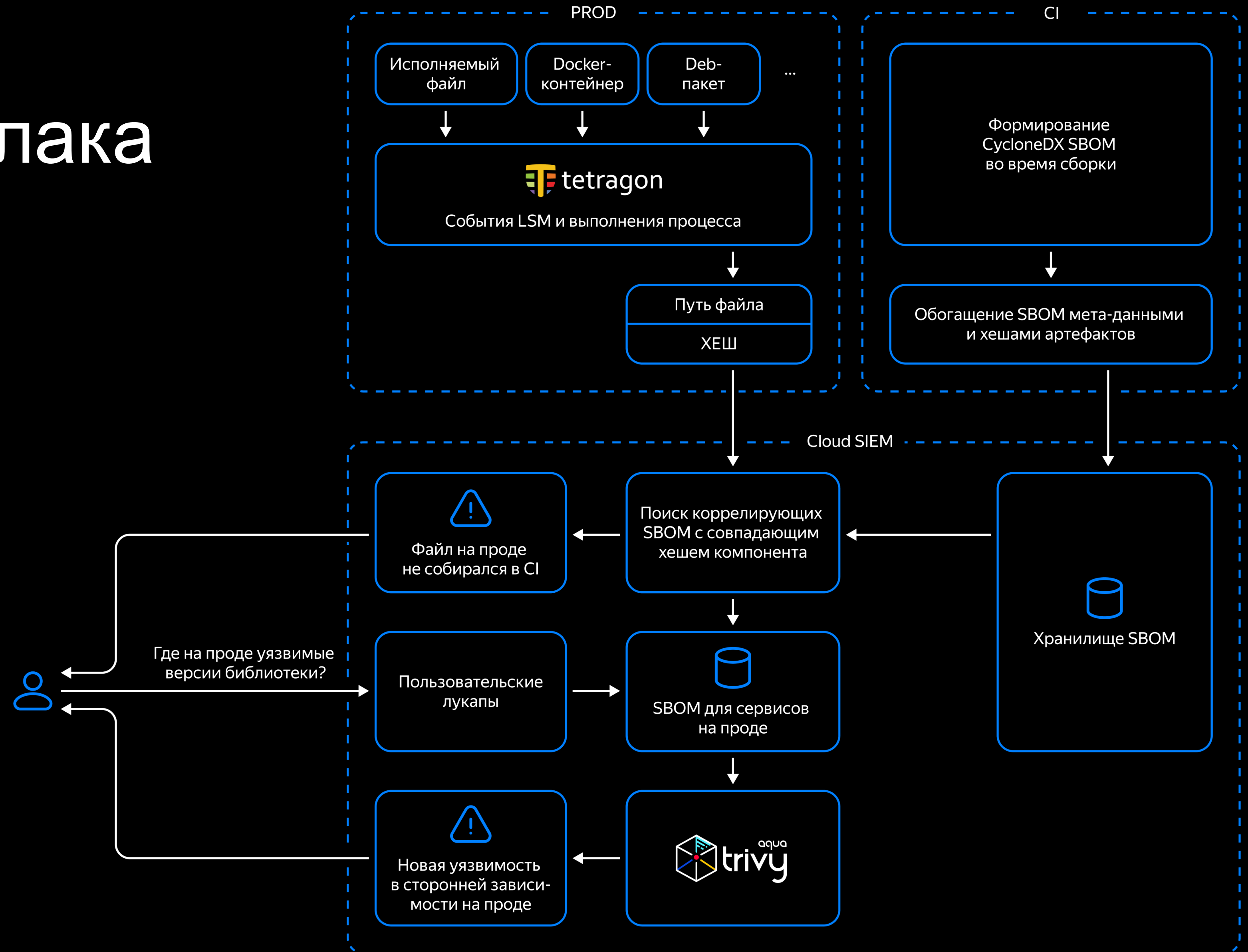
Регулярно запускаем Trivy на полученных SBOM

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 версионизируются

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