Security best practice airflow

Секреты

- Нельзя пушить секреты в открытом виде внутри кода. Ни в каком виде.
- Для работы с секретами нужно использовать коннекшены airflow.
- Не использовать .eval() для чтения секретов из коннекшена, для этого использовать библиотеку json
- Следить за тем, чтобы секреты не логировались ни в каком виде
- Чтобы разобраться, что считать секретом + раскрыть тему работы с ними, прочитай документацию: Хранение секретов
- Если ваш код не использует airflow и его коннекшены вам недоступны, то инструкция выше - вам поможет

Внешние хранилища данных(docs.google.com)

- Не использовать гугл доки ни в каком виде, даже для временной передачи данных
- Для функционала электронных таблиц использовать nextcloud.sbmt.io
- Доступ к гугл докам скоро будет ПОЛНОСТЬЮ запрещен (на уровне сетевого соединения)
- Если у вас остались гугл документы - переносить их на nextcloud.sbmt.io
- Ваш даг не должен запрашивать данные извне, даже библиотеки

Небезопасные функции/зависимости

- Использование **eval()** - только в крайних случаях и только после согласования с Appsec BP Ташлыков Данил Вениаминович
- Нельзя использовать ***pip install*** *(и остальные менеджеры зависимостей)* внутри кода, все зависимости нужные для работы должны быть указаны в репозитории и скачаны с nexsus
- Нельзя нарушать флоу сборки дага, вся логика зависимостей сборки и деплоя должна быть четко прописана в ci репозитория
- Так же избегайте **прямого исполнения команд в ваших дагах внутри пода** (os.process(echo 'kek')), это является уязвимостью и должно быть исправлено
- Если вам нужно использовать **yaml.load()** - вместо этого используйте **yaml.safe_load()** - это безопасная реализация того же метода
- При распаковке **xml** файла, используйте **defusedxml** библиотеку вместо **xml**.
  НАПРИМЕР:
  - вместо *xml.minidom.parseString()* нужно использовать *defusedxml.minidom.parseString()*
  - вместо *xml.etree.ElementTree.fromString()* нужно использовать *defusedxml.ElementTree.fromstring()*

Процесс детекта и исправления уязвимостей

- При первой сборке проекта в сайд ветке, обратить внимание на пайплайн с названием security:base
  В этом пайплайне собраны все проблемы безопасности у вашего кода
  Если пайп желтый или красный - значит есть проблемы
- Внутри пайплайна вас интересует 4 джобы с названиями :results в конце, а именно:
  - sast:results - Это уязвимости в коде дага, которые нужно исправить или добавить в исключение
  - secrets:results - Здесь находится список кредов/секретов, которые нужно перенести в коннекшены
  - dependencies:results - Здесь находится список зависимостей, которые нужно обновить
  - image_scan:results - Здесь список пакетов, которые нужно обновить в image проекта
- Заходите в эти джобы - смотрите проблемы - исправляете их или добавляете в исключения
- Как работать с исключениями: Руководство по работе с исключениями для уязвимостей, найденных в пайплайне
- Если не знаете как исправлять уязвимость - пишите вашему Appsec BP Ташлыков Данил Вениаминович
- Если у вас есть уязвимость в проде - есть sla на исправление уязвимости исходя из ее критичности, подробнее об этом можно почитать в документе: Политика управления уязвимостями
- Так же у нас есть политика Security Quality Gate, которая нас всех ждет, описание вот тут: Стандарт проведения тестирования разрабатываемых приложений на соответствие требованиям кибербезопасности при использовании конвейера разработки
- Отдельно скажу, что если вам интересна безопасность и вы хотите стать секьюрити чемпионом в своей команде, то напишите своему Appsec BP Ташлыков Данил Вениаминович
  Практика чемпионов в аналитику не внедрена, но обучение пройти можно 🙂

Code style

- Отличную заметку написал Неизвестный пользователь (mukhamedzhanov.dyu) вот тут: Code-style Airflow и что-то еще
  Настоятельно советую ознакомиться