Содержимое проекта:

- dump

- backup.sql (резервная копия хранилища данных, используемого в проекте)

- nginx

- nginx.conf (главный конфигурационный файл nginx)

- templates

- index.html (файл вёрстки веб-сайта)

- docker-compose.yml (файл конфигурации сети контейнеров)

- Dockerfile (файл с инструкциями, необходимыми для создания образа контейнера)

- main.py (файл веб-приложения на Flask)

- requirements.txt (файл с необходимым набором модулей и пакетов)

Используемые контейнеры:

- PostgreSQL (postgres\_1lab\_docker)

- Flask (python\_1lab\_docker)

- Nginx (nginx\_1lab\_docker)

Потом мы запускаем проект:

1. Установить Docker Desktop

2. Открыть командную строку из директории скачанного проекта

3. Выполнить команду

docker-compose up

4. В pgAdmin создать новый сервер с настройками

hostname: localhost

port: 5432

database: db\_person

username: postgres

password: 1234

5. Создать новую таблицу "Person" c колонками (или восстановить бэкап backup.sql из папки dump):

id (int)

full\_name (varchar(100))

group\_number (varchar(100))

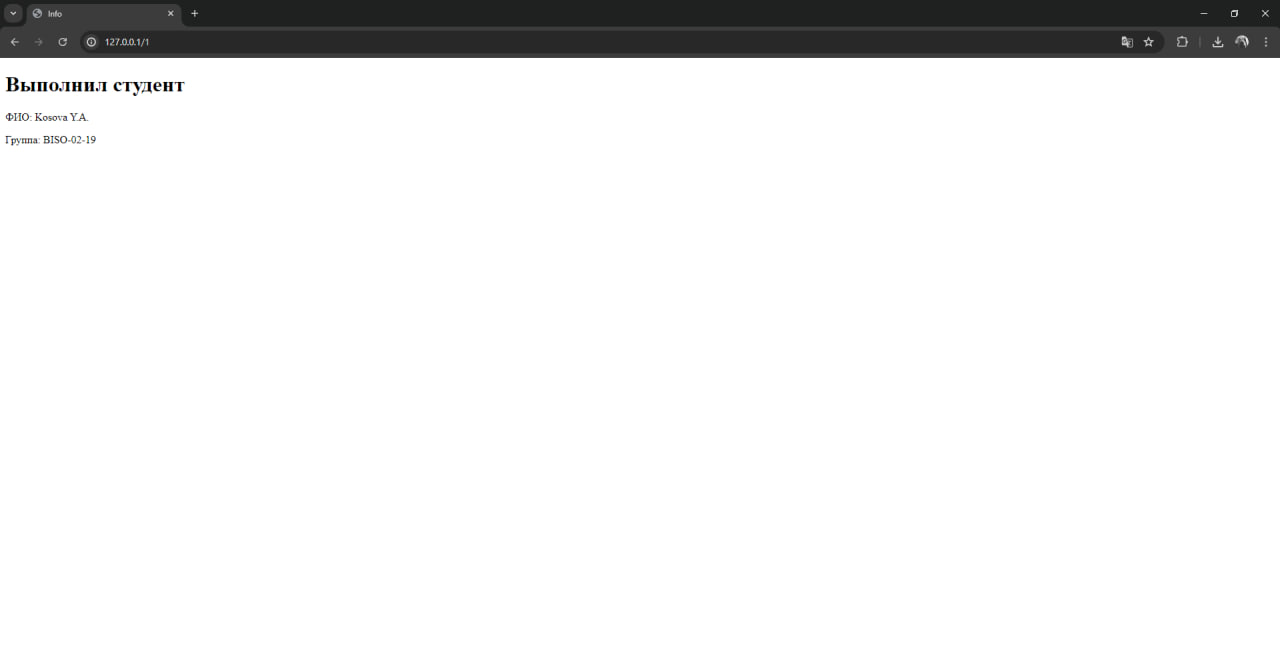
6. Заполнить таблицу "Person" информацией

7. Сделать запрос в браузере по адресу:

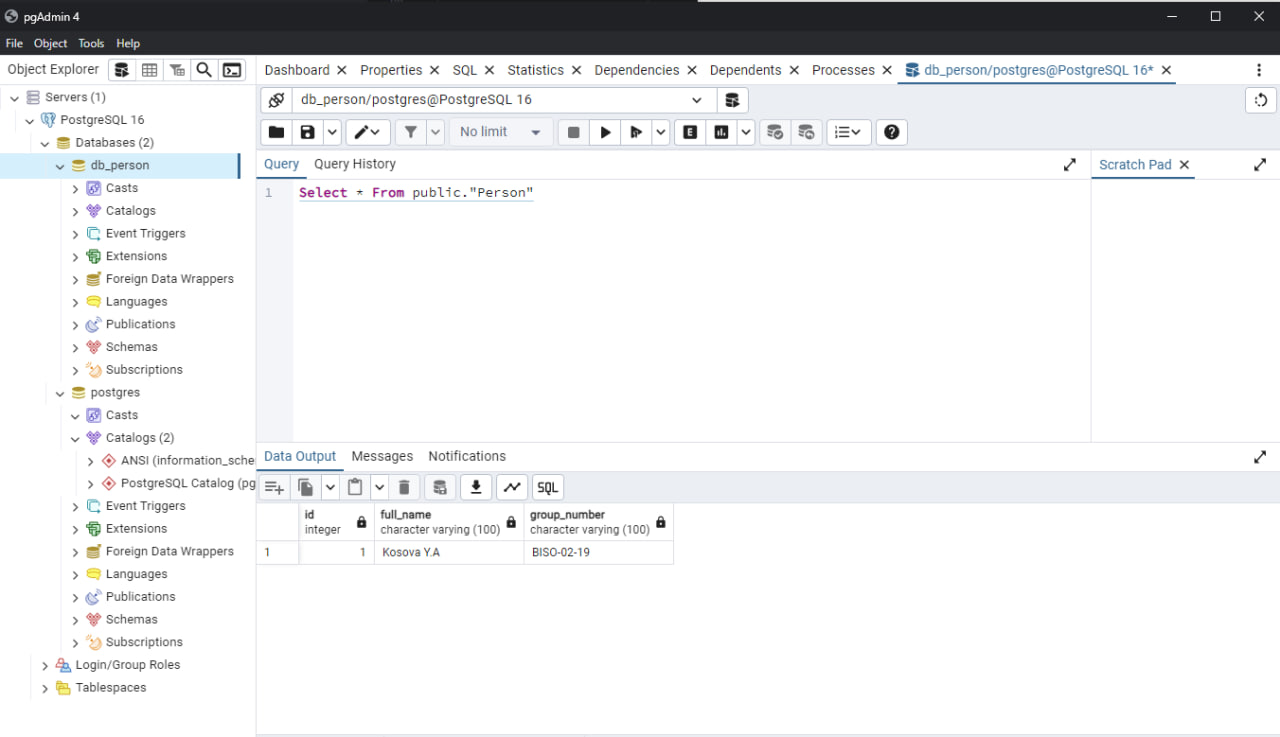
http://127.0.0.1/1

## Результаты выполнения проекта

- Скриншот работоспособности веб-сайта



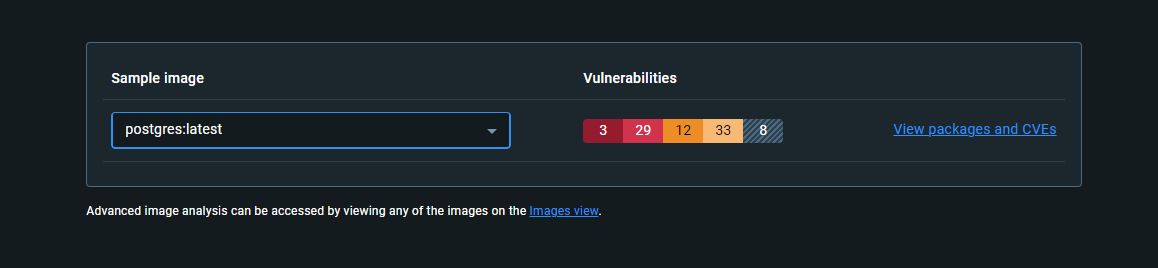
- Скриншот хранилища данных



## Обнаруженные уязвимости (сканирование Docker Scout)

1. В контейнере PostgreSQL

- Общий результат



- Наиболее значимые уязвимости по оценкам сканера

- Уязвимость 1

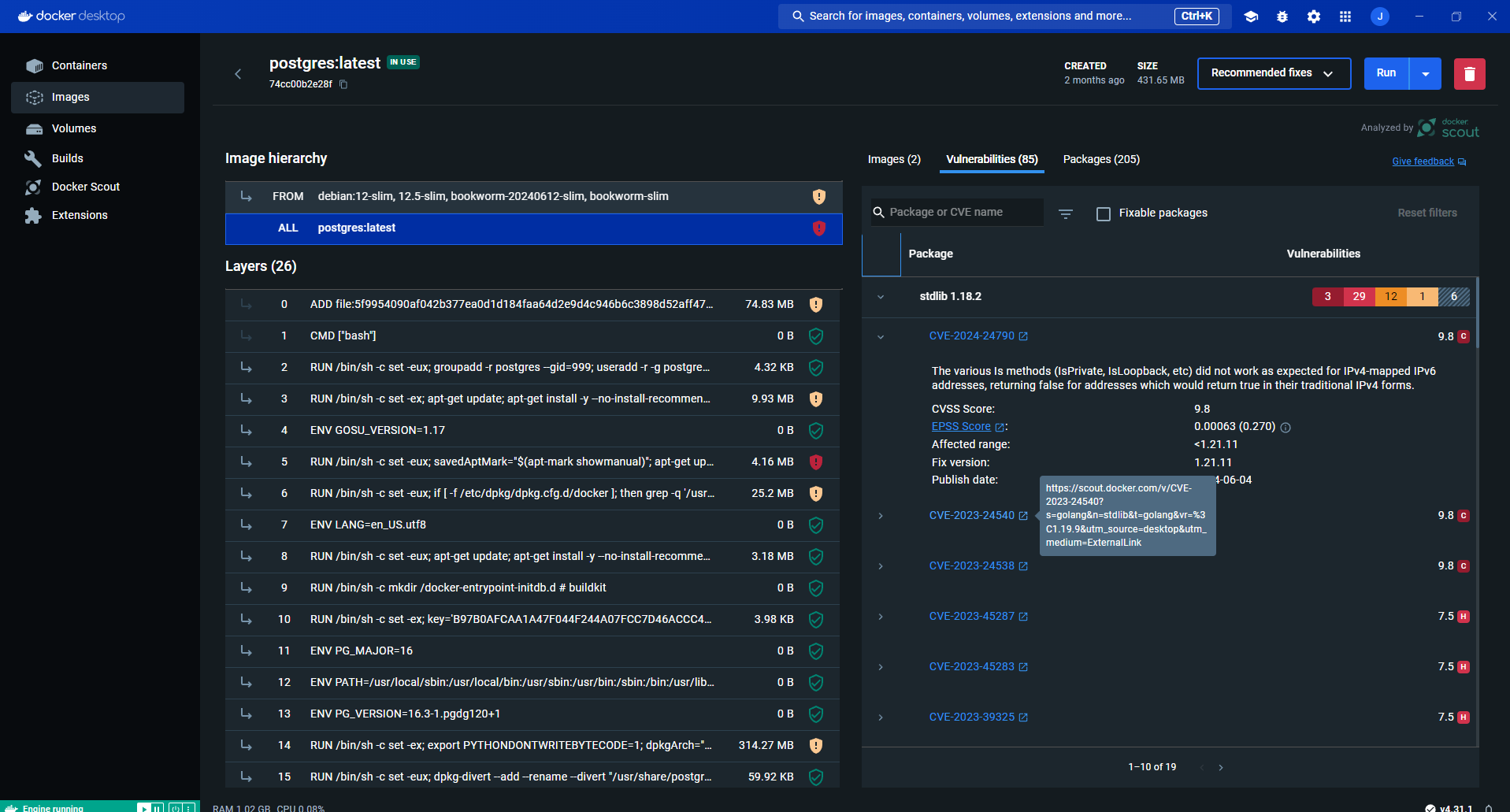
- Код уязвимости:

CVE-2023-24790

- Описание уязвимости:

Различные методы Is (IsPrivate, IsLoopback и т. д.) не работали должным образом для адресов IPv6, сопоставленных с IPv4, возвращая false для адресов, которые возвращали бы true в своих традиционных формах IPv4.

- Устранение уязвимости:

Запретить использование в ОС пакета golang или Установить обновление для пакета(ов) golang  
  
Скриншот отображения уязвимости:  
 

- Уязвимость 2

- Код уязвимости:

CVE-2023-24540

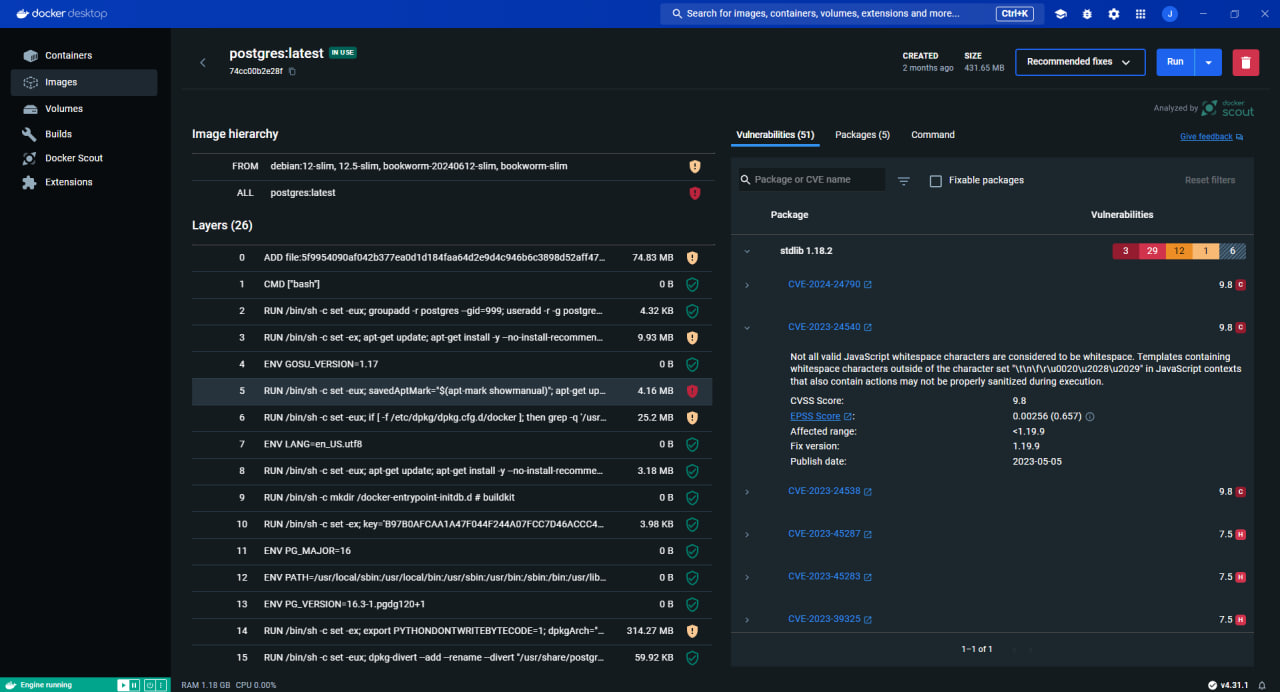
- Описание уязвимости:

Не все допустимые пробельные символы JavaScript считаются пробельными. Шаблоны, содержащие пробельные символы за пределами набора символов "\t\n\f\r \u0020\u2028\u2029" в контекстах JavaScript, которые также содержат действия, могут не подвергаться надлежащей очистке во время выполнения.

Устранение уязвимости:

Избегание использования пробельных символов, которые не входят в набор "\t\n\f\r \u0020\u2028\u2029", в особенности в шаблонах, где они могут быть введены пользователем или внешними источниками

Скриншот отображения уязвимости



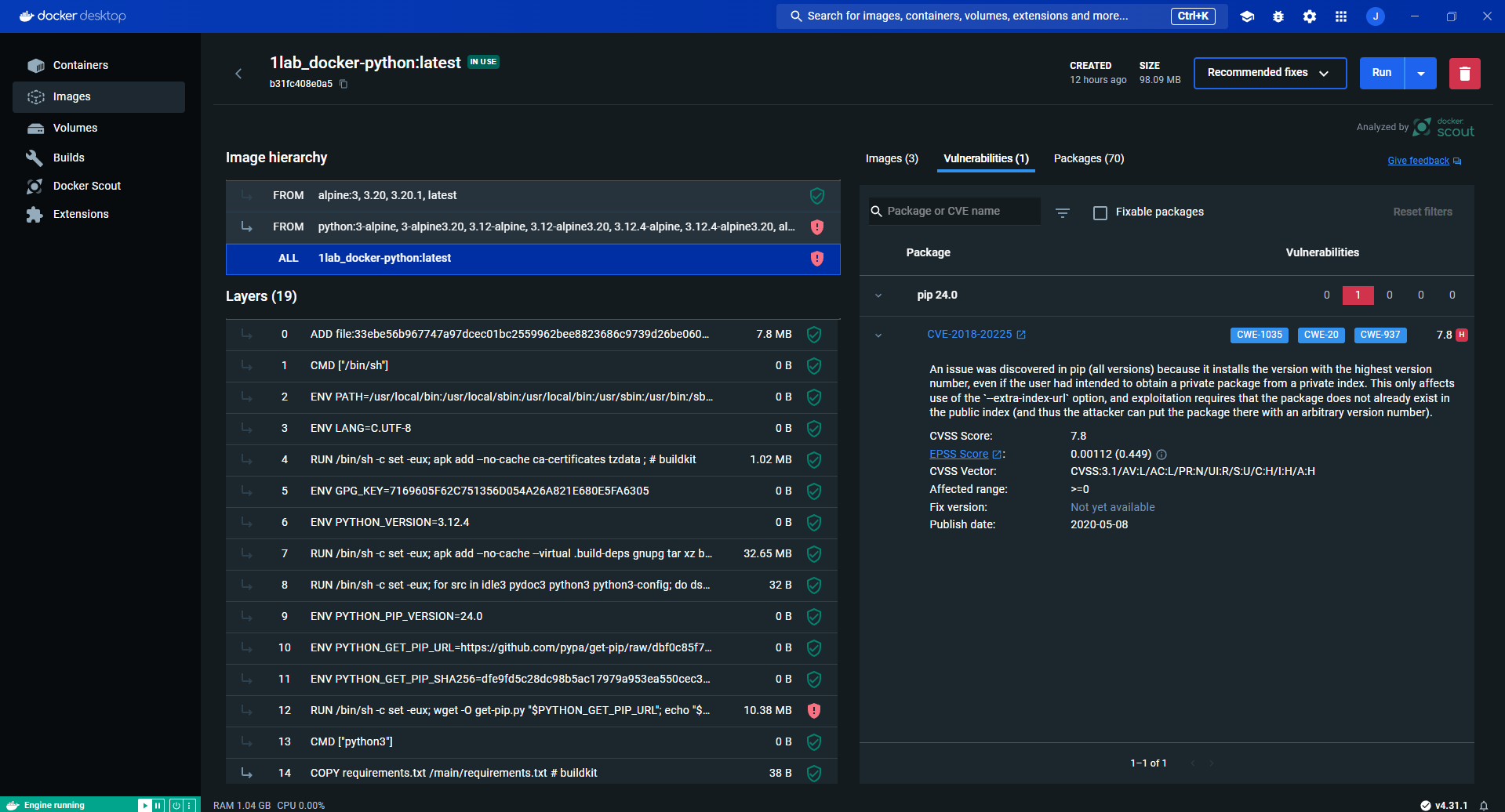
-Уязвимость 3   
- Код уязвимости:   
 CVE-2018-24538   
 - Описание уязвимости:

Шаблоны неправильно рассматривают обратные кавычки (`) как разделители строк Javascript и не экранируют их должным образом. Обратные кавычки используются, начиная с ES6, для литералов шаблонов JS. Если шаблон содержит действие шаблона Go внутри литерала шаблона Javascript, содержимое действия можно использовать для завершения литерала, внедряя произвольный код Javascript в шаблон Go. Поскольку литералы шаблона ES6 довольно сложны и сами могут выполнять интерполяцию строк, было принято решение просто запретить использование действий шаблона Go внутри них (например, «var a = {{.}}»), поскольку очевидно безопасный способ разрешить такое поведение. Здесь используется тот же подход, что и на github.com/google/safehtml. После исправления Template.Parse возвращает ошибку при обнаружении подобных шаблонов со значением ErrorCode 12. Этот ErrorCode в настоящее время не экспортируется, но будет экспортирован в версии Go 1.21. Пользователи, которые полагаются на предыдущее поведение, могут снова включить его, используя флаг GODEBUG jstmpllitinterp=1, с оговоркой, что обратные кавычки теперь будут экранированы. Это следует использовать с осторожностью.

- Устранение уязвимости: Данная уязвимость устраняется официальным патчем вендора. В связи со сложившейся обстановкой и введенными санкциями против Российской Федерации рекомендуем устанавливать обновления программного обеспечения только после оценки всех сопутствующих рисков.

2. В контейнере Flask

- Общий результат (включая скриншот Уязвимости 1)



- Уязвимость 1

- Код уязвимости:

CVE-2018-20225⁠

- Описание уязвимости:

В pip (все версии) обнаружена проблема, поскольку он устанавливает версию с наибольшим номером версии, даже если пользователь намеревался получить закрытый пакет из частного индекса. Это влияет только на использование параметра --extra-index-url, и для его использования требуется, чтобы пакет еще не существовал в общедоступном индексе (и, таким образом, злоумышленник может поместить туда пакет с произвольным номером версии.

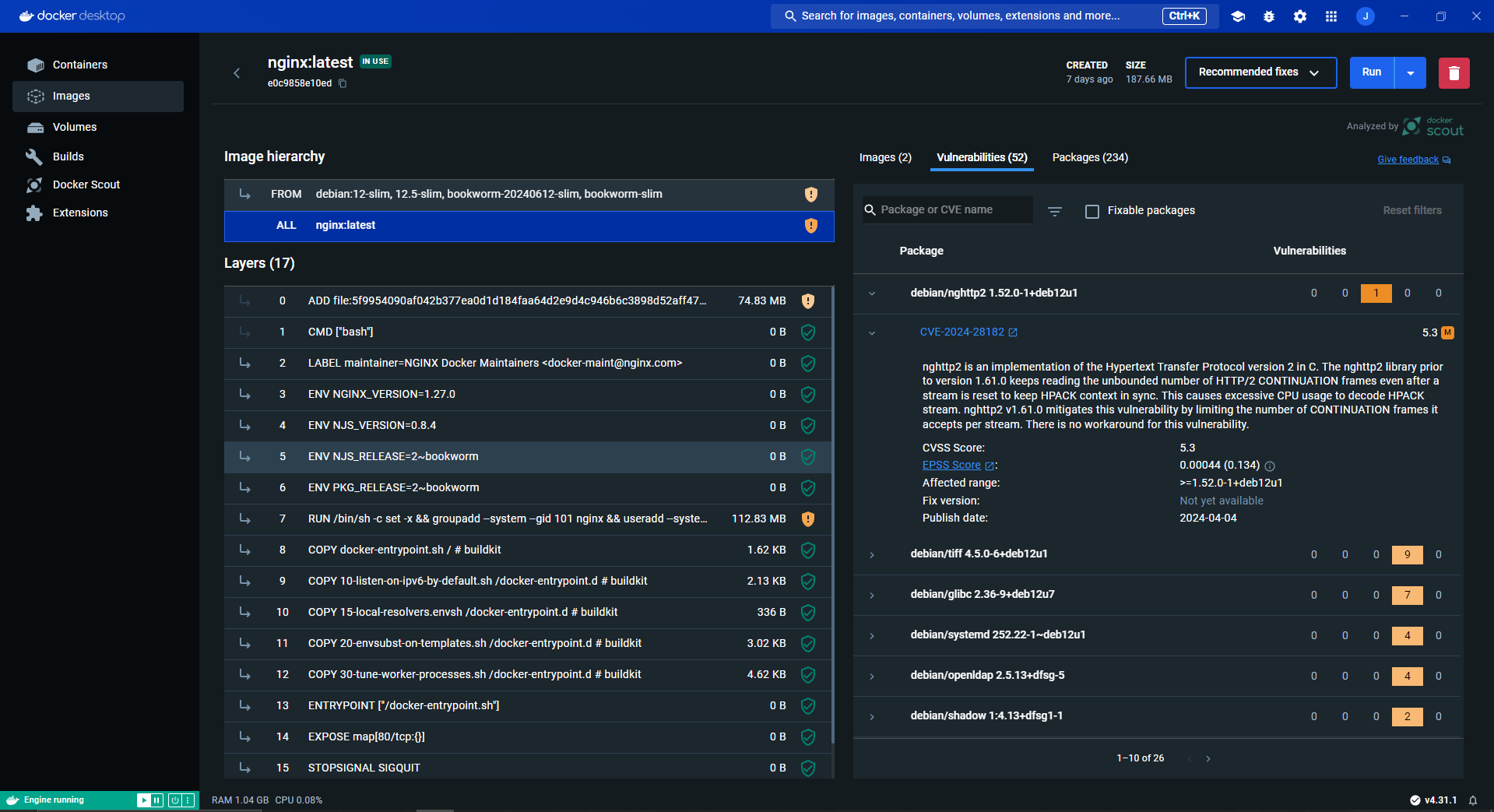
- Устранение уязвимости:

Обновить pip до последней версии. Новые версии pip могут содержать исправления.

pip install --upgrade pip

3. В контейнере Nginx

- Общий результат(Уязвимость 1)



- Наиболее значимые уязвимости по оценкам сканера

- Уязвимость 1

- Код уязвимости:

CVE-2024-28182 ⁠

- Описание уязвимости:

nghttp2 — это реализация протокола передачи гипертекста версии 2 на C. Библиотека nghttp2 до версии 1.61.0 продолжает читать неограниченное количество кадров ПРОДОЛЖЕНИЯ HTTP/2 даже после сброса потока, чтобы поддерживать синхронизацию контекста HPACK. Это приводит к чрезмерной загрузке ЦП для декодирования потока HPACK. nghttp2 v1.61.0 смягчает эту уязвимость, ограничивая количество кадров ПРОДОЛЖЕНИЯ, которые он принимает на поток. Обхода этой уязвимости не существует.

- Устранение уязвимости:

Запретить использование в ОС пакета nghttp2 или установить обновление для пакета(ов) nghttp2