|  |  |  |
| --- | --- | --- |
|  |  |  |
| МИНОБРНАУКИ РОССИИ | | |
| Федеральное государственное бюджетное образовательное учреждение  высшего образования  **«МИРЭА – Российский технологический университет»**  **РТУ МИРЭА** | | |
| **Институт кибербезопасности и цифровых технологий** | | |
|  |  |  |

Кафедра КБ-3 «Безопасность программных решений»

**Отчёт**

по выполнению практического задания по Docker

по дисциплине

«Разработка безопасного программного обеспечения»

Выполнил:

Студент 4 курса

Еланцев Артём

Группа БCБО-02-20

Проверил:

Латыпов И. Т.

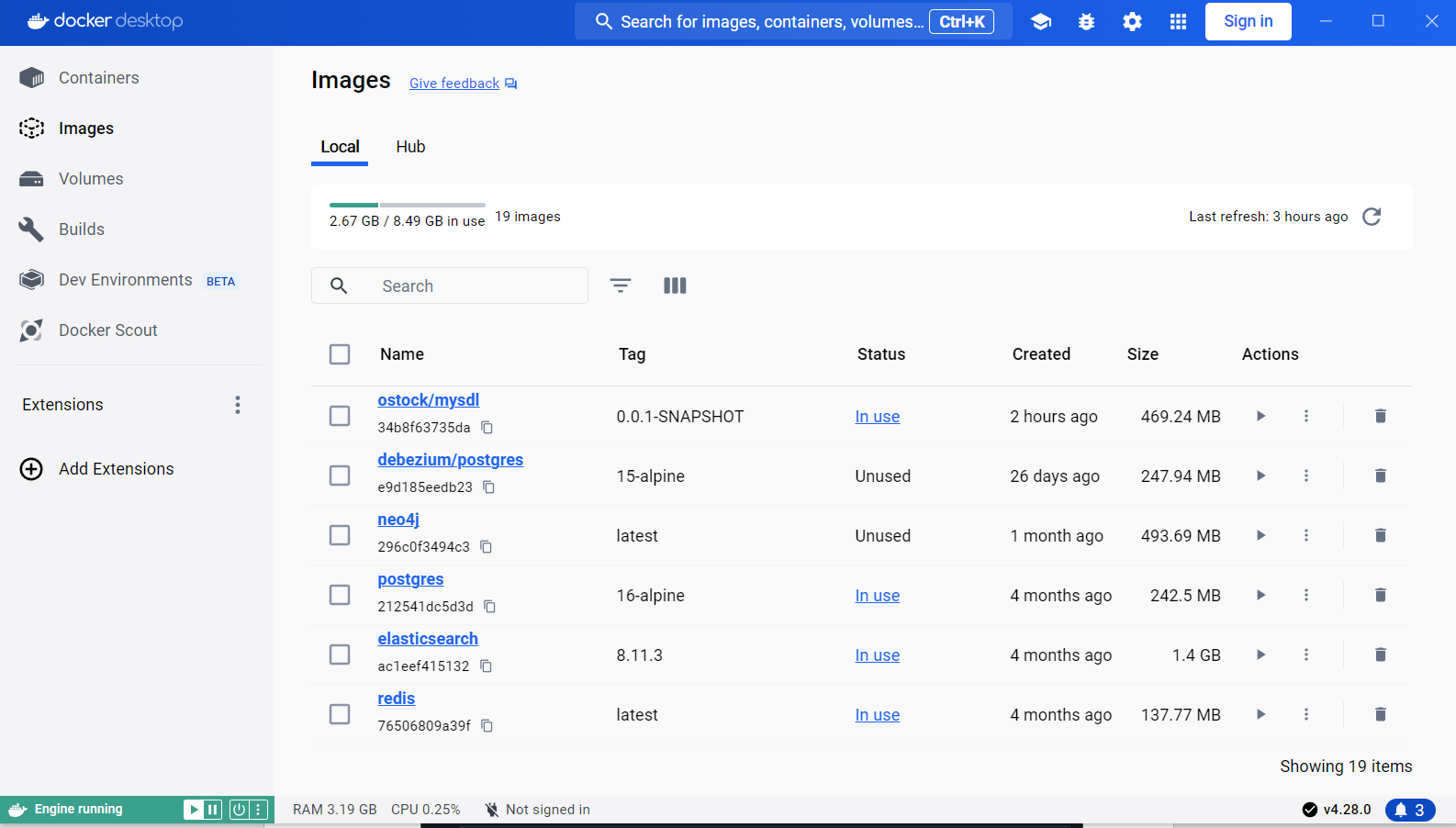
Москва, 2024

**Задание**

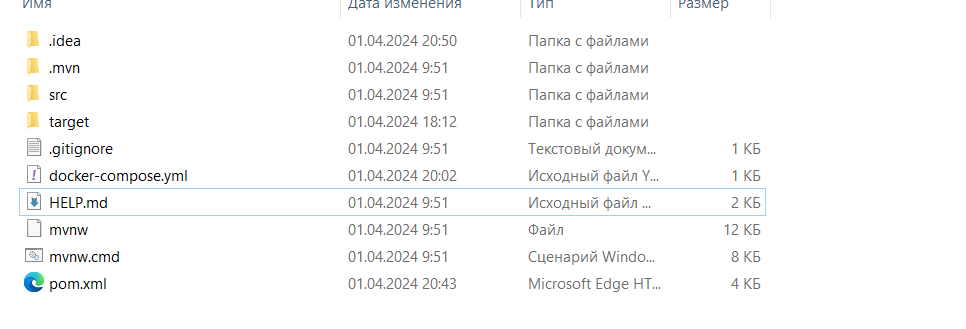
Необходимо создать сайт внутри контейнера, который отображает ФИО, номер группы, написать отчет о выполненных действия со скриншотами. Проверить безопасность контейнеров с использованием утилит и написать вывод в тот же отчет о найденных уязвимостях и предложить исправления

**Ход работы**

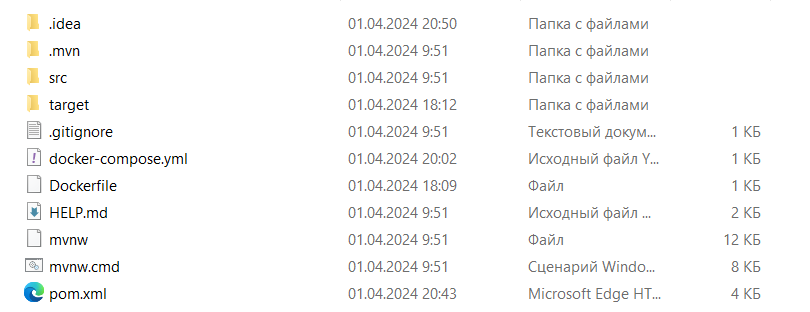
1. Установить Docker компьютер.



1. После установки Docker создаем новую папку проекта на компьютере.



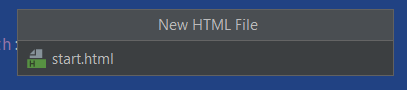
1. Создаем Dockerfile в корневом каталоге проекта.



1. Добавляем в Dockerfile следующие строки:

FROM openjdk:21-slim  
  
ARG *JAR\_FILE*COPY ${*JAR\_FILE*} app.jar  
  
ENTRYPOINT ["java","-jar","/app.jar"]  
  
WORKDIR /app

1. Создаем файл start.html в папке «templates» в корневом каталоге проекта.

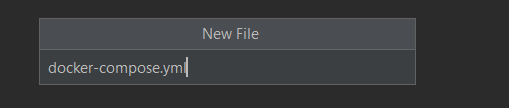


1. Добавляем следующий код в файл start.html:

<!DOCTYPE html>  
<html lang="en">  
<head>  
 <meta charset="UTF-8">  
 <title>Title</title>  
</head>  
<body>  
<form method="POST">  
 <input type="text" name="id">  
 <input type="text" name="fullName">  
 <input type="submit"/>  
</form>  
  
<h1>Данные из базы</h1>  
<table>  
  
 <tr>  
 <th>Key</th>  
 <th>Value</th>  
 </tr>  
 <tr th:each="item : ${students}">  
 <td>  
 <a th:text="${item.getId()}">a</a>  
 </td>  
 <td>  
 <a th:text="${item.getFullName()}">a</a>  
 </td>  
 <br>  
 </tr>  
</table>  
  
</body>  
</html>

Этот код устанавливает HTML-шаблон, отображающий список учащихся, полученный из Redis.

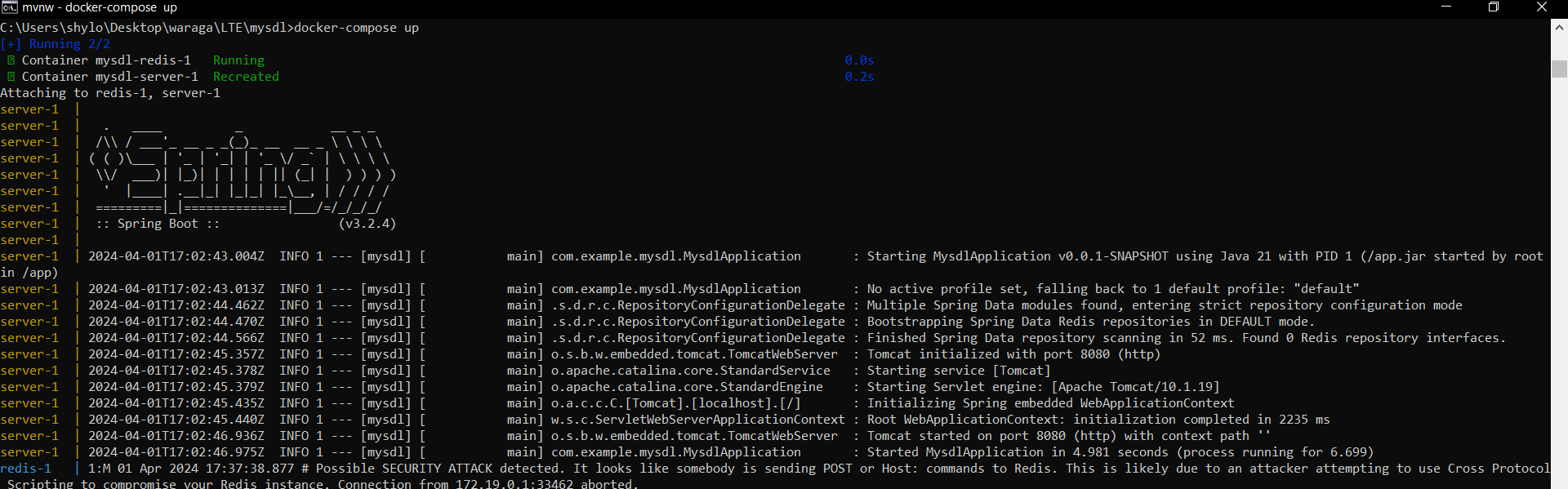
1. Создаем файл docker-compose.yml в корневом каталоге проекта.



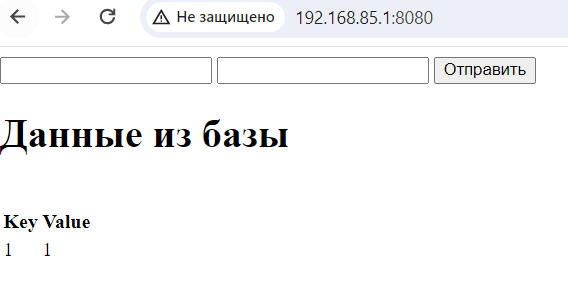
1. Добавляем следующий код в файл docker-compose.yml:

version: '3'  
services:  
 server:  
 image: ostock/mysdl:0.0.1-SNAPSHOT  
 ports:  
 - "80:8081"  
 depends\_on:  
 - redis  
  
 redis:  
 image: "redis:latest"  
 ports:  
 - "6379:6379"

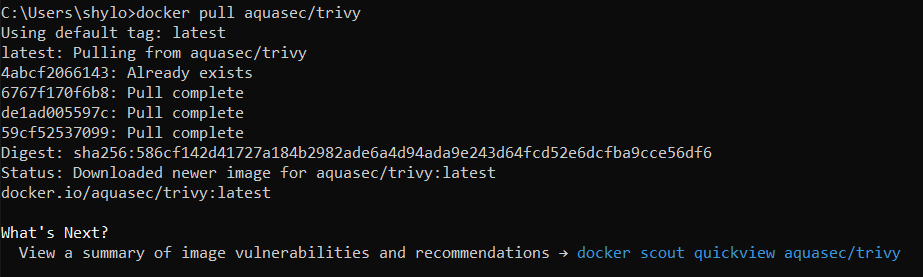
1. Открываем терминал и переходим в корневой каталог проекта.
2. Выполняем команду docker-compose up, чтобы запустить контейнеры Docker.



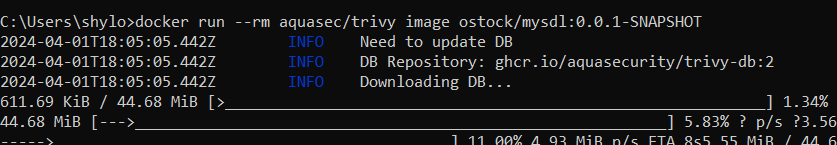
1. Открываем веб-браузер и проверяем.



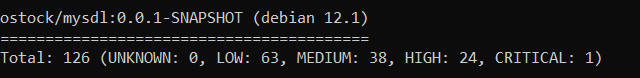
1. Для проверки безопасности контейнеров мы воспользуемся утилитой Trivy. Для установки его в Docker контейнер воспользуемся следующей командой:



1. Запустим сканирование: после того, как мы установим Trivy, начнем использовать его для сканирования образов Docker на наличие уязвимостей. Для этого нам нужно указать имя образа, которое мы хотим отсканировать:



1. Получаем следующий отчет:



Рассмотрим одну критическую ошибку и одну высокую ошибку.

CVE-2023-45853 – Critical

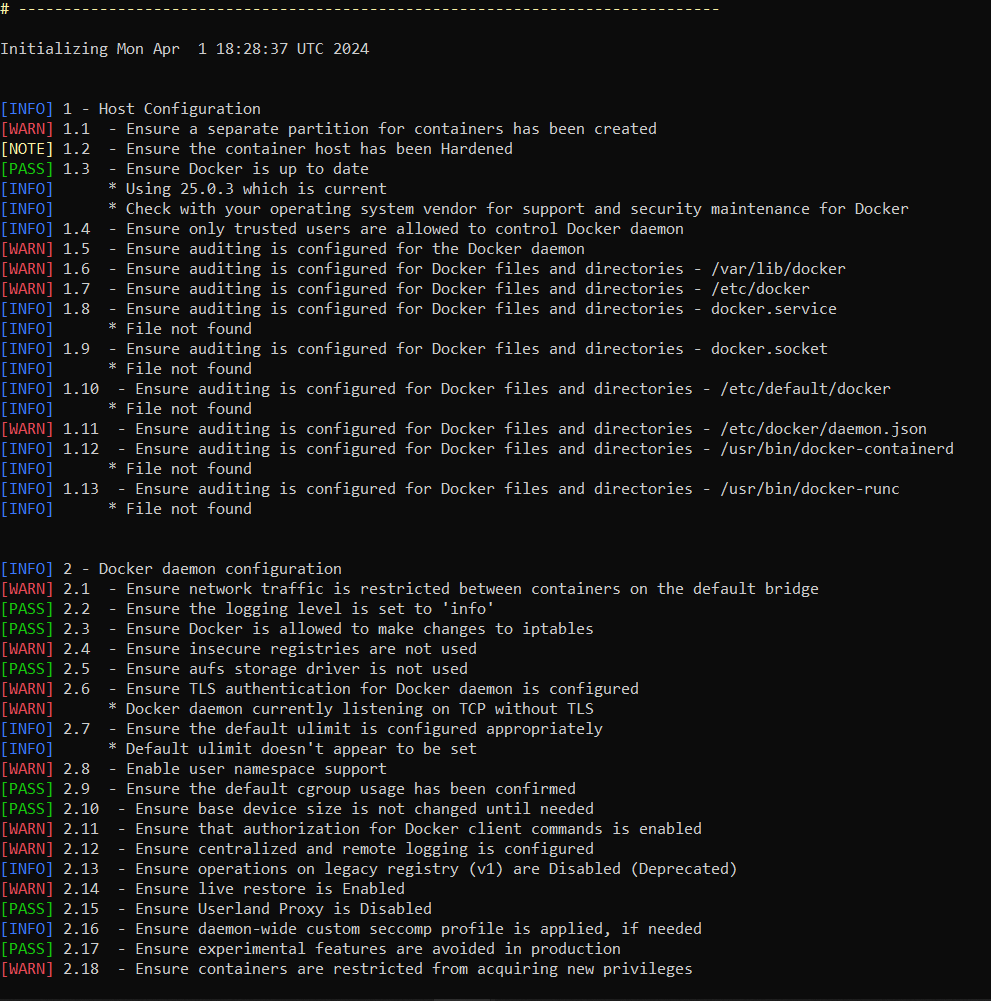
MiniZip в zlib до версии 1.3 имеет переполнение целых чисел и результирующее переполнение буфера на основе кучи в zipOpenNewFileInZip4\_64 из-за длинного имени файла, комментария или дополнительного поля. ПРИМЕЧАНИЕ: MiniZip не является поддерживаемой частью продукта zlib. ПРИМЕЧАНИЕ: pyminizip через 0.2.6 также уязвим, поскольку он объединяет уязвимую версию zlib и предоставляет соответствующий код MiniZip через свой compress API.

CVE-2022-3715 – High

В пакете bash обнаружена уязвимость, из-за которой может произойти переполнение cache-буфера в допустимом параметре parameter\_transform. Эта проблема может привести к проблемам с памятью. Решения этой проблемы пока нет, нужно ждать обновления bash.

Большинство других ошибок указывают на уязвимости Linux, которые могут быть исправлены обновлением различных пакетов.

1. Проверим наш контейнер через Docker Bench Security:



1. Running as root: flaskproject2-web-1 – показывает, что мы запускаем наш проект с помощью root-прав. Для решения нужно настроить отдельного пользователя с нужными только для этого приложения привилегии.
2. No Healthcheck found: [flaskproject2-web:latest] – указывает на то, что у нас нет проверки работоспособности контейнера, соответственно мы не сможем его перезапускать, если его не добавим. Для решения нужно в Dockerfile добавить следующую команду, которая будет, грубо говоря, пинговать сайт:

HEALTHCHECK CMD curl --fail http://localhost:5000/ || exit 1

1. No AppArmorProfile Found: flaskproject2-web-1 – нужен для отслеживания профилей и допустимых к ним привилегиям.
2. No SecurityOptions Found: flaskproject2-web-1 – не установлен SELinux, который нужен для принудительного контроля доступа к компонентам системы.
3. Container running without memory restrictions: flaskproject2-web-1 – мы не установили ограничение потребляемой памяти, из-за чего контейнер может есть памяти больше, чем ему надо.
4. Container running without CPU restrictions: flaskproject2-web-1 - мы не установили ограничение потребления CPU, из-за чего контейнер может потреблять больше мощности, чем ему надо.
5. Container running with root FS mounted R/W: flaskproject2-web-1 – контейнер запущен с режимом Write. Если нам не нужно что-либо записывать, достаточно просто поставить в read-only.
6. Port being bound to wildcard IP: 0.0.0.0 in flaskproject2-web-1 – установлена wildcard на все сервера, из-за чего доступ к нему может получить любое устройство в сети
7. MaximumRetryCount is not set to 5: flaskproject2-web-1 – не установлен параметр перезапуска, из-за чего контейнер может перезапускаться с ошибкой постоянно.
8. PIDs limit not set: flaskproject2-web-1 – не установлен лимит на процессы, из-за злоумышленник может запускать все, что ему нужно. Достаточно ограничить процессы на то количество, которое необходимо контейнеру.

**Листинг программы**

**HomeController.java**

package com.example.mysdl;  
  
import org.springframework.stereotype.Controller;  
import org.springframework.ui.Model;  
import org.springframework.web.bind.annotation.GetMapping;  
import org.springframework.web.bind.annotation.PostMapping;  
import org.springframework.web.bind.annotation.RequestMapping;  
  
@Controller  
@RequestMapping("/")  
public class HomeController {  
 private RedisStudentRepository repository;  
  
 public HomeController(RedisStudentRepository repository) {  
 this.repository = repository;  
 }  
  
 @GetMapping  
 public String getHome(Model model) {  
 model.addAttribute("students", repository.findAll());  
 return "start";  
 }  
  
 @PostMapping("/")  
 public String addStudent(Student student) {  
 repository.save(student);  
 return "redirect:/";  
 }  
}

**docker-compose.yml**

version: '3'  
services:  
 server:  
 image: ostock/mysdl:0.0.1-SNAPSHOT  
 ports:  
 - "80:8081"  
 depends\_on:  
 - redis  
  
 redis:  
 image: "redis:latest"  
 ports:  
 - "6379:6379"

**Dockerfile**

FROM openjdk:21-slim  
  
ARG *JAR\_FILE*COPY ${*JAR\_FILE*} app.jar  
  
ENTRYPOINT ["java","-jar","/app.jar"]  
  
WORKDIR /app

**start.html**

<!DOCTYPE html>  
<html lang="en">  
<head>  
 <meta charset="UTF-8">  
 <title>Title</title>  
</head>  
<body>  
<form method="POST">  
 <input type="text" name="id">  
 <input type="text" name="fullName">  
 <input type="submit"/>  
</form>  
  
<h1>Данные из базы</h1>  
<table>  
  
 <tr>  
 <th>Key</th>  
 <th>Value</th>  
 </tr>  
 <tr th:each="item : ${students}">  
 <td>  
 <a th:text="${item.getId()}">a</a>  
 </td>  
 <td>  
 <a th:text="${item.getFullName()}">a</a>  
 </td>  
 <br>  
 </tr>  
</table>  
  
</body>  
</html>