



МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение  
высшего образования

**«МИРЭА – Российский технологический университет»  
РТУ МИРЭА**

---

Институт Искусственного Интеллекта  
Базовая кафедра БК252

---

## **ПРАКТИЧЕСКАЯ РАБОТА**

По дисциплине «Основы построения защищённых компьютерных систем»

Студент группы ККСО-03-19                      Николенко В.О.

Студент группы ККСО-03-19                      Воеводин К.А.

Студент группы ККСО-03-19                      Митин М.Д.

Студент группы ККСО-03-19                      Базарова Е.О.

Студент группы ККСО-03-19                      Никишина А.А.

Преподаватель                                      Жанкевич А.О.

Москва – 2023

# СОДЕРЖАНИЕ

1. Введение . . . . .	3
2. Описание проекта . . . . .	4
2.1. Задачи практической работы. . . . .	4
2.2. Утилита Nadolint. . . . .	4
2.3. Каковы преимущества использования Nadolint по сравнению с другими линтерами Dockerfile? . . . . .	5
2.4. Каковы некоторые распространенные проблемы, которые Nadolint может обнаружить в Dockerfiles? . . . . .	6
2.5. Dockle. . . . .	7
2.6. Ключевые возможности Dockle. . . . .	7
2.7. Trivy. . . . .	8
2.8. Ключевые возможности Trivy. . . . .	8
2.9. Сравнительная таблица исследуемых утилит. . . . .	9
3. Практическая часть . . . . .	11
3.1. Проверка корректности и безопасности инструкций Dockerfile утилитой линтером Nadolint. . . . .	11
3.2. Проверка корректности и безопасности конечного и промежу- точных образов - утилитой Dockle. . . . .	12
3.3. Проверка наличия общеизвестных уязвимостей (CVE) в базо- вом образе и ряде зависимостей - утилитой Trivy. . . . .	15
4. Заключение . . . . .	18

# 1. ВВЕДЕНИЕ

Docker - это платформа для разработки, доставки и запуска приложений в контейнерах. Контейнеры обеспечивают изоляцию приложений и их зависимостей, что позволяет легко упаковывать и переносить приложения между различными окружениями. Docker стал широко используемым инструментом в индустрии разработки программного обеспечения благодаря своей простоте, масштабируемости и повторяемости среды исполнения.

Однако, с ростом популярности Docker и использования контейнеров возникают новые вызовы в области безопасности. Виртуализация на уровне операционной системы, характерная для Docker, предоставляет высокую гибкость и эффективность, но может также вести к уязвимостям, если контейнеры не настроены и не управляются должным образом.

Необходимо активно исследовать возможности обеспечения безопасности Docker-образов, чтобы предотвратить нарушения, утечки данных, атаки и другие потенциальные проблемы безопасности. В этом контексте проверка безопасности и корректности Docker-образов становится критическим шагом в процессе разработки и развертывания контейнеризованных приложений.

В данном отчете мы представляем результаты проверки Docker-образов с помощью трех утилит: Hadolint, Dockle и Trivy. Hadolint используется для проверки корректности и безопасности инструкций Dockerfile, Dockle помогает анализировать конечные и промежуточные образы на предмет потенциальных проблем безопасности, а Trivy осуществляет сканирование образов и зависимостей на предмет известных уязвимостей (CVE).

Цель данного исследования состоит в выявлении и устранении потенциальных уязвимостей, ошибок конфигурации и проблем безопасности в Docker-образах. Результаты проверки и анализ, представленные в данном отчете, помогут разработчикам и администраторам повысить безопасность и надежность контейнеризованных приложений.

## 2. ОПИСАНИЕ ПРОЕКТА

### 2.1. Задачи практической работы.

- 1) Проверка корректности и безопасности инструкций Dockerfile - утилитой линтером Hadolint.
- 2) Проверка корректности и безопасности конечного и промежуточных образов - утилитой Dockle.
- 3) Проверка наличия общеизвестных уязвимостей (CVE) в базовом образе и ряде зависимостей - утилитой Trivy.

### 2.2. Утилита Hadolint.

Hadolint - это утилита для статического анализа Dockerfile, которая помогает обнаруживать ошибки и проблемы в описании образа Docker. Hadolint использует набор правил (rule set) для проверки Dockerfile на соответствие лучшим практикам и рекомендациям по использованию Docker.

Hadolint проверяет Dockerfile на наличие следующих типов проблем:

- Нарушение правил форматирования и оформления Dockerfile
- Использование устаревших и небезопасных команд и параметров в Dockerfile
- Нарушение рекомендаций по масштабированию и оптимизации образов Docker
- Использование неэффективных и неоптимальных методов кэширования образов Docker
- Нарушение правил по наименованию и версионированию образов Docker

Hadolint может использоваться как локально, так и в процессе непрерывной интеграции (CI) для автоматической проверки Dockerfile на соответствие стандартам и правилам. Hadolint поддерживает многие популярные CI/CD-системы, включая Jenkins, Travis CI, CircleCI и другие.

Hadolint доступен для установки на различные операционные системы и платформы, включая Linux, macOS и Windows. Hadolint также доступен в виде Docker-образа, что позволяет использовать его в контейнерной среде.

### 2.3. Каковы преимущества использования Hadolint по сравнению с другими линтерами Dockerfile?

Hadolint имеет ряд преимуществ по сравнению с другими линтерами Dockerfile:

Всеобъемлющий набор правил: Hadolint имеет всеобъемлющий набор правил, который охватывает широкий спектр лучших практик и рекомендаций по разработке Dockerfile. Набор правил основан на отраслевых стандартах и руководствах, таких как Руководство по лучшим практикам Dockerfile и CIS Docker Benchmark, и регулярно обновляется, чтобы отразить изменения в экосистеме Docker.

Быстрый и эффективный: Hadolint разработан таким образом, чтобы быть быстрым и эффективным даже при анализе больших файлов Dockerfile или больших кодовых баз. Он написан на Haskell, высокопроизводительном языке программирования, и использует высокооптимизированный анализатор для быстрого анализа файлов Dockerfile и выявления проблем.

Настраиваемый: Hadolint легко настраивается и позволяет пользователям определять свои собственные правила и исключения. Это позволяет легко адаптировать инструмент к конкретным требованиям проекта и стилям кодирования.

Интеграция с CI / CD: Hadolint легко интегрируется с популярными системами CI / CD, такими как Jenkins, Travis CI и CircleCI, позволяя разработчикам включать компоновку Dockerfile в свои рабочие процессы непрерывной интеграции.

Кроссплатформенная поддержка: Hadolint является кроссплатформенным и может использоваться на компьютерах с Linux, macOS и Windows. Он также может быть запущен как контейнер Docker, что упрощает интеграцию в рабочие процессы разработки и развертывания на основе контейнеров.

В целом, Hadolint - это мощный и гибкий инструмент для компоновки файлов Dockerfile, который может помочь разработчикам улучшить качество и согласованность своих изображений Docker.

## 2.4. Каковы некоторые распространенные проблемы, которые Hadolint может обнаружить в Dockerfiles?

Hadolint может обнаруживать широкий спектр проблем в Dockerfiles, включая:

Проблемы безопасности: Hadolint может обнаруживать использование небезопасных или устаревших команд и конфигураций в файлах Dockerfiles, таких как использование `latest` тега, использование `RUN apt-get upgrade or apt-get dist-upgrade` и использование `ADD or COPY` с удаленными URL.

Нарушения рекомендаций: Hadolint может обнаруживать нарушения рекомендаций Dockerfile, таких как использование нескольких `RUN` команд вместо одной `RUN` с использованием нескольких команд, использование `EXPOSE` без указания протокола и использование `ENV` для установки учетных данных или конфиденциальной информации.

Проблемы с эффективностью: Hadolint может обнаруживать недостатки в файлах Dockerfiles, которые могут привести к увеличению времени сборки или увеличению размеров изображений, такие как использование `apt-get update` без очистки кэша, использование `ADD or COPY` с большими файлами или каталогами и использование `RUN` команд, которые генерируют большие временные файлы.

Проблемы с сопровождающим: Hadolint может обнаруживать проблемы, связанные с обслуживанием и документированием файлов Dockerfiles, такие как отсутствие метки сопровождающего, использование нестандартных меток и использование неописательных или вводящих в заблуждение названий изображений.

В целом, Hadolint предоставляет мощный инструмент для обнаружения и исправления проблем в файлах Dockerfiles, помогая гарантировать, что образы Docker безопасны, эффективны и хорошо документированы.

## 2.5. Dockle.

Dockle - это утилита командной строки, которая используется для автоматической проверки безопасности и корректности Docker-образов. Она позволяет обнаруживать потенциальные проблемы в образах, такие как уязвимости в компонентах, ошибки конфигурации и другие нарушения стандартов безопасности.

При работе с Dockle можно использовать настраиваемые правила, которые определяют, какие проверки должны быть выполнены. Например, можно настроить правила для проверки наличия установленных обновлений безопасности, выполнения образа от непривилегированного пользователя и других настроек безопасности.

Dockle использует базу данных уязвимостей, которая содержит информацию о известных уязвимостях в компонентах, используемых в Docker-образах, таких как операционные системы, библиотеки и программные пакеты. Эта база данных обновляется регулярно, чтобы включать в себя новые уязвимости и исправления.

При запуске проверки Dockle анализирует содержимое Docker-образа и сверяет его с настроенными правилами. После завершения проверки Dockle выдает отчет о найденных проблемах и рекомендациях по их устранению. Отчет может быть выведен в различных форматах, таких как JSON, JUnit и Text.

Dockle может быть использована на разных этапах жизненного цикла Docker-образов. Например, можно запускать проверку на этапе разработки, чтобы обнаружить проблемы еще до того, как образ будет загружен в репозиторий Docker. Также можно настроить проверку на этапе CI/CD, чтобы автоматически проверять образы на соответствие стандартам безопасности и корректности перед их развертыванием в production-среде.

## 2.6. Ключевые возможности Dockle.

- Анализ Docker-образов на наличие уязвимостей и ошибок конфигурации
- Поддержка настраиваемых правил и рекомендаций
- Интеграция с CI/CD системами
- Поддержка различных форматов вывода результатов

Использование Dockle позволяет повысить уровень безопасности и качества Docker-образов, что в свою очередь способствует улучшению производительности и уменьшению рисков возникновения проблем при работе приложений, развернутых в Docker-контейнерах.

## 2.7. Trivy.

Trivy - это мощный инструмент для проверки безопасности контейнеров, который используется для анализа Docker-образов и обнаружения общеизвестных уязвимостей (CVE) в базовом образе и зависимостях, включенных в образ. Он может быть использован на разных этапах жизненного цикла Docker-образов, включая этап разработки, тестирования и production-среды.

Trivy использует базу данных уязвимостей, которая содержит информацию о общеизвестных уязвимостях (CVE) в компонентах, используемых в Docker-образах, таких как операционные системы, библиотеки и программные пакеты. База данных обновляется регулярно, чтобы включать в себя новые уязвимости и исправления.

Одной из ключевых особенностей Trivy является поддержка проверки зависимостей, включенных в Docker-образ. Это позволяет обнаруживать уязвимости, которые могут быть связаны с использованием конкретных версий библиотек или программных пакетов, включенных в образ.

Trivy поддерживает различные форматы вывода результатов, такие как JSON, JUnit и Text, что облегчает интеграцию с другими инструментами. Это позволяет использовать Trivy в сочетании с другими инструментами для автоматического обнаружения уязвимостей в Docker-образах и принятия мер для их устранения.

Существует несколько способов использования Trivy. Он может быть использован в интерактивном режиме, что позволяет пользователю запускать проверку на конкретном образе и получать отчет о найденных уязвимостях. Также Trivy может быть интегрирован в систему непрерывной интеграции и непрерывного развертывания (CI/CD) для автоматической проверки образов при каждом обновлении кодовой базы.

## 2.8. Ключевые возможности Trivy.

- Анализ Docker-образов на наличие общеизвестных уязвимостей (CVE)
- Поддержка базы данных уязвимостей, которая содержит информацию о новых уязвимостях и исправлениях
- Проверка зависимостей, включенных в Docker-образ
- Поддержка различных форматов вывода результатов

Использование Trivy помогает повысить уровень безопасности и качества Docker-образов, что в свою очередь способствует улучшению производительности и уменьшению рисков возникновения проблем при работе приложений, развернутых в Docker-контейнерах.



## 2.9. Сравнительная таблица исследуемых утилит.

Название	Преимущества	Недостатки
Nadolint	<ul style="list-style-type: none"><li>• Помогает выявить потенциальные проблемы в Dockerfile, что может привести к улучшению безопасности и качества Docker-образов;</li><li>• Позволяет улучшить структуру и содержание Dockerfile, что может привести к уменьшению вероятности возникновения ошибок и уязвимостей в Docker-образах.</li></ul>	<ul style="list-style-type: none"><li>• Не проверяет Docker-образы на наличие уязвимостей и ошибок конфигурации, что ограничивает возможности обнаружения проблем в Docker-образах.</li></ul>
Dockle	<ul style="list-style-type: none"><li>• Проверяет Docker-образы на соответствие стандартам безопасности и корректности, что может привести к улучшению безопасности и качества Docker-образов;</li><li>• Позволяет обнаруживать уязвимости и ошибки конфигурации в Docker-образах, что может привести к уменьшению вероятности возникновения проблем при работе приложений, развернутых в Docker-контейнерах.</li></ul>	<ul style="list-style-type: none"><li>• Не проверяет Dockerfile на соответствие рекомендациям по его структуре и содержанию, что может привести к возникновению проблем при создании Docker-образов.</li></ul>

Trivy	<ul style="list-style-type: none"> <li>• Обнаруживает общеизвестные уязвимости (CVE) в Docker-образах и их зависимостях, что может привести к улучшению безопасности и качества Docker-образов;</li> <li>• Позволяет обнаруживать уязвимости в зависимостях, включенных в Docker-образ, что может привести к уменьшению вероятности возникновения проблем при работе приложений, развернутых в Docker-контейнерах.</li> </ul>	<ul style="list-style-type: none"> <li>• Не проверяет Dockerfile на соответствие рекомендациям по его структуре и содержанию, что может привести к возникновению проблем при создании Docker-образов;</li> <li>• Не покрывает все возможные уязвимости, что означает, что некоторые уязвимости могут остаться незамеченными.</li> </ul>
-------	---	---

### 3. ПРАКТИЧЕСКАЯ ЧАСТЬ

#### 3.1. Проверка корректности и безопасности инструкций Dockerfile утилитой линтером Nadolint.

Для проверки корректности и безопасности инструкций Dockerfile использовалась утилита Nadolint. Ниже приведены подробности выполненных действий:

```
grandf17@ubuntu:~/nginx-image$ docker build -t nginx_image .
[+] Building 67.4s (13/13) FINISHED
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 1.47kB
=> [internal] load .dockerignore
=> => transferring context: 2B
=> [internal] load metadata for docker.io/library/ubuntu:20.04
=> [1/8] FROM docker.io/library/ubuntu:20.04@sha256:f8f658407c35733471596f25fdb4ed748b80e545ab57e84efbdb1dbbb01bd70e
=> => resolve docker.io/library/ubuntu:20.04@sha256:f8f658407c35733471596f25fdb4ed748b80e545ab57e84efbdb1dbbb01bd70e
=> => sha256:f8f658407c35733471596f25fdb4ed748b80e545ab57e84efbdb1dbbb01bd70e 1.13kB / 1.13kB
=> => sha256:554e40b15453c788ec799badf8f1ad05c3e5c735b53f940Feb8f27cf2ec570c5 424B / 424B
=> => sha256:626a42b93d93241a6a48d81d921934891f73185547833a64dde06599cf3eafc2 2.30kB / 2.30kB
=> => sha256:56e0351b98767487b3c411034be95479ed1710bb6be86db6df0be3a98653027 27.51MB / 27.51MB
=> => extracting sha256:56e0351b98767487b3c411034be95479ed1710bb6be86db6df0be3a98653027
=> [internal] load build context
=> => transferring context: 1.72kB
=> [2/8] RUN apt update
=> [3/8] RUN apt install -y nginx php-fpm supervisor && rm -rf /var/lib/apt/lists/* && apt clean
=> [4/8] COPY default /etc/nginx/sites-available/default
=> [5/8] RUN sed -i -e 's/exitfix_pathinfo=1/cgi.fix_pathinfo=0/g' /etc/php/7.4/fpm/php.ini && echo "\ndaemon off;" >> /etc/nginx/nginx.conf
=> [6/8] COPY supervisord.conf /etc/supervisor/supervisord.conf
=> [7/8] RUN mkdir -p /run/php && chown -R www-data:www-data /var/www/html && chown -R www-data:www-data /run/php
=> [8/8] COPY start.sh /start.sh
=> => exporting to image
=> => exporting layers
=> => writing image sha256:644b7c4381308e925bca548978909baba0a578312e77f915566d64722e9fe98b
=> => naming to docker.io/library/nginx_image
```

Рис. 1. Сборка docker-контейнера при помощи Dockerfile

На рис.1 представлен процесс сборки docker-контейнера с использованием Dockerfile. Была выполнена команда

```
docker build -t nginx_image
```

которая основывается на нашем Dockerfile и создает образ с тегом "nginx\_image".

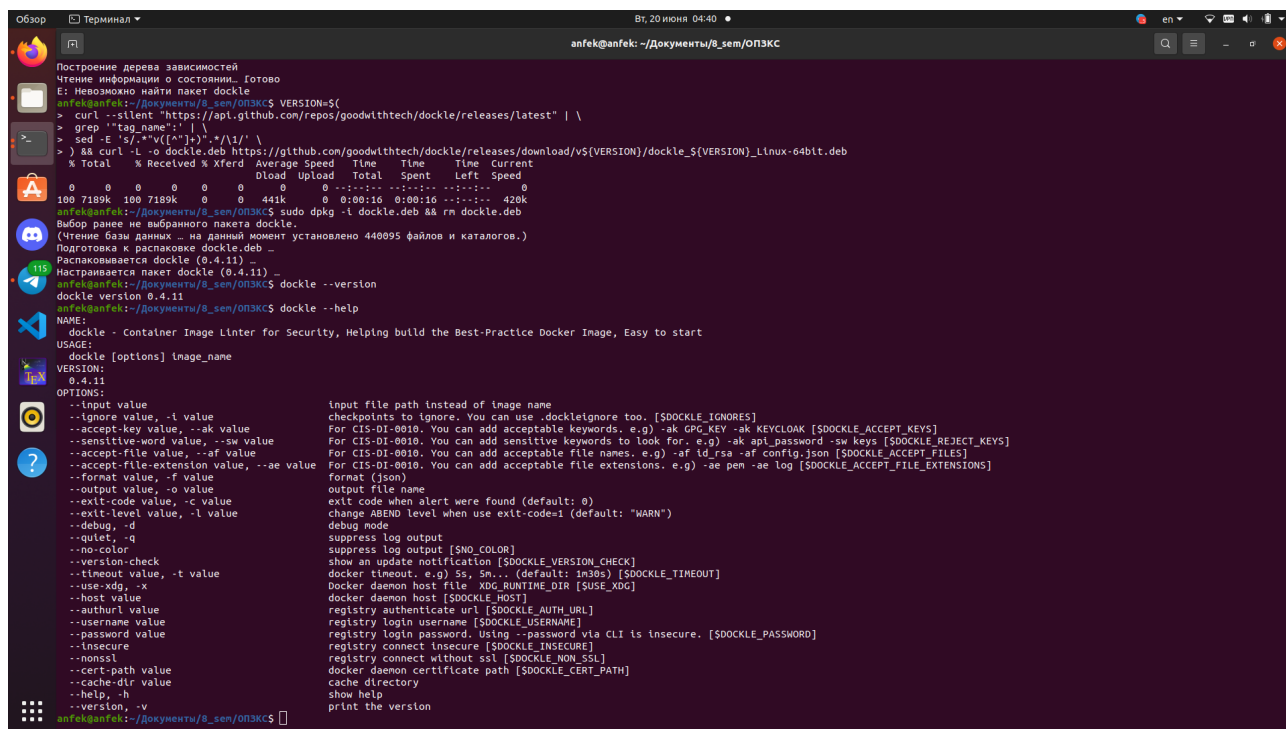
```
grandf17@ubuntu:~/nginx-image$ docker run --rm -i hadolint/hadolint < Dockerfile
--:14 DL3027 warning: Do not use apt as it is meant to be a end-user tool, use apt-get or apt-cache instead
--:17 DL3027 warning: Do not use apt as it is meant to be a end-user tool, use apt-get or apt-cache instead
--:29 SC2028 info: echo may not expand escape sequences. Use printf.
```

Рис. 2. Проверка Dockerfile при помощи Nadolint

На рис.2 показана проверка Dockerfile с помощью утилиты Nadolint. Была выполнена команда, которая сканирует Dockerfile на наличие ошибок и предоставляет рекомендации по улучшению.

## 3.2. Проверка корректности и безопасности конечного и промежуточных образов - утилитой Dockle.

Для проверки корректности и безопасности конечных и промежуточных Docker-образов использовалась утилита Dockle. Ниже приведены подробности выполненных действий:



```
Обзор Терминал 20 июня 04:40 anfe@anfe: ~/Документы/8_sen/ОПЗКС

Построение дерева зависимостей
Чтение информации о состоянии... Готово
E: Невозможно найти пакет dockle
anfe@anfe:~/Документы/8_sen/ОПЗКС$ VERSION=$(
> curl --silent "https://api.github.com/repos/goodwithtech/dockle/releases/latest" | \
> grep "tag_name:" | \
> sed -E 's/.*"v([^\"]+)"$/\1/' \
> ) && curl -L -o dockle.deb https://github.com/goodwithtech/dockle/releases/download/v$VERSION/dockle_${VERSION}_Linux-64bit.deb
% Total % Received % Xferd Average Speed Time Time Time Current
                                Dload Upload Total Spent Left Speed
100 719K 100 719K 0 0 441K 0 0:00:16 0:00:16 --:--:-- 420K
anfe@anfe:~/Документы/8_sen/ОПЗКС$ sudo dpkg -i dockle.deb && rm dockle.deb
Выбор ранее не выбранного пакета dockle.
(Чтение баз данных ... на данный момент установлено 440995 файлов и каталогов.)
Подготовка к распаковке dockle.deb ...
Распаковывается dockle (0.4.11) ...
Настраивается пакет dockle (0.4.11) ...
anfe@anfe:~/Документы/8_sen/ОПЗКС$ dockle --version
dockle version 0.4.11
anfe@anfe:~/Документы/8_sen/ОПЗКС$ dockle --help
NAME:
dockle - Container Image Linter for Security, Helping build the Best-Practice Docker Image, Easy to start
USAGE:
dockle [options] image_name
VERSION:
0.4.11
OPTIONS:
--input value          input file path instead of image name
--ignore value, -i value checkpoints to ignore. You can use .dockleignore too. [SDOCKLE_IGNORES]
--accept-key value, --ak value For CIS-DI-0010. You can add acceptable keywords. e.g) -ak GPG_KEY -ak KEYCLOAK [SDOCKLE_ACCEPT_KEYS]
--sensitive-word value, --sw value For CIS-DI-0010. You can add sensitive keywords to look for. e.g) -ak apt_password -sw keys [SDOCKLE_REJECT_KEYS]
--accept-file value, --af value For CIS-DI-0010. You can add acceptable file names. e.g) -af id_rsa -af config.json [SDOCKLE_ACCEPT_FILES]
--accept-file-extension value, --ae value For CIS-DI-0010. You can add acceptable file extensions. e.g) -ae pem -ae log [SDOCKLE_ACCEPT_FILE_EXTENSIONS]
--format value, -f value format (json)
--output value, -o value output file name
--exit-code value, -c value exit code when alert were found (default: 0)
--exit-level value, -l value change ABEND level when use exit-code=1 (default: "WARN")
--debug, -d debug mode
--quiet, -q suppress log output
--no-color suppress log output [SNO_COLOR]
--version-check show an update notification [SDOCKLE_VERSION_CHECK]
--timeout value, -t value docker timeout. e.g) 5s, 5m... (default: 1m30s) [SDOCKLE_TIMEOUT]
--use-xdg, -x Docker daemon host file. XDG_RUNTIME_DIR [SUSE_XDG]
--host value docker daemon host [SDOCKLE_HOST]
--authurl value registry authenticate url [SDOCKLE_AUTH_URL]
--username value registry login username [SDOCKLE_USERNAME]
--password value registry login password. Using --password via CLI is insecure. [SDOCKLE_PASSWORD]
--insecure registry connect insecure [SDOCKLE_INSECURE]
--nonssl registry connect without ssl [SDOCKLE_NON_SSL]
--cert-path value docker daemon certificate path [SDOCKLE_CERT_PATH]
--cache-dir value cache directory
--help, -h show help
--version, -v print the version
anfe@anfe:~/Документы/8_sen/ОПЗКС$
```

Рис. 3. Установка Dockle.

На данном скриншоте показана установка утилиты Dockle, а также основные опции для данной команды.

Затем нам нужно выбрать образ, который мы будем проверять.

```
Обзор Терминал
anfek@anfek: ~/Документы/8_sen/ОПЗКС

anfek@anfek:~/Документы/8_sen/ОПЗКС$ systemctl status docker
● docker.service - Docker Application Container Engine
   Loaded: loaded (/lib/systemd/system/docker.service; enabled; vendor preset: enabled)
   Active: active (running) since Tue 2023-06-20 04:55:10 MSK; 4min 3s ago
   TriggeredBy: ● docker.socket
   Docs: https://docs.docker.com
   Main PID: 2214 (dockerd)
   Tasks: 17
   Memory: 100.1M
   CGroup: /system.slice/docker.service
           └─2214 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.sock

инн 20 04:55:10 anfek dockerd[2214]: time="2023-06-20T04:55:10.035740389+03:00" level=info msg="Starting up"
инн 20 04:55:10 anfek dockerd[2214]: time="2023-06-20T04:55:10.039565892+03:00" level=info msg="detected 172.17.0.53 nameserver, assuming system-resolved, so using resolv.conf: /run/systemd/resolve/resolv.conf"
инн 20 04:55:10 anfek dockerd[2214]: time="2023-06-20T04:55:10.145837877+03:00" level=info msg="[graphdriver] using prior storage driver: overlay2"
инн 20 04:55:10 anfek dockerd[2214]: time="2023-06-20T04:55:10.147193096+03:00" level=info msg="Loading containers: start."
инн 20 04:55:10 anfek dockerd[2214]: time="2023-06-20T04:55:10.269555906+03:00" level=info msg="Default bridge (docker0) is assigned with an IP address 172.17.0.0/16. Daemon option --bip can be used to"
инн 20 04:55:10 anfek dockerd[2214]: time="2023-06-20T04:55:10.293962392+03:00" level=info msg="Loading containers: done."
инн 20 04:55:10 anfek dockerd[2214]: time="2023-06-20T04:55:10.353029852+03:00" level=info msg="Docker daemon" commit=659604f graphdriver=overlay2 version=24.0.2
инн 20 04:55:10 anfek dockerd[2214]: time="2023-06-20T04:55:10.353266032+03:00" level=info msg="Daemon has completed initialization"
инн 20 04:55:10 anfek dockerd[2214]: time="2023-06-20T04:55:10.375738564+03:00" level=info msg="API listen on /run/docker.sock"
инн 20 04:55:10 anfek systemd[1]: Started Docker Application Container Engine.

anfek@anfek:~/Документы/8_sen/ОПЗКС$ docker search nginx_image
NAME                                DESCRIPTION                                STARS     OFFICIAL   AUTOMATED
portallr/docker/nginx_image        0
ltsal123/nginx_image               0
planforfft/nginx_image              LEMP setup.                               0         [OK]
yaroslavsvtrida/nginx_image        nginx_image                                0
1321058034/nginx_image1            0
tanouna/nginx_image                0
lgormng/nginx_image                0
sanstone/nginx_image_for_bootcamp  frontend nginx image                       0
marutid/nginx_image                0
anupgtrprasad/nginx_image          nginx это один из компонентов связки NPM = N... 0         [OK]
mk77/nginx_image                   0
hhachint/nginx_image               0
watchdogworld/nginx_image          Nginx image                               0
vstngh7ncsu/nginx_image            0
ajaycreation/nginx_image           0
vellaiyappan/nginx_image2          0
jersen/nginx_image                 0
trin/nginx_image                   0
andyhu12/nginx_image               create an image which named nginx_image    0
nikolaev275/nginx_image            0
aboogie/nginx_image                0
albernana/nginx_image              0
annetkova/nginx_image              0
xygpago/nginx_image                0
rbadr/nginx_image                  0
anfek@anfek:~/Документы/8_sen/ОПЗКС$
```

Рис. 4. Проверка запуска докера и поиск нужного образа.

```
Обзор Терминал
anfek@anfek: ~/Документы/8_sen/ОПЗКС

инн 20 04:55:10 anfek dockerd[2214]: time="2023-06-20T04:55:10.269555906+03:00" level=info msg="Default bridge (docker0) is assigned with an IP address 172.17.0.0/16. Daemon option --bip can be used to"
инн 20 04:55:10 anfek dockerd[2214]: time="2023-06-20T04:55:10.293962392+03:00" level=info msg="Loading containers: done."
инн 20 04:55:10 anfek dockerd[2214]: time="2023-06-20T04:55:10.353029852+03:00" level=info msg="Docker daemon" commit=659604f graphdriver=overlay2 version=24.0.2
инн 20 04:55:10 anfek dockerd[2214]: time="2023-06-20T04:55:10.353266032+03:00" level=info msg="Daemon has completed initialization"
инн 20 04:55:10 anfek dockerd[2214]: time="2023-06-20T04:55:10.375738564+03:00" level=info msg="API listen on /run/docker.sock"
инн 20 04:55:10 anfek systemd[1]: Started Docker Application Container Engine.

anfek@anfek:~/Документы/8_sen/ОПЗКС$ docker search nginx_image
NAME                                DESCRIPTION                                STARS     OFFICIAL   AUTOMATED
portallr/docker/nginx_image        0
ltsal123/nginx_image               0
planforfft/nginx_image              LEMP setup.                               0         [OK]
yaroslavsvtrida/nginx_image        nginx_image                                0
1321058034/nginx_image1            0
tanouna/nginx_image                0
lgormng/nginx_image                0
sanstone/nginx_image_for_bootcamp  frontend nginx image                       0
marutid/nginx_image                0
anupgtrprasad/nginx_image          nginx это один из компонентов связки NPM = N... 0         [OK]
mk77/nginx_image                   0
hhachint/nginx_image               0
watchdogworld/nginx_image          Nginx image                               0
vstngh7ncsu/nginx_image            0
ajaycreation/nginx_image           0
vellaiyappan/nginx_image2          0
jersen/nginx_image                 0
trin/nginx_image                   0
andyhu12/nginx_image               create an image which named nginx_image    0
nikolaev275/nginx_image            0
aboogie/nginx_image                0
albernana/nginx_image              0
annetkova/nginx_image              0
xygpago/nginx_image                0
rbadr/nginx_image                  0
anfek@anfek:~/Документы/8_sen/ОПЗКС$ docker pull mk77/nginx_image
Using default tag: latest
latest: Pulling from mk77/nginx_image
f669f1d21059: Pull complete
ecbec583cfc: Pull complete
ea6f1825d6d3: Pull complete
54bde7b02897: Pull complete
a3ed95cae02: Pull complete
ce9e95ae0234: Pull complete
346026b9659b: Pull complete
3a6d22d19fee: Pull complete
bb5277256172: Pull complete
e3c1428bd275: Pull complete
92bd9d928928: Pull complete
Digest: sha256:bbe658eadb494ee10304f33f6cfc7f183cd49d16338c67a6dcca25765ef9e0f2
Status: Downloaded newer image for mk77/nginx_image:latest
docker.io/mk77/nginx_image:latest
anfek@anfek:~/Документы/8_sen/ОПЗКС$ docker images
REPOSITORY    TAG       IMAGE ID       CREATED        SIZE
mk77/nginx_image latest    820c75a5c208   6 years ago   314MB
anfek@anfek:~/Документы/8_sen/ОПЗКС$
```

Рис. 5. Загружаем образ и проверяем что он появился в списке обраов.

После того как мы загрузили нужную утилиту и образ, мы можем приступать непосредственно к проверке образа на безопасность.

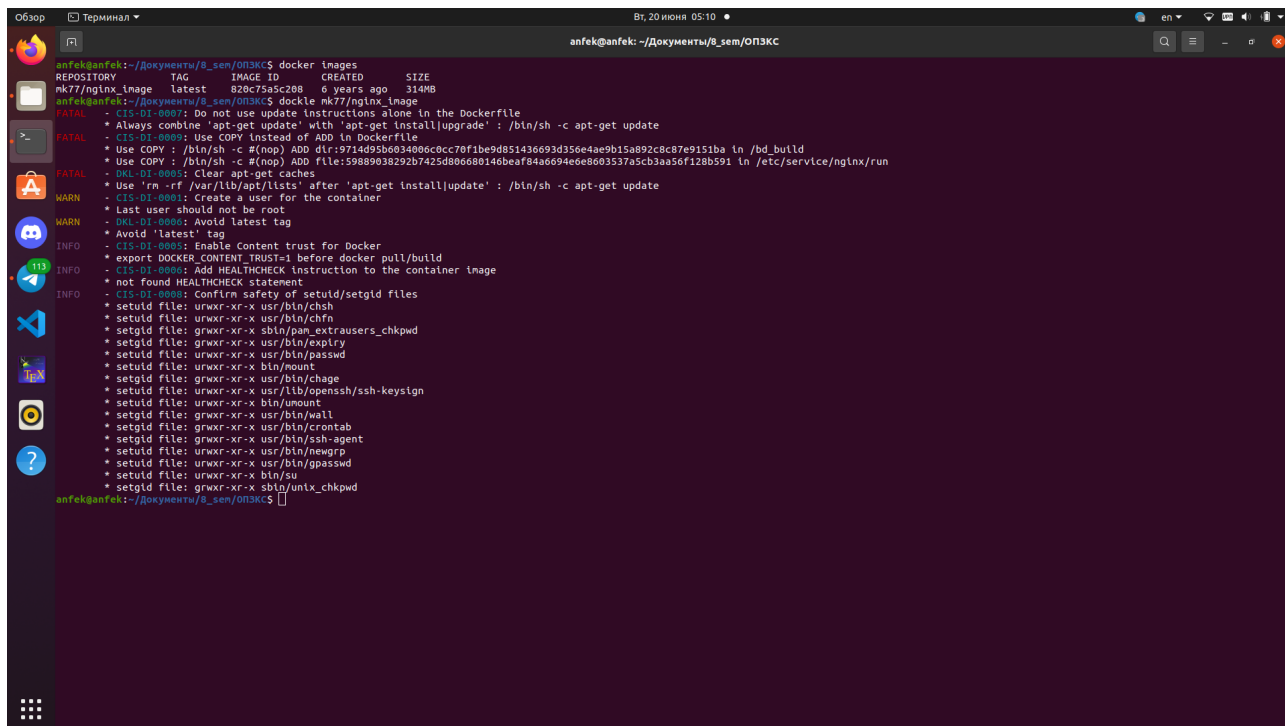


Рис. 6. Используем dockle для проверки образа.

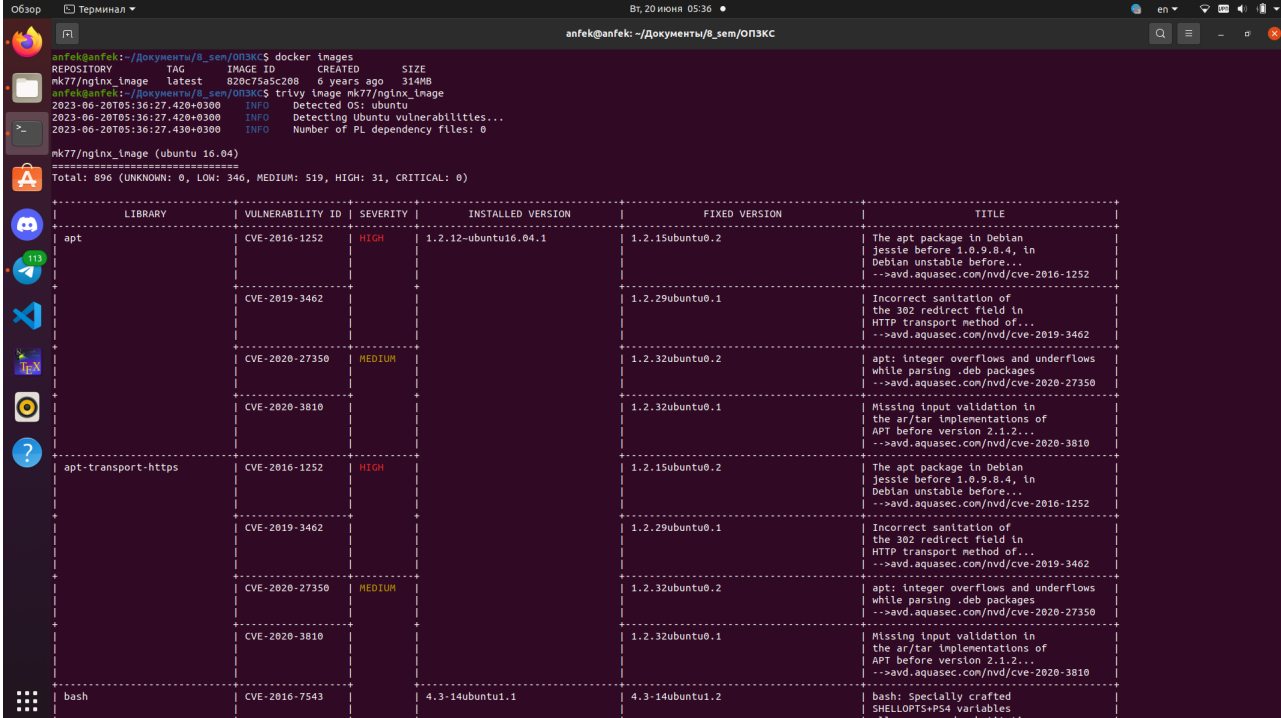
На данном скриншоте показано использование утилиты Dockle для проверки Docker-образа с помощью команды

`dockle <image_name>`

Результаты проверки отображаются с указанием найденных проблем и рекомендаций по улучшению.

### 3.3. Проверка наличия общеизвестных уязвимостей (CVE) в базовом образе и ряде зависимостей - утилитой Trivy.

Для проверки наличия общеизвестных уязвимостей (CVE) в Docker-образах и их зависимостях использовалась утилита Trivy. Ниже приведены подробности выполненных действий:



The screenshot shows a terminal window with the following content:

```
anfek@anfek: ~/Документы/8_sem/ОПЗКС
REPOSITORY TAG IMAGE ID CREATED SIZE
mk77/nginx_image latest 820c75a5c208 6 years ago 314MB
anfek@anfek:~/Документы/8_sem/ОПЗКС$ trivy image mk77/nginx_image
2023-06-20T05:36:27.420+0300 INFO Detected OS: ubuntu
2023-06-20T05:36:27.420+0300 INFO Detecting Ubuntu vulnerabilities...
2023-06-20T05:36:27.430+0300 INFO Number of PL dependency files: 0

mk77/nginx_image (ubuntu 16.04)
=====
Total: 896 (UNKNOWN: 0, LOW: 346, MEDIUM: 519, HIGH: 31, CRITICAL: 0)
```

LIBRARY	VULNERABILITY ID	SEVERITY	INSTALLED VERSION	FIXED VERSION	TITLE
apt	CVE-2016-1252	HIGH	1.2.12-ubuntu16.04.1	1.2.15ubuntu0.2	The apt package in Debian Jessie before 1.0.9.8.4, in Debian unstable before...
	CVE-2019-3462			1.2.29ubuntu0.1	Incorrect sanitization of the 302 redirect field in HTTP transport method of...
	CVE-2020-27350	MEDIUM		1.2.32ubuntu0.2	apt: integer overflows and underflows while parsing .deb packages
	CVE-2020-3810			1.2.32ubuntu0.1	Missing input validation in the ar/tar implementations of APT before version 2.1.2...
apt-transport-https	CVE-2016-1252	HIGH		1.2.15ubuntu0.2	The apt package in Debian Jessie before 1.0.9.8.4, in Debian unstable before...
	CVE-2019-3462			1.2.29ubuntu0.1	Incorrect sanitization of the 302 redirect field in HTTP transport method of...
	CVE-2020-27350	MEDIUM		1.2.32ubuntu0.2	apt: integer overflows and underflows while parsing .deb packages
	CVE-2020-3810			1.2.32ubuntu0.1	Missing input validation in the ar/tar implementations of APT before version 2.1.2...
bash	CVE-2016-7543		4.3-14ubuntu1.1	4.3-14ubuntu1.2	bash: Specially crafted SHELLOPTS+PS4 variables allows command substitution

Рис. 7. Проверяем, что образ существует в докере и запускаем сканирование образа в trivy.

На данном скриншоте показана проверка наличия Docker-образа в Docker и запуск сканирования с использованием утилиты Trivy с помощью команды `trivy <image_name>`

Полученная в результате работы утилиты таблица содержит большой объём данных, для анализа которого может потребоваться некоторое время, поэтому удобней будет вывести всю информацию в файл.

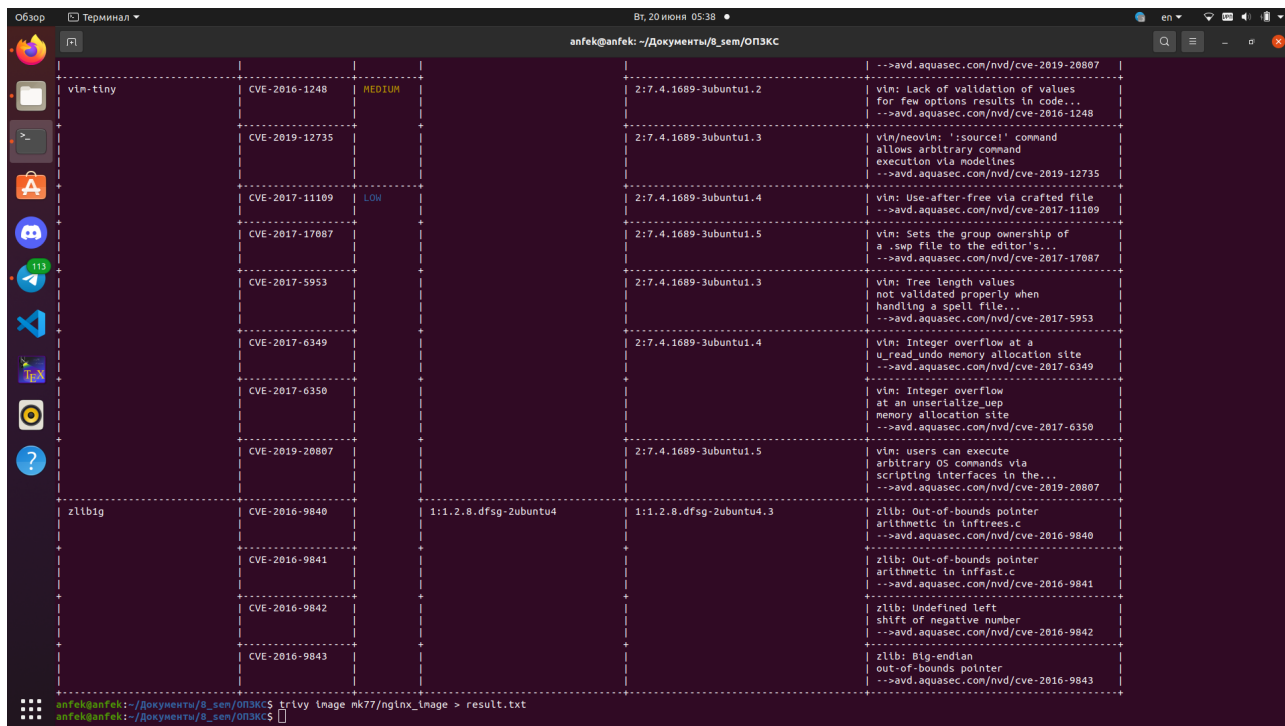


Рис. 8. Запись результата работы утилиты в файл.

Также для этого мы могли использовать альтернативную команду:

`trivy --output result.txt <image_name>`

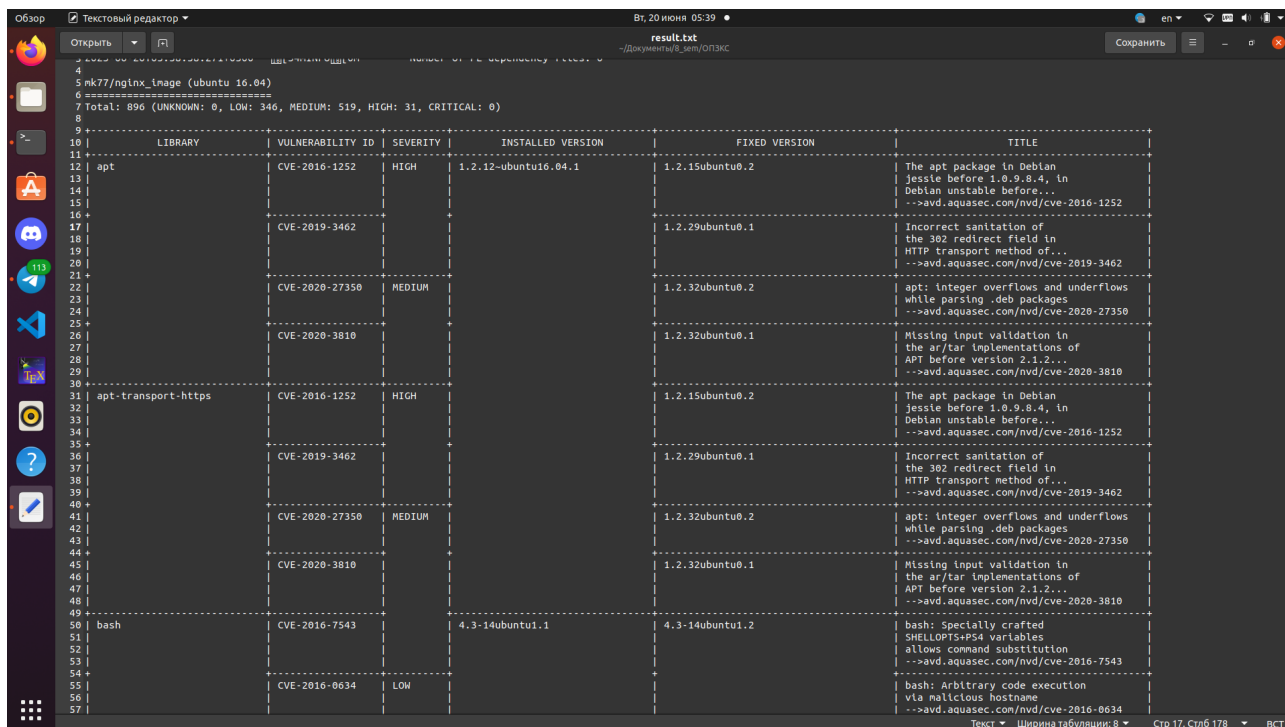


Рис. 9. Непосредственно файл.



```
Обзор Терминал
anfek@anfek: ~/Документы/8_sem/ОПЭКС

CVE-2019-20807 | | 2:7.4.1689-3ubuntu1.5 | | vin: users can execute
arbitrary OS commands via
scripting interfaces in the...
-->avd.aquasec.com/nvd/cve-2019-20807
zlib | CVE-2016-9840 | 1:1.2.8.dfsg-2ubuntu4 | 1:1.2.8.dfsg-2ubuntu4.3 | | zlibc: Out-of-bounds pointer
arithmetic in lztfrees.c
-->avd.aquasec.com/nvd/cve-2016-9840
| | | | | | zlibc: Out-of-bounds pointer
arithmetic in lztfrees.c
-->avd.aquasec.com/nvd/cve-2016-9841
| | | | | | zlibc: Undefined left
shift of negative number
-->avd.aquasec.com/nvd/cve-2016-9842
| | | | | | zlibc: Big-endian
out-of-bounds pointer
-->avd.aquasec.com/nvd/cve-2016-9843

anfek@anfek:~/Документы/8_sem/ОПЭКС$ trivy image mk77/nginx_image > result.txt
anfek@anfek:~/Документы/8_sem/ОПЭКС$ trivy image --help
NAME:
  trivy image - scan an image

USAGE:
  trivy image [command options] image_name

OPTIONS:
  --template value, -t value      output template [TRIVY_TEMPLATE]
  --format value, -f value         format (table, json, template) (default: "table") [TRIVY_FORMAT]
  --input value, -i value          input file path instead of image name [TRIVY_INPUT]
  --severity value, -s value       severities of vulnerabilities to be displayed (comma separated) (default: "UNKNOWN,LOW,MEDIUM,HIGH,CRITICAL") [TRIVY_SEVERITY]
  --output value, -o value         output file name [TRIVY_OUTPUT]
  --exit-code value               exit code when vulnerabilities were found (default: 0) [TRIVY_EXIT_CODE]
  --skip-update                   skip db update (default: false) [TRIVY_SKIP_UPDATE]
  --download-db-only              download/update vulnerability database but don't run a scan (default: false) [TRIVY_DOWNLOAD_DB_ONLY]
  --reset                         remove all caches and database (default: false) [TRIVY_RESET]
  --clear-cache, -c              clear image caches without scanning (default: false) [TRIVY_CLEAR_CACHE]
  --no-progress                  suppress progress bar (default: false) [TRIVY_NO_PROGRESS]
  --ignore-unfixed               display only fixed vulnerabilities (default: false) [TRIVY_IGNORE_UNFIXED]
  --removed-pkgs                 detect vulnerabilities of removed packages (only for Alpine) (default: false) [TRIVY_REMOVED_PKGS]
  --vuln-type value              comma-separated list of vulnerability types (os,library) (default: "os,library") [TRIVY_VULN_TYPE]
  --ignorefile value             specify .trivyignore file (default: ".trivyignore") [TRIVY_IGNOREFILE]
  --timeout value                timeout (default: 5m0s) [TRIVY_TIMEOUT]
  --light                         light mode: it's faster, but vulnerability descriptions and references are not displayed (default: false) [TRIVY_LIGHT]
  --ignore-policy value          specify the Rego file to evaluate each vulnerability [TRIVY_IGNORE_POLICY]
  --list-all-pkgs               enabling the option will output all packages regardless of vulnerability (default: false) [TRIVY_LIST_ALL_PKGS]
  --skip-files value             specify the file paths to skip traversal [TRIVY_SKIP_FILES]
  --skip-dirs value              specify the directories where the traversal is skipped [TRIVY_SKIP_DIRS]
  --cache-backend value          cache backend (e.g. redis://localhost:6379) (default: "fs") [TRIVY_CACHE_BACKEND]
  --help, -h                     show help (default: false)
```

Рис. 10. Дополнительные опции для работы с образами.

```
Обзор Терминал
anfek@anfek: ~/Документы/8_sem/ОПЭКС

USAGE:
  trivy image [command options] image_name

OPTIONS:
  --template value, -t value      output template [TRIVY_TEMPLATE]
  --format value, -f value         format (table, json, template) (default: "table") [TRIVY_FORMAT]
  --input value, -i value          input file path instead of image name [TRIVY_INPUT]
  --severity value, -s value       severities of vulnerabilities to be displayed (comma separated) (default: "UNKNOWN,LOW,MEDIUM,HIGH,CRITICAL") [TRIVY_SEVERITY]
  --output value, -o value         output file name [TRIVY_OUTPUT]
  --exit-code value               exit code when vulnerabilities were found (default: 0) [TRIVY_EXIT_CODE]
  --skip-update                   skip db update (default: false) [TRIVY_SKIP_UPDATE]
  --download-db-only              download/update vulnerability database but don't run a scan (default: false) [TRIVY_DOWNLOAD_DB_ONLY]
  --reset                         remove all caches and database (default: false) [TRIVY_RESET]
  --clear-cache, -c              clear image caches without scanning (default: false) [TRIVY_CLEAR_CACHE]
  --no-progress                  suppress progress bar (default: false) [TRIVY_NO_PROGRESS]
  --ignore-unfixed               display only fixed vulnerabilities (default: false) [TRIVY_IGNORE_UNFIXED]
  --removed-pkgs                 detect vulnerabilities of removed packages (only for Alpine) (default: false) [TRIVY_REMOVED_PKGS]
  --vuln-type value              comma-separated list of vulnerability types (os,library) (default: "os,library") [TRIVY_VULN_TYPE]
  --ignorefile value             specify .trivyignore file (default: ".trivyignore") [TRIVY_IGNOREFILE]
  --timeout value                timeout (default: 5m0s) [TRIVY_TIMEOUT]
  --light                         light mode: it's faster, but vulnerability descriptions and references are not displayed (default: false) [TRIVY_LIGHT]
  --ignore-policy value          specify the Rego file to evaluate each vulnerability [TRIVY_IGNORE_POLICY]
  --list-all-pkgs               enabling the option will output all packages regardless of vulnerability (default: false) [TRIVY_LIST_ALL_PKGS]
  --skip-files value             specify the file paths to skip traversal [TRIVY_SKIP_FILES]
  --skip-dirs value              specify the directories where the traversal is skipped [TRIVY_SKIP_DIRS]
  --cache-backend value          cache backend (e.g. redis://localhost:6379) (default: "fs") [TRIVY_CACHE_BACKEND]
  --help, -h                     show help (default: false)

anfek@anfek:~/Документы/8_sem/ОПЭКС$ trivy --help
NAME:
  trivy - A simple and comprehensive vulnerability scanner for containers

USAGE:
  trivy [global options] command [command options] target

VERSION:
  0.18.3

COMMANDS:
  image, i      scan an image
  filesystem, fs scan local filesystem
  repository, repo scan remote repository
  client, c      client mode
  server, s      server mode
  plugin, p      manage plugins
  help, h        shows a list of commands or help for one command

GLOBAL OPTIONS:
  --quiet, -q      suppress progress bar and log output (default: false) [TRIVY_QUIET]
  --debug, -d      debug mode (default: false) [TRIVY_DEBUG]
  --cache-dir value cache directory (default: "/home/anfek/.cache/trivy") [TRIVY_CACHE_DIR]
  --help, -h        show help (default: false)
  --version, -v      print the version (default: false)
```

Рис. 11. Наглядная демонстрация функционала trivy для работы с образами, файловыми системами, репозиториями и т.д.

## 4. ЗАКЛЮЧЕНИЕ

В ходе данной долгосрочной работы мы провели исследование и выполнение практических заданий, связанных с проверкой корректности и безопасности Docker-образов с использованием утилит Hadolint, Dockle и Trivy. Docker стал широко распространенным инструментом в разработке и развертывании приложений благодаря своей легковесности, портативности и масштабируемости. Однако, при работе с Docker существуют потенциальные проблемы безопасности, связанные с использованием уязвимых образов или нарушением рекомендаций по безопасности.

Проведенные задания позволили нам применить утилиту Hadolint для проверки корректности и безопасности инструкций Dockerfile. Мы смогли обнаружить потенциальные ошибки и проблемы, которые могут повлиять на безопасность и функциональность Docker-образов. Утилита Hadolint стала ценным инструментом для статического анализа Dockerfile и обеспечения соответствия стандартам и рекомендациям.

Далее мы приступили к проверке корректности и безопасности конечных и промежуточных Docker-образов с использованием утилиты Dockle. Мы осуществили установку Dockle и протестировали его функционал, который позволил нам выявить потенциальные уязвимости и нарушения безопасности в образах. Dockle стал незаменимым инструментом для проверки безопасности Docker-образов на ранней стадии разработки и внедрения.

Также мы провели проверку наличия общеизвестных уязвимостей (CVE) в базовом образе и ряде зависимостей с использованием утилиты Trivy. Мы просканировали Docker-образы и их зависимости на наличие известных уязвимостей и получили детальные отчеты о найденных проблемах. Trivy показал свою эффективность и помог нам предотвратить использование уязвимых компонентов и повысить уровень безопасности нашей инфраструктуры.

Таким образом, мы считаем, что использование утилит Hadolint, Dockle и Trivy является важным шагом в обеспечении безопасности Docker-окружений, а их применение в практике разработки и эксплуатации контейнеров является неотъемлемой частью процесса обеспечения безопасности и надежности приложений.