Docker — ????????????? ? ?????

Эта страница объясняет базовые практики безопасности при работе с Docker. Здесь разобраны команды для проверки уязвимостей образов, анализа конфигурации Docker Compose и рекомендации по безопасной эксплуатации контейнеров. Материал рассчитан на начинающих и помогает понять, **как обнаруживать потенциальные проблемы безопасности и минимизировать риски при работе с Docker-проектами**.

?????? ????????????? Docker ?????

Контейнеры упрощают развертывание приложений, но сами по себе **не гарантируют безопасность**. Основные риски могут возникать из-за:

- использования устаревших образов
- уязвимостей в зависимостях
- неправильных настроек сети
- небезопасных прав доступа
- запуска контейнеров с избыточными привилегиями

Поэтому рекомендуется регулярно:

- проверять образы на уязвимости
- анализировать конфигурацию контейнеров
- обновлять используемые образы
- минимизировать доступ контейнеров к системе

???????? ??????? ?? ??????????

Docker предоставляет инструменты для анализа безопасности образов.

????????????? ??????

```
docker scan <image_name>
```

Пример:

```
docker scan nginx:latest
```

Команда анализирует образ и ищет известные уязвимости (CVE) в:

- системных пакетах
- библиотеках
- зависимостях

Пример результата:

```
✗ High severity vulnerability found in openssl
✗ Medium severity vulnerability found in libssl
```

?????? ??????? ????? Docker Scout

Docker Scout — более современный инструмент анализа образов.

```
docker scout quickview <image_name>
```

Пример:

```
docker scout quickview node:18
```

Инструмент покажет:

- количество найденных уязвимостей
- уровень риска
- рекомендации по обновлению образа

????? ????????????? ?????????????

Рекомендуется проверять образы:

- перед использованием нового образа
- перед деплоем на production
- после обновления зависимостей

???????? ????????????? Docker Compose

Иногда проблемы безопасности возникают из-за неправильной конфигурации сервисов.

???????? ???????? ???????

```
docker compose config --services
```

Команда выводит список сервисов, описанных в `docker-compose.yml`.

Пример:

```
nginx
php
mysql
redis
```

Это помогает убедиться, что Compose видит именно те сервисы, которые ожидаются.

???????? volumes

```
docker compose config --volumes
```

Команда показывает volumes, используемые проектом.

Пример:

```
mysql_data
redis_cache
```

?????? ??? ?????

Volumes могут содержать:

- базы данных
- пользовательские файлы
- конфиденциальную информацию

Важно понимать, где именно хранятся данные.

????????? ????? ???????

```
docker compose config --networks
```

Команда показывает сети, которые использует проект.

Пример:

```
default
internal_network
```

Это помогает проверить, какие сервисы могут взаимодействовать между собой.

??????????? Docker-???????

Одной из главных причин уязвимостей является использование старых образов.

Обновить образы можно так:

```
docker compose pull
```

После загрузки новых образов нужно перезапустить сервисы:

```
docker compose up -d
```

?????????? ???? ??????????

По умолчанию контейнеры могут иметь довольно широкий доступ к системе.

Рекомендуется:

- запускать контейнеры от обычного пользователя
- избегать режима `privileged`
- ограничивать доступ к файловой системе

Пример безопасной настройки в `docker-compose.yml`:

```
services:
  app:
    user: "1000:1000"
```

Это означает, что контейнер будет работать от обычного пользователя.

?????????? ???????? ??????????

Чтобы контейнер не мог использовать все ресурсы сервера, можно задать лимиты.

Пример:

```
services:
  app:
    deploy:
```

```
    resources:
      limits:
        memory: 512M
```

Это защищает сервер от ситуаций, когда приложение начинает потреблять слишком много памяти.

?????????? ??????? ? ??????

Не стоит открывать порты контейнеров без необходимости.

Пример безопасной конфигурации:

```
ports:
  - "127.0.0.1:8080:80"
```

Это означает:

- порт доступен только локально
- извне сервера он не будет открыт

????????????? .env ??????

Часто конфиденциальные данные хранятся в `.env`.

Пример:

```
DB_PASSWORD=secret_password
API_KEY=xxxxxxxx
```

Важно:

- не добавлять `.env` в публичные репозитории
- использовать `.gitignore`

???????? ????????????????? ???????????

Контейнеры, запущенные в режиме `privileged`, имеют почти полный доступ к системе.

Проверить можно так:

```
docker inspect <container>
```

В выводе нужно обратить внимание на поле:

```
Privileged: true
```

Если оно включено — контейнер имеет расширенные права.

???????? ??????????? ???????????

Полезно периодически проверять список контейнеров:

```
docker ps
```

Это помогает обнаружить:

- неизвестные контейнеры
- забытые сервисы
- тестовые окружения

???????? ???????????? ???????

Можно посмотреть, какие образы используются:

```
docker images
```

Пример вывода:

```
nginx        latest
mysql        8
redis        7
```

Если используются очень старые версии — рекомендуется обновить их.

???????? ?????? ?????????????

????????????? latest ? production

Тег `latest` может неожиданно измениться.

Лучше указывать конкретную версию:

```
nginx:1.25
```

???????? ????? ??? ?????????????

Если порт не нужен извне — не стоит его публиковать.

?????? ?????????? ? root-????????

Лучше запускать контейнеры от обычного пользователя.

???????? ???????? ? ???????????

Пароли и ключи лучше хранить:

- в `.env`
- в системных переменных

???????? ???????? ???????? ????????????

1 ????????? ???????????

```
docker ps
```

---

2 ????????? ??????

```
docker images
```

---

3 ????????? ???????????

```
docker scout quickview nginx
```

---

4 ????????? ???????????? compose

```
docker compose config
```

---

5 ???????? ??????

```
docker compose pull
```

??????? ???????

```
docker scan <image>                    # сканирование уязвимостей
docker scout quickview <image>         # анализ безопасности образа

docker compose config --services       # список сервисов
docker compose config --volumes        # volumes проекта
docker compose config --networks       # сети проекта

docker compose pull                    # обновить образы

docker ps                              # список контейнеров
docker images                          # список образов
docker inspect <container>             # информация о контейнере
```

Revision #1
Created 2026-03-11 08:06:03 UTC by Crimson
Updated 2026-03-11 08:06:43 UTC by Crimson