Программирование: как проверить безопасность конфигурации Docker Compose до деплоя

04.11.2026 · PDF-версия статьи с сайта cms.dolinais.ru

В 2026 году контейнеризация стала стандартом, но ошибки в конфигурации Docker Compose по-прежнему приводят к инцидентам. Один из самых распространённых сценариев: в файле docker-compose.yml сервис публикует порт базы данных на все интерфейсы, хотя должен быть доступен только внутри сети. В результате UFW может показывать, что входящие соединения запрещены, но контейнер обходит правила файрвола и PostgreSQL оказывается открытым для всего интернета. В этой статье мы разберём, как предотвратить такие ситуации с помощью двух небольших скриптов на Python: первый проверяет compose-файлы до деплоя, второй — анализирует, что на самом деле слушает сервер.

Почему стандартные средства защиты не всегда работают

Многие администраторы полагаются на **UFW** (Uncomplicated Firewall) — удобную надстройку над iptables. Однако контейнеры Docker управляют правилами iptables самостоятельно и часто добавляют цепочки, которые обходят правила UFW. В результате порт, опубликованный в compose-файле как "5432:5432", становится доступен извне, даже если UFW настроен на блокировку всех входящих соединений, кроме 22, 80 и 443.

Проблема усугубляется тем, что разработчики часто копируют примеры конфигураций из документации, не задумываясь о безопасности. Поэтому важно проверять compose-файлы ещё на этапе сборки или в процессе CI/CD.

Скрипт 1: проверка compose-файлов перед деплоем

Первый скрипт читает YAML-файл и ищет сервисы, у которых в секции ports указан порт, привязанный ко всем интерфейсам (например, "5432:5432"). Если такой порт найден, скрипт завершается с ненулевым кодом возврата, что останавливает пайплайн.

```
import sys
import yaml

def check_compose(file_path):
    with open(file_path, 'r') as f:
        compose = yaml.safe_load(f)
    errors = []
    for service_name, service in compose.get('services', {}).items():
        ports = service.get('ports', [])
        for port in ports:
            # ???? ????? ???? ??????? ??? ????????
            if isinstance(port, str):
                # ?????? "host:container" ??? "host:container/proto"
                if ':' in port:
                    host_part = port.split(':')[0]
                    if host_part and host_part != '127.0.0.1':
                        errors.append(f"?????? '{service_name}': ???? {port} ?
```

```
?????????? ?? ??? ??????????")
            elif isinstance(port, dict):
                # ??????? ?????????
                published = port.get('published')
                host_ip = port.get('host_ip', '0.0.0.0')
                if published and host_ip not in ('127.0.0.1', '::1'):
                    errors.append(f"?????? '{service_name}': ???? {published}
??????????? ?? {host_ip}")
    return errors

if __name__ == '__main__':
    file_path = sys.argv[1] if len(sys.argv) > 1 else 'docker-compose.yml'
    errors = check_compose(file_path)
    if errors:
        print("?????????? ??????? ?????????? ??????:")
        for err in errors:
            print(f"  - {err}")
        sys.exit(1)
    print("???????? ????????: ??????? ?????????? ?????? ?? ???????.")
    sys.exit(0)
```

Скрипт проверяет оба формата указания портов: короткий (строка) и длинный (словарь). Он запускается на Python 3.13 с библиотекой PyYAML 6.0.3. Если в файле есть хотя бы одна опасная публикация, скрипт выводит сообщение и завершается с кодом 1, что сигнализирует об ошибке в CI.

Скрипт 2: аудит слушающих портов без ss и netstat

Второй скрипт полезен для проверки уже работающего сервера. Он читает файлы из /proc/net/tcp и /proc/net/tcp6, которые содержат информацию о текущих TCP-соединениях, и определяет, какие порты находятся в состоянии LISTEN (ожидание входящих подключений). Это позволяет обойтись без внешних утилит, которые могут быть не установлены или требовать прав root.

```
import glob import struct import socket def hex_to_ip(hex_str): #
 ????????????? ???????????????? ???????????? IP-?????? ? ???????? ??? #
 ??? IPv4: 8 ????????, ??? IPv6: 32 ??????? if len(hex_str) == 8: addr =
 int(hex_str, 16) return socket.inet_ntoa(struct.pack('
```

????????: https://cms.dolinais.ru/programmirovanie-kak-proverit-bezopasnost-konfiguracii-docker-compose-do-deploya