

СРАВНИТЕЛЬНЫЙ АНАЛИЗ ИНСТРУМЕНТОВ СКАНИРОВАНИЯ DOCKER-ОБРАЗОВ НА НАЛИЧИЕ УЯЗВИМОСТЕЙ

Я.И. Савелиев¹, Д.А. Алтунин¹, И.С. Поздняк¹

*¹Поволжский государственный университет телекоммуникаций
и информатики, г. Самара
e-mail: i.pozdnyak@psuti.ru*

Аннотация

Рассматривается задача оценки согласованности результатов сканирования контейнерных образов на наличие уязвимостей. Описаны методы обнаружения уязвимостей, реализованные в трёх инструментах, и предложена методика количественного сравнения их результатов на основе коэффициента Жаккара. Приведены результаты эксперимента на выборке из официальных образов публичного реестра.

Ключевые слова: контейнеризация, уязвимости, сканеры, DevSecOps, коэффициент Жаккара, CI/CD, Docker.

Введение

Контейнеризация на основе Docker стала стандартным подходом к развёртыванию приложений в современных информационных системах. Каждый контейнерный образ представляет собой многослойную файловую систему, включающую операционную систему, системные библиотеки и прикладные компоненты. Любой из этих элементов может содержать известные уязвимости, что делает контейнерные образы потенциально крупной поверхностью атаки [1].

Для обнаружения уязвимостей до развёртывания применяются специализированные инструменты сканирования. Однако разные инструменты опираются на разные базы данных уязвимостей и используют разные алгоритмы анализа, что приводит к существенным расхождениям в результатах. Это создаёт неопределённость при выборе средства защиты и ставит вопрос о полноте обнаружения при использовании одного сканера.

В работе [2] было проведено сравнение двух инструментов на большой выборке образов, однако третий популярный сканер не рассматривался, а использованные версии инструментов уже устарели. Целью настоящей работы является экспериментальное сравнение трёх наиболее распространённых инструментов сканирования на актуальных версиях и выработка практических рекомендаций по их совместному применению.

1. Методы обнаружения уязвимостей в контейнерных образах

Современные инструменты применяют три основных подхода к обнаружению уязвимостей в контейнерных образах.

Первый подход – сопоставление установленных компонентов с базами известных уязвимостей. Сканер извлекает из образа перечень системных и прикладных пакетов,

определяет их версии и проверяет наличие зарегистрированных уязвимостей в одной или нескольких публичных базах данных. Каждый инструмент использует свой набор источников: один агрегирует более десяти баз, включая отраслевые, другой формирует собственную базу на основе двух основных публичных источников, третий работает с закрытой проприетарной базой [3–5]. Именно различие в источниках данных является главной причиной расхождения результатов.

Второй подход – формирование перечня программных компонентов образа в машиночитаемом формате. Такой перечень позволяет повторно проверить тот же набор компонентов на обновлённой базе уязвимостей без повторного обращения к самому образу, что повышает воспроизводимость анализа.

Третий подход – анализ конфигурации и рекомендации по выбору базового образа. Один из инструментов оценивает выбор базового образа и предлагает менее уязвимую альтернативу. Другой проверяет конфигурационные файлы на типичные ошибки: запуск процессов с избыточными привилегиями, отсутствие проверок работоспособности.

2. Методика эксперимента

Для проведения эксперимента были выбраны три наиболее распространённых инструмента сканирования контейнерных образов: Trivy (разработчик – Aqua Security), Gype (разработчик – Anchore) и Docker Scout (разработчик – Docker, Inc.). Все три инструмента установлены на единый стенд под управлением операционной системы Ubuntu.

Выборка составила 20 официальных образов из публичного реестра Docker Hub, охватывающих основные категории: веб-серверы, среды исполнения языков программирования, системы управления базами данных, промежуточное программное обеспечение и базовые операционные системы. Каждый образ был последовательно просканирован всеми тремя инструментами в течение одного дня, что исключило влияние обновлений баз уязвимостей на результаты.

Из результатов сканирования для каждого образа извлечены множества уникальных идентификаторов уязвимостей. Степень согласованности результатов оценивалась с помощью коэффициента Жаккара – отношения числа уязвимостей, обнаруженных обоими инструментами, к общему числу уязвимостей, обнаруженных хотя бы одним из них. Значение коэффициента варьируется от 0 (ни одного совпадения) до 1 (полное совпадение).

3. Результаты

Результаты сканирования всех 20 образов представлены в таблице 1.

Таблица 1 – Результаты сканирования 20 образов

Образ	Trivy	Gype	Scout	J(T,G)	J(T,S)
node	1050	294	171	0,276	0,149
python	495	261	158	0,521	0,293
ruby	477	243	146	0,506	0,279
wordpress	437	204	126	0,463	0,268
golang	359	125	91	0,348	0,243
php	334	100	87	0,299	0,249
nginx	78	77	54	0,914	0,571
postgres	72	70	53	0,919	0,667
httpd	59	57	38	0,902	0,565

mongo	39	47	39	0,792	1,000
redis	32	29	22	0,848	0,588
debian	32	28	22	0,875	0,588
memcached	32	28	22	0,875	0,588
mariadb	28	52	28	0,538	1,000
mysql	20	28	20	0,500	1,000
tomcat	13	29	13	0,448	1,000
temurin	13	29	13	0,448	1,000
ubuntu	9	13	9	0,692	1,000
rabbitmq	9	48	41	0,188	0,220
alpine	0	2	2	0,000	0,000
Итого	3588	1764	1155	0,568*	0,563*

$J(T,G)$ — коэффициент Жаккара для пары Trivy–Grype; $J(T,S)$ — коэффициент Жаккара для пары Trivy–Scout; * — среднее значение по 20 образам

Суммарно Trivy обнаружил наибольшее число уязвимостей и лидировал в 12 из 20 образов, Grype – в 8. По уязвимостям высокой и критической степени опасности соотношение аналогичное: 481, 263 и 165 соответственно.

Средний коэффициент Жаккара для пары Trivy–Grype составил 0,568, для Trivy–Scout – 0,563. Это означает, что около 43% уязвимостей, найденных одним инструментом, отсутствуют в результатах другого. Разброс значителен: от практически полного совпадения на образах с системными пакетами до полного расхождения на минимальном образе alpine, где Trivy не обнаружил ни одной уязвимости.

Основной источник расхождений – языковые зависимости. На образах с большим числом таких зависимостей Trivy находит значительно больше уязвимостей. На образах без них Grype показывает преимущество за счёт лучшей работы с отраслевыми базами уязвимостей.

4. Выводы

Полученные результаты противоречат выводам работы [2], где Grype опережал Trivy в подавляющем большинстве случаев. Причина – существенное улучшение покрытия языковых зависимостей в новых версиях Trivy. На образах без таких зависимостей расклад сил между инструментами не изменился.

Использование одного сканера оставляет около 43% уязвимостей необнаруженными. Для CI/CD рекомендуется Trivy как основной инструмент благодаря наибольшему покрытию. Grype целесообразен как дополнение для образов без языковых зависимостей, где он показывает преимущество. Docker Scout полезен как источник рекомендаций по выбору базового образа, но не как самостоятельный сканер.

Заключение

В работе проведено экспериментальное сравнение трёх инструментов сканирования контейнерных образов на наличие уязвимостей. Показано, что ни один из инструментов не обеспечивает полного покрытия: значительная часть уязвимостей обнаруживается только одним из сканеров. Выявлена зависимость степени расхождения от типа образа. Результаты работы могут быть использованы при проектировании политик безопасности контейнерных сред.

Список использованных источников

1. Palo Alto Networks. State of Cloud-Native Security Report. – 2024. – URL: <https://www.paloaltonetworks.com/state-of-cloud-native-security> (дата обращения: 01.03.2026).
2. Deciphering Discrepancies: A Comparative Analysis of Docker Image Security / B. Boles, E. O'Donoghue, R. M. Muneza [et al.] // 24th IEEE International Conference on Source Code Analysis and Manipulation (SCAM). – Flagstaff, 2024.
3. Aqua Security. Trivy: Comprehensive Security Scanner. – URL: <https://github.com/aquasecurity/trivy> (дата обращения: 01.03.2026).
4. Anchore, Inc. Gype: A Vulnerability Scanner for Container Images. – URL: <https://github.com/anchore/gype> (дата обращения: 01.03.2026).
5. Docker, Inc. Docker Scout documentation. – URL: <https://docs.docker.com/scout/> (дата обращения: 01.03.2026).

COMPARATIVE ANALYSIS OF DOCKER IMAGE VULNERABILITY SCANNING TOOLS

Ya.I. Saveliev¹, D.A. Altunin¹, I.S. Pozdnyak¹

¹*Povolzhsky State University of Telecommunications and Informatics, Samara
e-mail: i.pozdnyak@psuti.ru*

Abstract. Three container image vulnerability scanners were experimentally compared on 20 official Docker Hub images. The tools showed significant disagreement: the average Jaccard coefficient was 0.568, meaning approximately 43% of vulnerabilities found by one tool were absent from the other. The main source of discrepancy is language-specific dependencies. A combination of two scanners is recommended for CI/CD pipelines to maximize detection coverage.

Keywords: containerization, vulnerabilities, image scanning, Jaccard coefficient, continuous integration.