DOI: https://doi.org/10.23939/csn2024.02.034
УДК 004.75; 004.8

МЕТОДИ СКАНУВАННЯ ОБРАЗУ DOCKER КОНТЕЙНЕРІВ НА ПРЕДМЕТ ВРАЗЛИВОСТЕЙ БЕЗПЕКИ

Д.Г. Дарієнко, Н.М. Когут

Національний університет "Львівська політехніка",
кафедра захисту інформації
E-mail: dmytro.h.dariienko@lpnu.ua, nazarii.m.kohut@lpnu.ua

© Дарієнко Д.Г., Когут Н.М., 2024

З розвитком контейнеризованих середовищ, питання безпеки стає критично важливим для розгортання додатків. У цій статті проведено порівняльний аналіз статичного та динамічного методів сканування образів Docker контейнерів. Статичний аналіз використовується для виявлення потенційних вразливостей до розгортання контейнера, тоді як динамічний аналіз проводиться в ізольованому середовищі під час виконання, забезпечуючи надійність продукту. Порівняно роботу сканерів Trivy, JFrog Xray, Snyk і Docker Scout, підкреслено їх переваги, недоліки та ефективність у різних умовах. Доведено, що Trivy знаходить найбільше вразливостей серед протестованих сканерів. Snyk та Xray видають приблизно однакові результати, проте Xray також перевіряє шифрування важливих даних, таких як паролі. Docker Scout виявився найслабшим представником, єдиною перевагою якого є відкритий доступ результатів, які можна проаналізувати без завантаження образу на сервер чи персональний комп'ютер розробника. Особливу увагу приділено статичному аналізу через його ширше покриття вразливостей, включаючи операційні пакети та залежності додатків. Продемонстровано різницю в оцінці критичності вразливостей різними сканерами, а також обговорено, як велика кількість знайдених вразливостей не завжди означає високий рівень ризику. На основі проведеного аналізу запропоновано критерії для вибору сканера, щоб уникнути витоку інформації через непомічені вразливості.

Ключові слова: захист інформації, кібербезпека, контейнер, docker, сканування, вразливість безпеки.

1. Вступ

За останні роки контейнерні архітектури набрали широкого поширення. Linux-контейнер - це метод віртуалізації на рівні операційної системи, що дозволяє запускати та виконувати програми та сервіси в ізольованих середовищах на Linux. Вони використовують ядро операційної системи для віртуалізації ресурсів, таких як пам'ять, процесор та мережа. Контейнери дозволяють збирати програмне забезпечення та його залежності в один виконуваний пакет, що полегшує розгортання та управління програмами [1]. Проте використання цієї технології викликає вразливості, які не виникають при роботі з віртуальними машинами. Для визначення наявності цих вразливостей використовуються різноманітні сканери, проте вони мають свої сліпі зони. Кожна з цих сліпих зон може призвести до серйозних втрат. Зловмисники можуть скористатися цією сліпою для нас зоною, щоб отримати доступ до контейнера або навіть до системи, на якій запущений контейнер. Це може викликати витік важливих даних, запуск команд чи навіть повний контроль над інфраструктурою.

2. Огляд літературних джерел

Вчені активно вивчають різні аспекти цієї технології, зокрема, її вразливості та методи захисту [2-6]. Завдяки цим дослідженням ми отримуємо нові рішення для забезпечення безпеки програмних продуктів, що базуються на Docker, що є критично важливим у сучасному цифровому світі.

Для формування підходу з використанням паралельно кількох сканерів необхідно визначити параметри, які впливають на цей процес. А саме: визначення типу сканування, зручність використання інструменту для розробника, можливості з виявлення вразливостей. Розглянути вплив кожного з цих параметрів на процес сканування і побудувати модель порівняння результатів. Для того щоби це зробити слід зрозуміти будову образу Docker контейнера.

Принцип побудови контейнера

Контейнерний образ Docker [7] складається з різних шарів, що включають файли системи, бібліотеки, залежності та інші компоненти, необхідні для роботи додатку. Загалом, образ можна розглядати як уявне представлення файлової системи, яка є статичною та залишається незмінною при запуску контейнера (з винятком змін, що можуть бути внесені під час виконання)[8]. Приклад будови зображений на рисунку 1.

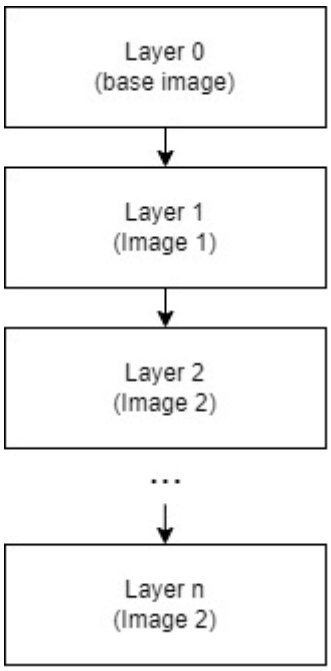

Рисунок 1. Будова Docker контейнера

Базовий образ (Base Image): Базовий образ є основою для побудови контейнера. Він містить базову операційну систему (наприклад, Ubuntu, Alpine Linux тощо) та мінімальний набір утиліт та пакунків. Базовий образ мусить бути простим настільки це можливо, адже додаткові файли можуть збільшити шанс наявності вразливостей [9].

Шари (Layers): Образ контейнера складається з набору шарів, кожен з яких представляє собою зміни, які вносяться у файлову систему під час створення образу. Кожен шар може містити файли, конфігураційні параметри, бібліотеки та інші складові. Docker використовує концепцію кешування шарів, що дозволяє ефективно зберігати та обмінюватися частинами образів між різними контейнерами.

Метадані (Metadata): Крім файлової системи, образ також містить метадані, які описують його. Це може включати інформацію про автора, версію, команди для запуску контейнера, залежності та іншу конфігурацію.

Така структура образу впливає на процес його сканування, адже вразливість може існувати в базовому образі і бути присутньою у всіх подальших нашаруваннях, а може бути виправлена в одному з подальших прошарків.

Концепція сканування

Існує кілька методів сканування Docker-образів на предмет вразливостей. Основні з них використовують різні технології та алгоритми, але мета у всіх одна – знайти потенційні проблеми безпеки та вразливості в інфраструктурі контейнерів.

Статичний аналіз образів – використовує аналіз структури образу, включаючи файли, бібліотеки, залежності і т.д., для виявлення вразливостей. Інструменти, які працюють на цьому принципі, перевіряють відомі бази даних вразливостей та порівнюють компоненти образу з цими даними.

Динамічний аналіз образів – полягає в тестуванні образів в режимі виконання, а не просто їх статичного аналізу. Інструменти, що використовують динамічний аналіз, можуть симулювати виконання образу та виявляти потенційні вразливості шляхом перехоплення взаємодії з системою та додатками в образі.

Технічна основа цих інструментів часто включає в себе використання алгоритмів аналізу вразливостей, які порівнюють складові образу з відомими базами даних вразливостей. Вони також можуть використовувати техніки динамічного аналізу, щоб перевірити образи в реальному часі. Багато з цих інструментів використовують API Docker та інші інтерфейси для взаємодії з образами та контейнерами.

Розглянемо кожний тип сканування детальніше.

Статичний аналіз

Статичний аналіз образів Docker є процесом аналізу внутрішньої структури образу без його запуску. Основна мета – виявлення потенційних вразливостей та проблем безпеки на рівні файлів, бібліотек, залежностей та конфігураційних параметрів.

Прикладом програмного забезпечення для такого сканування може бути Trivy – це безкоштовний інструмент для сканування контейнерів на предмет вразливостей.

Він здатний аналізувати Docker-зображення і виявляти вразливості, які можуть бути використані зловмисниками. Trivy підтримує кілька різних джерел вразливостей, таких як OS пакети, бази даних CVE і файлові системи.

Сканування охоплює такі етапи:

1. ***Завантаження Бази даних вразливостей.*** Містить Базу даних вразливостей. Використовує власну базу даних вразливостей, яка є комбінацією з декількох ресурсів таких як Ubuntu CVE Tracker, Amazon Linux Security Center [10-13].
2. ***Завантаження всіх шарів образу.*** Так як обрах побудований на основі шарів, кожен з них треба сканувати окремо для більш точного результату.
3. ***Аналіз.***
4. ***Запис знайдених вразливостей.***

Статично образи тестують кожного разу, як збирається новий. Також сканери підтримують виставлення порогу чутливості, що дозволяє ігнорувати некритичні вразливості. Такий аналіз не потребує багато фінансових витрат, адже не потребує окремого середовища для роботи. Процес сканування може займати близько 100 мегабайт оперативної пам’яті.

Динамічний аналіз

Динамічний аналіз образів Docker контейнерів – це процес аналізу контейнера, який включає його запуск та спостереження за його поведінкою в реальному часі в ізольованому середовищі. Основна мета цього аналізу полягає у виявленні потенційно шкідливих дій, вразливостей або ненормальної поведінки, яка може вказувати на компрометацію безпеки або інші проблеми [14].

Основні етапи динамічного аналізу образів Docker контейнерів включають такі дії:

1. ***Запуск контейнера:*** Образ Docker контейнера запускається в ізольованому середовищі, зазвичай на спеціальній платформі для контейнерів, такій як Docker Engine або Kubernetes.
2. ***Спостереження за діями контейнера:*** Під час роботи контейнера відбувається моніторинг його дій та взаємодій з іншими системами, середовищем та мережею. Це може включати моніторинг мережевого трафіку, системних викликів, взаємодії з файловою системою тощо.
3. ***Виявлення аномалій:*** Аналізатор динамічного аналізу виявляє будь-які аномальні дії або вразливості, такі як намагання доступу до конфіденційної інформації, несподівані мережеві з'єднання або використання вразливих функцій програм.
4. ***Застосування політик безпеки:*** На основі виявлених аномалій або порушень політик безпеки можуть бути застосовані відповідні заходи, такі як блокування доступу до ресурсів, автоматичне видалення контейнера або відправлення сповіщень про інцидент безпеки.
5. ***Звітність:*** Детальний звіт про результати динамічного аналізу може бути згенерований для подальшого аналізу та використання. Цей звіт може містити інформацію про виявлені вразливості, аномальну поведінку та рекомендації з проблем безпеки.

Динамічний аналіз дозволяє отримати більш повний огляд безпеки контейнерів, оскільки він враховує їхню реальну поведінку в робочому середовищі. Це допомагає виявляти складні вразливості та атаки, які можуть бути непомітними під час статичного аналізу.

Проте на відміну від статичного аналізу динамічний потребує значно більше ресурсів. Таке тестування потребує виділення коштів на окреме середовище для запуску контейнера, розробник мусить налаштувати сканер власноруч, або цим займається окремий спеціаліст. Таким чином, динамічний аналіз рідше використовується в реальних проєктах, переважно на великих, де компанія має фінансові можливості найняти pen-тестерів (з англ. penetration - проникнення).

Отже, мета цього дослідження полягає в оцінці та порівнянні рівня безпеки сканерів для образів Docker контейнерів, з метою виявлення їх потенційних вразливостей. Сканери Docker контейнерів використовуються для автоматизованої перевірки безпеки контейнерів на наявність вразливостей і відповідності стандартам безпеки [15-17]. Це дослідження спрямоване на ідентифікацію сильних та слабких сторін сканерів, їх здатності до виявлення вразливостей, швидкості виконання та точності результатів. Відповідний аналіз забезпечить корисну інформацію для розробників, що працюють з Docker контейнерами, щоб вони могли зробити обґрунтований вибір щодо використання сканерів з точки зору безпеки їхніх додатків та інфраструктури.

3. Постановка Задачі

Провести сканування образів від офіційних розробників, від не офіційних розробників, що є доступними у відкритому репозиторії Docker Hub. А також провести дослідити результати сканування офіційного образу з внесеними змінами у вигляді виконавчого файлу, з необмеженими правами доступу.

Провести аналіз образу побудованого на основі Ubuntu: 24.04 з додаванням одного файлу, з правами доступу 777 (можливість читати, писати та виконувати файл для всіх користувачів).

Провести сканування образів, що є у відкритому доступі на Docker Hub за пошуком ubuntu. Єдиним критерієм за якими відбулась фільтрація – розробник образу має бути не офіційним з метою збільшити кількість вразливостей, а також порівняти звіти від різних сканерів.

Для сканування обрано 4 представника: Snyk, Jforg Xray, Aqua Trivy та Docker Scout.

Їх обрано, через те, що вони можуть репрезентувати типові сценарії використання сканеру:

- Вбудований у приватний репозиторій для зберігання образів - Xray [18-19]
- Безкоштовний сканер, що поширюється за ліцензією Apache – Trivy [20]
- Багатофункціональний сканер, з інтеграцією в різноманітні середовища розробки - Snyk [21]
- Вбудований у відкритий репозиторій Docker Scout. [22-23]

Варто зазначити, що Docker Scout та Xray мають схожу роль, тому варто визначити чи є потреба у використанні кожного з них.

4. Порівняння статичного та динамічного методів сканування

Критично важливо обирати правильний засіб тестування. На таблиці 1 зображено приклади вразливостей які можна знайти за допомогою кожного з типів сканування.

Таблиця 1

Порівняння можливостей різних типів сканування

Статичний аналіз			Динамічний аналіз
Пакети ОС	Залежності	Додаткові файли	
apk, rpm, dpkg, yum	gem, pip, poetry, composer, yarn, cargo, java, python, nodejs, js, ruby, php	shell script, CMD parameters, file contents	malware analysis, strace, tcpdump

Як бачимо, покриття статичного аналізу значно більше ніж динамічного. Проте важливо зауважити, що вразливості, які можуть бути виявлені динамічним методом не менш критичні. Також існують відмінності в покритті вразливостей.

Розглянемо компоненти кожного типу детальніше. Пакети операційної системи залежать від типу операційної системи. Вони часто знаходяться у вільному доступі і не є заблокованими для завантаження за замовчуванням. Деякі образи, таких як alpine роблять максимально легкими, щоб уникнути потенційно вразливих застосунків і зменшити розмір.

Залежності – це додаткові бібліотеки, встановлені додатково і потрібні для виконання конкретного програмного коду. Наприклад для застосунку на python потрібен модуль requests, щоб виконувати API запити.

Додаткові файли містять конфігурацію оточення, іноді в них можуть помістити не шифровані секретні дані. Що є великою помилкою, адже через такі контейнери зловмисник може отримати доступ до всіх інших ресурсів.

Динамічний аналіз виконується за рахунок аналізу трафіку, системних викликів і пошуку вірусних програм, які не можливо помітити під час статичного аналізу.

Слід також врахувати, що не публічному репозиторії Docker Hub офіційні образи проходять обов’язкове сканування.

5. Результати дослідження

Перша частина експерименту показала, що статичне сканування виявило лише вразливості пов’язані з бібліотеками програм встановлених в контейнері. Жоден зі сканерів не виявив файл, до якого не правильно налаштовані права доступу. Також жоден зі сканерів не помітив, що контейнер запускається під root користувачем. Отже, розробник сам має контролювати файли які створює, на додає в образ при збірці.

Велика кількість результатів однакова для усіх сканерів, що брали участь у досліджені. Це спричинене тим, що вони посилаються на одні і ті ж бази вразливостей. Проте є відмінності пов’язані з якістю і кількістю виявлених загроз. Серед них Jfrog Xray показує себе найкраще. За допомогою нього були виявлені вразливості пов’язані з наявністю не зашифрованих даних та налаштуванням NGINX. Вразливості пов’язані з незашифрованими даними часто можуть бути помилково позитивними. В контейнерах GoCD є присутній зразок для пароля, що викликає помилкове спрацювання. Відповідно такі результати потребують контролю вручну. Також Xray здатен виявляти наявність незашифрованих ssh ключів, паролів та інших даних, що мають бути відсутні.

Варто зазначити, що більшість вразливостей, пов'язаних з наявністю незашифрованих секретів, виявлена у образах Docker від неофіційних розробників. Це часто зумовлено тим, що неофіційні розробники можуть недооцінювати важливість безпеки або не мати достатнього досвіду у захисті

конфіденційних даних. Незашифровані секрети, такі як паролі, ключі API або конфіденційні налаштування, можуть бути залишені в образах Docker, що створює значний ризик для безпеки. Використання образів від неофіційних джерел вимагає додаткової обережності та перевірки, щоб уникнути потенційних загроз та забезпечити надійний захист додатків і даних. У багатьох компаніях заборонено використання таких образів Docker взагалі. Це рішення ґрунтується на необхідності забезпечення високого рівня безпеки та надійності IT-інфраструктури.

Також виявлено, що кількість знайдених вразливостей може різко відрізнятись серед різних сканерів.

Це продемонстровано на рисунку 2. На діаграмі можна чітко бачити, як Trivy випереджає всіх у кількості знайдених вразливостей

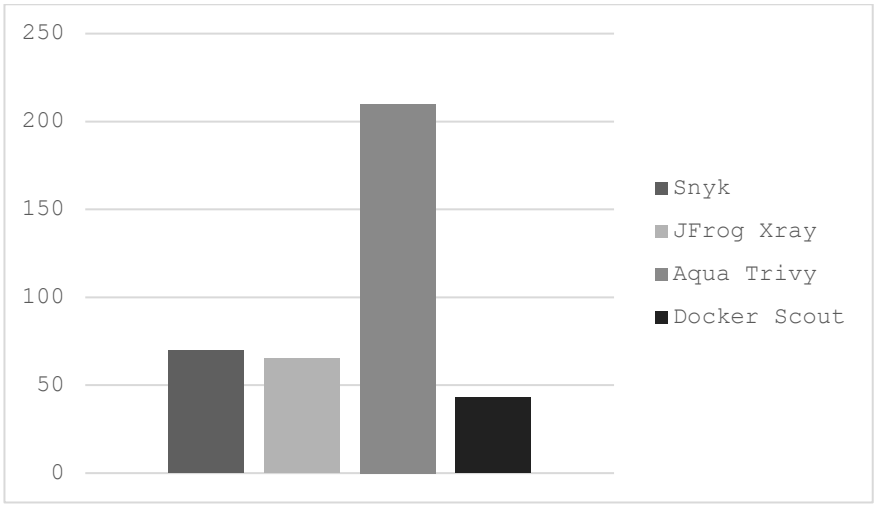

Рисунок 2. Середня кількість знайдених вразливостей кожним зі сканерів

Проте цю статистику не можна вважати вичерпною, так як одні і ті самі вразливості можуть бути віднесені до різних категорій. Тому треба також розглянути середні значення серед знайдених вразливостей кожним сканером. Проте вибір сканера базуючись лише на кількості виявлених вразливостей є хибним рішенням, оскільки цей підхід не враховує якість виявлених вразливостей, їх критичність та контекст, у якому вони виникають. Кількість вразливостей сама по собі не дає повної картини про безпеку системи; важливо також враховувати, наскільки легко ці вразливості можуть бути використані зловмисниками і які наслідки це може мати для організації. Крім того, різні сканери можуть мати різні алгоритми та бази даних, що призводить до різного рівня виявлення тих самих проблем.

Також чітко видно, сканери самостійно відносять вразливість до відповідної категорії базуючись на власних алгоритмах. Інструменти можуть мати різні підходи до оцінки ризиків та пріоритизації вразливостей. Один інструмент може вважати певну вразливість критичною, тоді як інший оцінить її як менш значущу, виходячи з контексту та можливостей експлуатації. Це продемонстровано на рисунку 3.

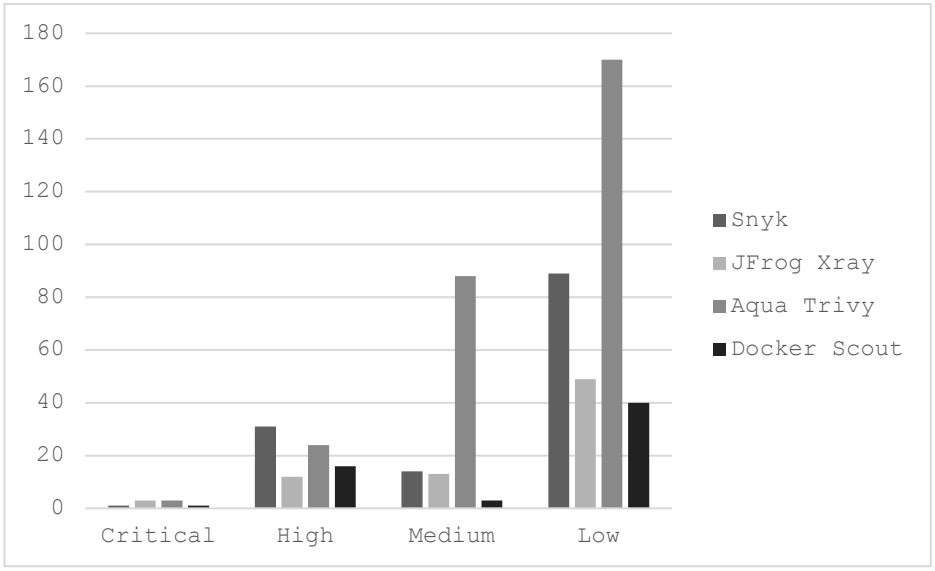

Рисунок 3. Середні значення кількості знайдених вразливостей кожного типу

6. Аналіз результатів

Така різниця спричинена тим, що різні інструменти мають унікальні алгоритми та підходи до виявлення вразливостей. І використання лише одного не дає повної картини бачення. Для аналізу результатів найкраще підходить вивід звіту у форматі JSON.

Такий вивід результатів сканування дозволяє значно спростити автоматизацію аналізу та обробки даних. JSON є структурованим та зручним для читання форматом, який підтримується багатьма мовами програмування і бібліотеками. Це дає можливість легко інтегрувати результати сканування у різні інструменти та системи для подальшого аналізу, моніторингу та звітності. Крім того, завдяки структурованому формату JSON, дані можуть бути легко перетворені або фільтровані відповідно до конкретних потреб, що робить процес управління вразливостями більш ефективним та гнучким. Також за допомогою Json часто можна отримати більше фінформації, так як стандартнай вигляд у вигляді таблиці включає лише категорію вразливості (Critical, High, Medium, Low, Undefined), пакет, в якому ця проблема виявлена, посилання на базу даних, з якої цю вразливість знайшов сканер. Проте ми не можемо знати конкретного шару, на якому з’явилась ця вразливість та шляху до файлу. Ця інформація може допомогти в усунені проблем з безпекою, а також виявленню підмінених образів. Це можна зробити, якщо використати однаковий базовий образ і такий саме тег, що і цільовий. Проте sha256 код, що присвоюється кожному шару є унікальним, і за його наявнісю можна визначати цей параметр.

Доцільно об'єднувати результати кількох сканерів. Використовуючи кілька інструментів для сканування, можна значно підвищити точність і повноту перевірки, що сприяє виявленню більшої кількості потенційних проблем. Це, в свою чергу, дозволяє розробникам своєчасно реагувати на виявлені ризики та забезпечувати більш надійну та безпечну роботу контейнеризованих додатків.

Комбінація кількох сканерів дозволяє отримати більш комплексне та детальне бачення стану безпеки. Це важливо, оскільки кожен сканер має свої сильні та слабкі сторони, а їх поєднання допомагає знизити ризик пропуску критичних вразливостей.

Також це дає можливість підтвердження результатів. Результати одного сканера можуть бути підтверджені іншим, що зменшує ймовірність хибно позитивних або хибно негативних результатів. Це підвищує точність виявлення вразливостей і дозволяє краще оцінити реальний стан безпеки системи. Прикладом таких результатів може бути сканування mysql@ sha256:a8298c3d874784d92f5f34f279a6fbf92c6ba69628548bb380db29f2bca9dd18 за допомогою двох різних сканерів. Trivy знаходить 34 вразливості критичного і високих рівнів, в той час як snyk – 6.

Притому snyk помітив проблеми тильки з openssl та openssl-libs, а trivy з stdlib. Таким чином ми можемо вважати що пошук вразливостей openssl та openssl-libs для trivy дав хибно позитивні результати і, відповідно, пошук вразливостей stdlibза допомогою snyk дав хибно позитивні результати. Вразливості, які можуть містити негативні результати, варто перевіряти в ручному режимі. Так як деякі з них можуть бути не актуальними для розробника в конкретний момент часу. До прикладу в зазначеному вище зразку trivy знайшов вразливість під номером CVE-2023-24538, що свідчить про некоректне виконання команд GO в середині JavaScript, що є рідкісним випадком і не потребує трактування як критичне [23]. Проте наш сканер позначив цю вразливість як Critical, що ми можемо вважати хибно негативним результатом і ігнорувати дану вразливість. Це допоможе більш точно налаштовувати оточення і сам сканер під час аналізу. Різні сканери можуть бути спеціалізовані на певних аспектах безпеки, таких як аналіз коду, сканування контейнерів, перевірка залежностей або конфігурацій. Тому їх результати можуть відрізнятись, але використання кількох інструментів забезпечує охоплення всіх цих аспектів, що сприяє більш повному аналізу і усуненню потенційних загроз.

Ще однією важливою причиною є можливість кращої пріоритизації вразливостей. Різні інструменти можуть мати різні підходи до оцінки ризиків та пріоритизації вразливостей. Використання кількох джерел дозволяє краще зрозуміти, які вразливості є найбільш критичними і вимагають негайного усунення, що оптимізує процес реагування на загрози.

Таким чином, перегляд результатів кількома сканерами забезпечує більш надійний, комплексний та детальний аналіз безпеки, допомагаючи виявити та усунути максимально можливу кількість вразливостей.

Важливо зазначити, що для деяких образів жодним сканером не було виявлено вразливостей високого та критичного рівнів. Це свідчить, що таких не має в даному образі контейнера і їх можна використовувати як базовий шар для створення власних. Серед перевірених образів, винятком є лише ruby. Критичні вразливості якого виявлено навіть за допомогою навіть Docker Scout. Цей сканер працює гірше за усіх інших представників. Хоча його доцільно використовувати як базову перевірку образу, перед його використанням. Головне в такому випадку, не брати образи з тегом latest. Так, як він може бути замінений в будь-який момент, а отже результати перевірки попередньої версії вже не будуть дійсні.

7. Висновки

Вибір інструменту повинен базуватися на комплексному аналізі його можливостей, точності, частоті оновлення бази вразливостей та відповідності конкретним потребам організації, а не лише на кількісних показниках.

Проведення аналізу сканування образів Docker на предмет вразливостей та проблем безпеки за допомогою об’єднання масивів з результатами аналізу двома і більше сканерами дозволяє зробити кілька ключових висновків. В першу чергу, такий аналіз більш точно визначає вразливості образу, які можуть бути присутні і в подальших нашаруваннях. За допомогою цього аналізу можна оцінити рівень ризику, пов'язаний з кожною знайденою вразливістю, враховуючи його можливі наслідки для безпеки системи та інформаційної безпеки в цілому.

Суть цього підходу полягає у використанні одночасно кількох сканерів. Вивід у форматі json дає можливість програмно порівняти виводи, виділити, наприклад, лише критичні вразливості і уникнути проблем з порушенням периметру безпеки підприємства в подальшому. Також це підхід є досить гнучким, так як інженер має можливість самостійно налаштовувати алгоритм за допомогою якого створюються результуючі масиви. Також запис вже проаналізованих результатів дозволяє зменшити об’єми файлів для збереження і вивчення тенденції розвитку конкретного образу в умовах завчасно виставлених методів аналізу результатів. Іншим варіантом використання такого підходу є вибір найкращого сканера для образу з яким працює інженер. Ми можемо порівняти всі доступні сканери і обрати найкращий з них.

Крім того, результати аналізу вразливостей конкретного образу можуть вказати на необхідність вдосконалення заходів безпеки, таких як оновлення бібліотек або налаштування конфігурації. В іншому випадку це може викликати серйозні ризики.

Список літератури

1. A. Ahmed and G. Pierre, « Docker-pi: Docker container deployment in fog computing infrastructures, » International Journal of Cloud Computing, vol. 1, no. 6, 2019. DOI: 10.1109/EDGE.2018.00008

2. T. Alyas, S. Ali, H. Khan, A. Samad, K. Alissa ma M. A. Saleem, «Container Performance and Vulnerability Management for Container Security Using Docker Engine, » Security and Communication Networks, 2022. DOI: 10.1155/2022/6819002

3. V. Jain, B. Singh, M. Khenwar ma M. Sharma, «Static Vulnerability Analysis of Docker Images» в IOP Conference Series: Materials Science and Engineering, Jaipur, India, 2021. DOI: 10.1088/1757-899X/1131/1/012018

4. Efe, Doç. Dr. Ahmet & Aslan, Ulaş & Kara, Aytekin. (2020). Securing Vulnerabilities in Docker Images. International Journal of Innovative Engineering Applications. 4. 31-39. DOI: 10.46460/ijiea.617181.

5. Over 30% of Official Images in Docker Hub Contain High Priority Security Vulnerabilities. (n.d.). Retrieved from banyanops Available https://www.banyansecurity.io/blog/over-30-of-official-images-in-docker-hub-contain-high-priority-security-vulnerabilities/ (Accessed: 18 March 2024)

6. R. A. Martin, "Managing vulnerabilities in networked systems," in Computer, vol. 34, no. 11, pp. 32-38, Nov. 2001, DOI: 10.1109/2.963441.

7. Docker, Inc., «Overview of Docker Desktop» Docker, Inc., [Онлайновий]. Available: https://docs.docker.com/desktop/*. (Accessed: 18 March 2024)*

8. C. Hashemi-Pour, S. J. Bigelow ma M. Courtemanche «DEFINITION Docker, » TechTarget., [Онлайновий]. Available: https://www.techtarget.com/searchitoperations/definition/Docker/. (Accessed: 18 March 2024)

9. Five Security concerns when using docker. (n.d.). Retrieved from Oreilly [Онлайновий]. Available: https://www.oreilly.com/ideas/five-security-concerns-when-using-docker (Accessed: 18 March 2024)

10. Aqua Security Software Ltd., «Data Sources - Trivy,» Aqua Security Software Ltd., [Онлайновий]. Available: https://aquasecurity.github.io/trivy/v0.32/docs/vulnerability/detection/data-source/. (Accessed: 18 March 2024)

11. Aqua Security Software Ltd., «Data Sources - Trivy,» Aqua Security Software Ltd., [Онлайновий]. Available: https://aquasecurity.github.io/trivy/v0.32/docs/vulnerability/detection/data-source/. (Accessed: 18 March 2024)

12. Canonical Ltd., «CVE reports, » Canonical Ltd., [Онлайновий]. Available: https://ubuntu.com/security/cves/. (Accessed: 18 March 2024)

13. Amazon Web Services, Inc., «Amazon Linux Security Center, » Amazon Web Services, Inc., [Онлайновий]. Available: https://alas.aws.amazon.com/. (Accessed: 18 March 2024)

14. P. Doan ma S. Jung, «DAVS: Dockerfile Analysis for Container Image Vulnerability Scanning» Computers, Materials & Continua, т. 72, № 1, pp. 1699-1711, 2022. DOI: 10.32604/cmc.2022.025096

15. S. Ugale ma A. Potgantwar, «Container Security in Cloud Environments: A Comprehensive Analysis and Future Directions for DevSecOps, » в RAiSE, Woodhouse, Leeds, 2023. DOI: 10.3390/engproc2023059057

16. D. Huang, H. Cui, S. Wen and C. Huang, "Security Analysis and Threats Detection Techniques on Docker Container," 2019 IEEE 5th International Conference on Computer and Communications (ICCC), Chengdu, China,2019, pp. 1214-1220, DOI: 10.1109/ICCC47050.2019.9064441

17. K. Brady, S. Moon, T. Nguyen and J. Coffman, "Docker Container Security in Cloud Computing" 2020 10th Annual Computing and Communication Workshop and Conference (CCWC), Las Vegas, NV, USA, 2020, pp. 0975-0980, DOI: 10.1109/CCWC47524.2020.9031195.

18. JFrog Ltd., «JFROG Artifactory» JFrog Ltd., [Онлайновий]. Available:

https://jfrog.com/artifactory/. (Accessed: 18 March 2024)

19. JFrog Ltd., «JFrog Xray» JFrog Ltd., [Онлайновий]. Available: https://jfrog.com/help/r/get-started-with-the-jfrog-platform/jfrog-xray. (Accessed: 18 March 2024)

20. Aqua Security Software Ltd., «Trivy Documentation,» Aqua Security Software Ltd., [Онлайновий]. Available: https://aquasecurity.github.io/trivy/v0.49/. (Accessed: 18 March 2024)

21. Snyk Limited, «Snyk Vulnerability Database», Snyk Limited [Онлайновий]. Available: https://security.snyk.io/. (Accessed: 18 March 2024)

22. Docker, Inc., «Docker Hub» Docker, Inc., [Онлайновий]. Available: https://hub.docker.com/. (Accessed: 18 March 2024)

23. Red Hat, Inc., «CVE-2023-24538» Red Hat, Inc., [Онлайновий]. Available: https://access.redhat.com/security/cve/cve-2023-24538. (Accessed: 18 March 2024)

DOCKER CONTAINER IMAGE SCANNING METHODS ON THE SUBJECT OF SECURITY VULNERABILITIES

D. Drienko, N. Kohut

Lviv Polytechnic National University,
Department of Information Protection

© Darienko D., Kohut N., 2024

With the development of containerized environments, the issue of security is becoming critical for application deployments. This article provides a comparative analysis of static and dynamic methods for scanning Docker container images. Static analysis is used to identify potential vulnerabilities before container deployment, while dynamic analysis is performed in an isolated environment at runtime, ensuring product reliability. The work of Trivy, JFrog Xray, Snyk, and Docker Scout scanners is compared, and their advantages, disadvantages, and effectiveness in different conditions are emphasized. Trivy has been proven to find the most vulnerabilities among the scanners tested. Snyk and Xray give similar results, but Xray also checks for encryption of important data such as passwords. Docker Scout turned out to be the weakest representative, the only advantage of which is open access to results that can be analyzed without uploading an image to the server or personal developers' computer. Particular attention is paid to static analysis due to its broader coverage of vulnerabilities, including operating packages and application dependencies. The difference in the assessment of the criticality of vulnerabilities by different scanners is demonstrated, and it is also discussed how many vulnerabilities found do not always mean a high level of risk. Based on the analysis, criteria for choosing a scanner are proposed to avoid information leakage due to unnoticed vulnerabilities.

Keywords: information protection, cybersecurity, container, docker, scanning, security vulnerability.