for Standardization, Geneva, Switzerland, 2022, Режим доступу:.https://www.iso.org/home.html
6. ISO/IEC 27002:2022. Information technology – Security techniques – Code of practice for information security controls. International Organization for Standardization, Geneva, Switzerland, 2022, https://www.iso.org/home.html

DOI 10.70286/EOSS-05.01.2026.005.50-57

КОНТЕЙНЕРИЗАЦІЯ ТА БЕЗПЕКА: АНАЛІЗ ЗАГРОЗ І МЕХАНІЗМІВ ЗАХИСТУ У СЕРЕДОВИЩАХ DOCKER ТА KUBERNETES

Пономаренко Владислав
аспірант
Коляда Андрій
к.т.н.
Кафедра інформаційних технологій проєктування та дизайну
НУ «Одеська політехніка», Україна

Анотація. У статті виконано системний аналіз загроз інформаційній безпеці контейнеризованих середовищ та обґрунтовано механізми захисту при використанні платформ Docker і Kubernetes у сучасних інформаційних системах. Показано, що спільне використання ядра операційної системи та висока динамічність контейнерної інфраструктури формують специфічну поверхню атаки, відмінну від традиційної віртуалізації. Загрози розглянуто як багаторівневу систему з урахуванням ієрархії: ризики, пов'язані з контейнерними образами (походження, цілісність, застарілі компоненти, наявність секретів), загрози середовища виконання (ескалація привілеїв, container escape, небезпечні режими запуску), уразливості рівня оркестрації Kubernetes (компрометація API, помилки RBAC, витік секретів) та мережеві ризики (відсутність сегментації, бічне переміщення, перехоплення/підміна трафіку). Обґрунтовано доцільність комплексного підходу до захисту на основі принципів security by design і defense in depth, що охоплює весь життєвий цикл контейнерного застосунку: мінімізацію та сканування образів, контроль цілісності, ізоляцію й обмеження привілеїв (namespaces, cgroups, seccomp/AppArmor), коректне налаштування RBAC і керування секретами, впровадження мережевих політик, а також моніторинг і журналювання подій безпеки. Отримані результати можуть бути використані для аудиту та проєктування захищених Docker/Kubernetes-інфраструктур у корпоративних і промислових середовищах.

Ключові слова: контейнеризація, Docker, Kubernetes, інформаційна безпека, оркестрація контейнерів, security by design, defense in depth.

Введення. Контейнеризація стала однією з ключових технологій сучасної IT-інфраструктури, забезпечуючи ізольоване середовище виконання застосунків, високу масштабованість і спрощення процесів розгортання. Платформи Docker і Kubernetes активно використовуються у хмарних сервісах, мікросервісних архітектурах, системах безперервної інтеграції та доставки програмного забезпечення (CI/CD), а також у корпоративних і промислових інформаційних системах [1, 2]. На відміну від традиційної віртуалізації, контейнерні середовища працюють на рівні операційної системи, спільно використовуючи її ядро. Такий підхід суттєво зменшує накладні витрати, однак одночасно формує новий клас загроз інформаційній безпеці. Порушення ізоляції контейнерів, уразливості образів, небезпечні конфігурації оркестратора або компрометація API керування можуть призвести до повного захоплення хост-системи чи кластера [3]. Особливу складність становить те, що Kubernetes додає над контейнерною платформою ще один рівень абстракції — систему оркестрації, яка автоматизує масштабування, балансування навантаження та відмовостійкість. Разом із цим зростає й поверхня атаки: з’являються ризики, пов’язані з керуванням ролями доступу (RBAC), зберіганням секретів, мережевими політиками та міжподовою взаємодією [4]. У багатьох практичних випадках інциденти безпеки спричинені не помилками реалізації платформи, а некоректною конфігурацією або відсутністю комплексного підходу до захисту контейнерної інфраструктури.

У сучасних умовах контейнерні середовища дедалі частіше використовуються для обробки чутливих даних і виконання критично важливих сервісів, що вимагає переходу від моделі «безпека після розгортання» до підходу security by design. Це передбачає інтеграцію механізмів захисту на всіх етапах життєвого циклу контейнерного застосунку — від створення образу та перевірки його вразливостей до контролю виконання в кластері Kubernetes і моноказу аномальної активності [5].

Таким чином, актуальним є системний аналіз загроз контейнеризованих середовищ і механізмів їхнього нейтралізування з урахуванням особливостей Docker і Kubernetes. Такий аналіз дозволяє сформувати цілісне уявлення про ризики контейнерної інфраструктури та визначити практичні підходи до підвищення її захищеності в корпоративних інформаційних системах.

Мета та задачі дослідження. Метою дослідження є системний аналіз загроз інформаційній безпеці контейнеризованих середовищ та обґрунтування ефективних механізмів захисту при використанні платформ Docker та Kubernetes у сучасних інформаційних системах.

Досягнення поставленої мети передбачає комплексний розгляд контейнерної інфраструктури як багаторівневої системи, що охоплює рівень контейнерних образів, середовище виконання, мережеву взаємодію та рівень оркестрації. Особлива увага приділяється питанням ізоляції, контролю доступу та конфігураційної безпеки, оскільки саме ці аспекти найчастіше стають

джерелом інцидентів у практичних сценаріях експлуатації контейнерних платформ [3, 4].

На відміну від традиційних підходів до захисту віртуалізованих середовищ, контейнеризація вимагає адаптації моделей безпеки з урахуванням спільного використання ядра операційної системи та високої динамічності інфраструктури. У зв'язку з цим дослідження спрямоване не лише на перелік загроз, а й на аналіз їхнього походження та взаємозв'язків між окремими рівнями контейнерної екосистеми [1, 5].

Для досягнення поставленої мети в роботі сформульовано такі завдання дослідження:

1. Проаналізувати архітектурні особливості контейнерних середовищ, що впливають на формування поверхні атаки, зокрема механізми ізоляції контейнерів, взаємодію з ядром операційної системи та роль оркестратора.

2. Класифікувати основні загрози безпеці контейнеризованих систем, пов'язані з використанням контейнерних образів, середовища виконання Docker та інфраструктури керування Kubernetes.

3. Дослідити типові вектори атак у контейнерних середовищах, зокрема атаки через уразливі образи, компрометацію API, порушення політик доступу та некоректну мережеву сегментацію.

4. Розглянути наявні механізми захисту, що реалізуються на рівні Docker і Kubernetes, включаючи контроль привілеїв, керування ролями, мережеві політики та захист конфігураційних даних.

5. Оцінити ефективність комплексного підходу до безпеки, заснованого на принципах security by design і defense in depth, для зниження ризиків експлуатації контейнерних середовищ у корпоративних інформаційних системах.

Реалізація зазначених завдань дозволяє сформувати узагальнену модель загроз контейнерної інфраструктури та визначити практичні рекомендації щодо підвищення рівня її захищеності. Отримані результати можуть бути використані як у процесі проєктування нових систем, так і при аудиті безпеки вже розгорнутих контейнерних середовищ.

Результати дослідження і їх обговорення. Контейнеризовані середовища характеризуються багаторівневою архітектурою, у межах якої кожен рівень формує власні загрози безпеці. На відміну від монолітних систем, інциденти в контейнерній інфраструктурі часто мають каскадний характер: уразливість на одному рівні може бути використана для ескалації привілеїв і компрометації всієї платформи. У зв'язку з цим доцільно розглядати загрози з урахуванням ієрархії контейнерного середовища [3, 6].

Загрози, пов'язані з контейнерними образами. Контейнерні образи є початковою точкою розгортання будь-якого застосунку, тому їхня безпека безпосередньо впливає на захищеність усієї інфраструктури. Однією з найпоширеніших загроз є використання образів із відкритих репозиторіїв без належної перевірки їхнього походження та вмісту. Такі образи можуть містити

застарілі бібліотеки, відомі вразливості або навіть вбудований шкідливий код [1, 5]. Додаткову небезпеку становить надмірна складність образів. Використання базових образів загального призначення (наприклад, з повноцінною операційною системою) збільшує поверхню атаки за рахунок невикористовуваних пакетів і сервісів. Уразливості в цих компонентах можуть бути використані зловмисником для проникнення в контейнер, навіть якщо сам застосунок реалізований коректно [7]. До типових загроз цього рівня належать:

- відсутність перевірки цілісності та автентичності образів;
- використання застарілих або вразливих бібліотек;
- зберігання облікових даних і ключів безпосередньо в образі.

Загрози середовища виконання контейнерів. На рівні виконання контейнерів ключовим ризиком є порушення механізмів ізоляції між контейнерами або між контейнером і хост-системою. Оскільки контейнери спільно використовують ядро операційної системи, експлуатація вразливостей ядра або неправильна конфігурація механізмів ізоляції може призвести до так званого container escape — виходу за межі контейнера з отриманням доступу до хосту [3]. Особливо небезпечними є сценарії запуску контейнерів із підвищеними привілеями або з доступом до системних ресурсів хоста. Використання режиму privileged, монтування файлової системи хоста чи передавання пристроїв без належних обмежень створює умови для повної компрометації системи [6]. До основних загроз цього рівня належать:

- ескалація привілеїв у межах контейнера;
- атаки на ядро операційної системи;
- неконтрольований доступ до ресурсів хоста.

Загрози рівня оркестрації Kubernetes. Система оркестрації Kubernetes суттєво розширює функціональні можливості контейнерної платформи, проте водночас ускладнює модель безпеки. Центральним елементом управління є API-сервер, через який здійснюється керування кластером. Некоректна конфігурація доступу до API або використання надмірних привілеїв може призвести до повного захоплення кластера [4, 6]. Окрему категорію ризиків становлять помилки в налаштуваннях механізмів контролю доступу. Неправильно сконфігуровані ролі та прив’язки RBAC дозволяють сервісним обліковим записам виконувати дії, які виходять за межі їхніх функціональних обов’язків. У поєднанні з уразливостями застосунків це створює сприятливі умови для горизонтального або вертикального переміщення зловмисника в кластері [4]. Також суттєвим фактором ризику є зберігання конфіденційних даних (паролів, токенів, ключів) у відкритому вигляді в конфігураційних файлах або змінних середовища контейнерів. У разі компрометації поду такі дані можуть бути використані для подальших атак на інші компоненти системи [5].

Мережеві загрози у контейнерних кластерах. Мережеві взаємодії між контейнерами в Kubernetes за замовчуванням є досить відкритими, що спрощує розгортання сервісів, але водночас збільшує ризики бічного переміщення

атакуючого. Відсутність мережевої сегментації дозволяє зловмиснику, який отримав доступ до одного поду, взаємодіяти з іншими сервісами кластера [6, 7]. Додаткові загрози пов'язані з неправильною експозицією сервісів назовні, використанням незахищених протоколів або відсутністю контролю трафіку між компонентами. У таких умовах атаки типу man-in-the-middle, підміна сервісів або перехоплення даних стають реалістичними сценаріями [3].

Механізми захисту контейнеризованих середовищ Docker та Kubernetes. Ефективний захист контейнеризованих середовищ вимагає реалізації багаторівневої моделі безпеки, яка охоплює всі етапи життєвого циклу контейнерного застосунку — від створення образу до його виконання в кластері. У цьому контексті ключовим є поєднання вбудованих механізмів платформ Docker та Kubernetes з організаційними та інженерними практиками security by design і defense in depth [2, 4].

Захист на етапі створення та розповсюдження контейнерних образів. Першим рівнем захисту контейнерного середовища є контроль безпеки контейнерних образів. Основним принципом тут є мінімізація поверхні атаки шляхом використання полегшених базових образів і виключення зайвих компонентів. Практика minimal base image дозволяє зменшити кількість потенційних уразливостей і спростити аудит безпеки [1, 7]. Важливу роль відіграє автоматизоване сканування образів на наявність відомих вразливостей. Інтеграція засобів аналізу безпеки в CI/CD-конвеєри дає змогу виявляти проблеми ще до розгортання застосунку в продуктивному середовищі. Такий підхід знижує ризик використання уразливих бібліотек і сприяє підвищенню загального рівня захищеності контейнерної інфраструктури [5]. Додатковим механізмом є забезпечення цілісності та автентичності образів за допомогою цифрового підпису. Перевірка підписів при завантаженні образів із реєстрів дозволяє запобігти використанню підроблених або змінених контейнерів.

Механізми ізоляції та контролю привілеїв. На рівні виконання контейнерів ключове значення мають механізми ізоляції процесів і обмеження доступу до ресурсів хост-системи. Docker використовує namespaces і cgroups для розмежування доступу контейнерів до процесів, файлової системи, мережі та обчислювальних ресурсів. Коректне налаштування цих механізмів дозволяє суттєво знизити ризик ескалації привілеїв [3]. Особливу увагу слід приділяти принципу найменших привілеїв. Запуск контейнерів без прав суперкористувача, заборона використання режиму privileged і обмеження доступу до системних викликів за допомогою профілів seccomp та AppArmor істотно підвищують рівень безпеки середовища виконання [6]. У Kubernetes ці механізми доповнюються політиками безпеки подів, які визначають допустимі параметри запуску контейнерів. Такий підхід дозволяє централізовано контролювати вимоги до безпеки й запобігати небезпечним конфігураціям на рівні всього кластера.

Контроль доступу та захист рівня оркестрації. Рівень оркестрації є одним із найкритичніших з погляду безпеки, оскільки компрометація керуючих

компонентів Kubernetes може призвести до повної втрати контролю над кластером. Основним механізмом захисту тут є коректне налаштування керування доступом на основі ролей (RBAC). Чітке розмежування прав користувачів і сервісних облікових записів зменшує ризик зловживань і помилок конфігурації [4]. Не менш важливим є захист конфігураційних даних і секретів. Зберігання облікових даних у відкритому вигляді або передавання їх через змінні середовища контейнерів створює додаткові ризики. Використання спеціалізованих механізмів керування секретами дозволяє зменшити ймовірність їх компрометації у разі успішної атаки на окремий компонент системи [5].

Мережеві механізми безпеки та моніторинг. Мережевий рівень відіграє вирішальну роль у запобіганні горизонтальному переміщенню зловмисника в межах кластера. Застосування мережевих політик дозволяє обмежити взаємодію між подами та сервісами відповідно до принципу найменших необхідних зв'язків. Це суттєво ускладнює реалізацію атак на основі бічного переміщення [6, 7]. Доповненням до превентивних заходів є постійний моніторинг і журналювання подій безпеки. Аналіз журналів, виявлення аномальної поведінки контейнерів і контроль змін конфігурації дозволяють своєчасно реагувати на інциденти та мінімізувати їхні наслідки. Такий підхід є невід'ємною складовою сучасних систем захисту контейнеризованих середовищ.

На рис. 1 подано узагальнену архітектуру безпеки контейнеризованого середовища, що охоплює основні рівні захисту під час використання Docker і Kubernetes. На рівні контейнерних образів реалізується контроль цілісності та перевірка наявності відомих уразливостей, що знижує ризик розгортання небезпечних компонентів. Рівень виконання контейнерів забезпечує ізоляцію процесів і ресурсів хост-системи за допомогою механізмів ядра операційної системи, а також обмеження привілеїв контейнерів.

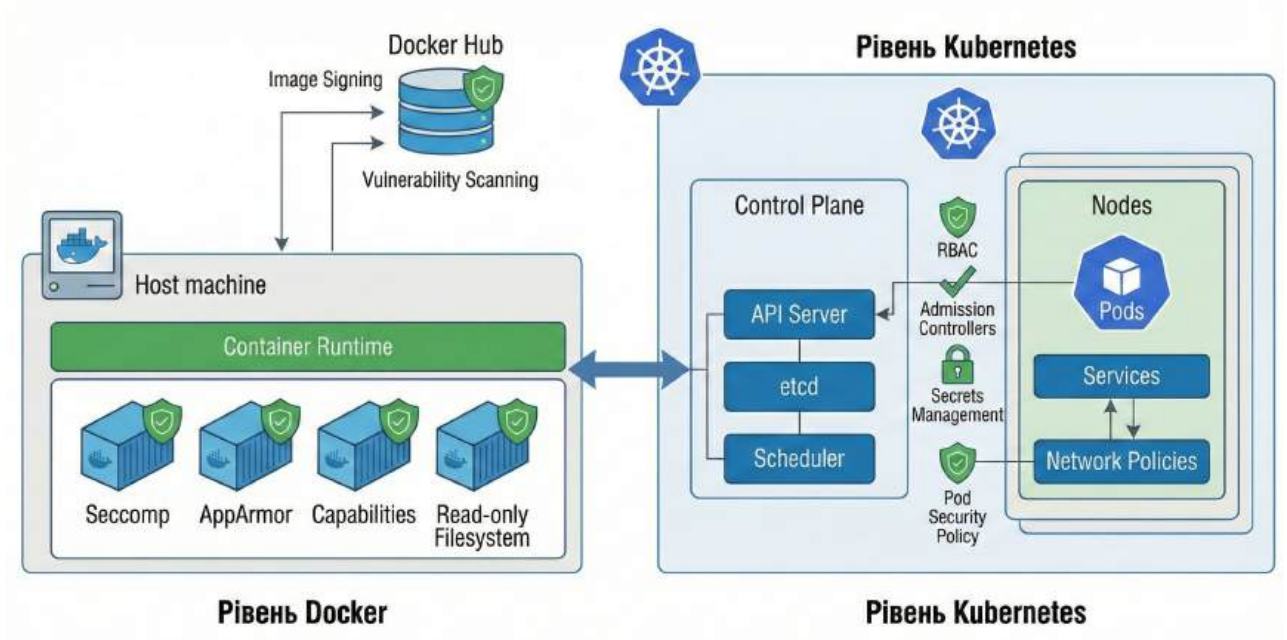

Рисунок 1. Архітектура безпеки контейнеризованого середовища Docker та Kubernetes

На рівні оркестрації Kubernetes застосовуються механізми керування доступом на основі ролей, контроль конфігурацій та захист секретів, що унеможливлює несанкціонований вплив на керування кластером. Мережевий рівень представлений політиками сегментації та контролю трафіку між подами, які зменшують ризик бічного переміщення зловмисника. Сукупність зазначених механізмів формує багаторівневу модель безпеки, що відповідає принципам defense in depth та забезпечує підвищення захищеності контейнерної інфраструктури.

Розглянуті механізми захисту демонструють, що безпека контейнерної інфраструктури не може бути забезпечена окремим інструментом або налаштуванням. Лише комплексне поєднання технічних і організаційних заходів дозволяє ефективно протидіяти загрозам у середовищах Docker і Kubernetes.

Висновки. У роботі здійснено системний аналіз загроз інформаційній безпеці контейнеризованих середовищ, що використовують платформи Docker та Kubernetes, з урахуванням їхніх архітектурних особливостей і практичних сценаріїв застосування в сучасних інформаційних системах. Показано, що контейнеризація, попри свої переваги у вигляді масштабованості та ефективного використання ресурсів, формує специфічну поверхню атаки, яка істотно відрізняється від традиційних віртуалізованих середовищ. У межах дослідження проаналізовано ключові групи загроз, пов’язані з контейнерними образами, середовищем виконання, рівнем оркестрації та мережевою взаємодією в кластері. Встановлено, що значна частина інцидентів безпеки зумовлена не вразливостями самих платформ, а помилками конфігурації, надмірними привілеями та відсутністю комплексного підходу до захисту контейнерної інфраструктури. Окрему увагу приділено механізмам протидії виявленим загрозам. Розглянуті засоби захисту — контроль безпеки контейнерних образів, ізоляція та обмеження привілеїв, керування доступом на основі ролей, мережеві політики й моніторинг — демонструють ефективність лише за умови їхнього узгодженого застосування. Це підтверджує доцільність використання принципів security by design та defense in depth при проєктуванні та експлуатації контейнеризованих середовищ.

Отримані результати свідчать, що безпека Docker- і Kubernetes-інфраструктур повинна розглядатися як безперервний процес, інтегрований у життєвий цикл розробки та розгортання програмного забезпечення. Запропонований аналітичний підхід може бути використаний як основа для аудитів безпеки, розроблення політик захисту та підвищення рівня захищеності контейнерних платформ у корпоративних інформаційних системах. Подальші дослідження доцільно спрямувати на аналіз автоматизованих засобів контролю безпеки контейнерних кластерів і їхню інтеграцію з системами управління подіями інформаційної безпеки.

Список використаних джерел

1. Merkel D. Docker: Lightweight Linux Containers for Consistent Development and Deployment / D. Merkel // Linux Journal. – 2014. – № 239.

2. Burns B. Borg, Omega, and Kubernetes / B. Burns, B. Grant, D. Oppenheimer, E. Brewer, J. Wilkes // ACM Queue. – 2016. – Vol. 14, № 1.
3. Pahl C. Containerization and the PaaS Cloud / C. Pahl // IEEE Cloud Computing. – 2015. – Vol. 2, № 3.
4. Kubernetes Documentation. Security Concepts [Електронний ресурс]. – CNCF, 2023. – Режим доступу: https://kubernetes.io/docs/concepts/security/ (дата звернення: 02.01.2026).
5. Zhang Q. Security Issues and Defense Strategies in Container-Based Cloud Environments / Q. Zhang, M. Chen, L. Li // IEEE Access. – 2022. – Vol. 10.
6. Shu R. A Study of Security Vulnerabilities on Docker Hub / R. Shu, X. Gu, W. Enck // Proceedings of the 7th ACM Conference on Data and Application Security and Privacy. – 2017.
7. Poncelet A. Securing Container-Based Infrastructures: Challenges and Solutions / A. Poncelet, E. Riviere // Journal of Cloud Computing. – 2021. – Vol. 10, № 1.

ARCHITECTURE OF VIRTUALIZED REMOTE LEARNING ENVIRONMENTS

Horval Dmytro
Ph.D., Teacher
Chernihiv Polytechnic National University, Ukraine

The delivery of technical education, particularly in the fields of data science, computer engineering, and computational sciences, has historically been plagued by the "environmental dependency" problem. In a traditional setting, instructors are forced to dedicate significant portions of their curriculum to helping students configure local development environments. This process involves navigating a matrix of operating systems (Windows, macOS, Linux), hardware architectures (x86, ARM), and conflicting library versions [1].

By leveraging a stack composed of JupyterHub, DockerSpawner, and nbgitpuller, it is possible to quickly deploy virtualized remote learning environments. These environments provide access to high-performance computing resources via a standard web browser, ensuring that every student interacts with an identical, reproducible, and professionally managed software stack [2].

The core of such environments is JupyterHub, a system that manages the lifecycle of single-user notebook servers through the interaction of three primary subsystems. The first is a high-performance HTTP proxy, which routes external traffic to the appropriate single-user container. This proxy relies on a dynamic routing table updated via a REST API, allowing routes to be added or removed seamlessly without dropping active connections [3]. Access to these resources is guarded by the Authenticator, a pluggable module handling identity verification. In academic settings, this component