Аналіз засобів виявлення та протидії атакам типу container escape у середовищах Linux-контейнерів

УДК 004.056.5:004.75
Вікторія Шумська[1], Юрій Дорофєєв[2], Ірина Назарова[3]

Національний університет «Одеська політехніка»,
[1]marfonya.999@stud.op.edu.ua, [2]dym@op.edu.ua, [3]nazarova.i.v@op.edu.ua

У сучасних умовах переходу інформаційних систем до хмарних платформ питання безпеки хмарної інфраструктури набуває особливої актуальності. Однією з базових технологій таких середовищ є контейнеризація, що забезпечує ізоляцію застосунків, спрощує їх розгортання, масштабування та перенесення між різними середовищами. Водночас атаки, пов'язані з порушенням контейнерної ізоляції, виникають дедалі частіше та можуть призводити до компрометації хостової системи. Метою роботи є аналіз основних векторів атак типу container escape у Linux-контейнерах та підходів до їх запобігання.

Контейнеризація є формою віртуалізації на рівні операційної системи, за якої кілька ізольованих середовищ виконання використовують спільне ядро хостової системи. Основними механізмами ядра Linux, що застосовуються для реалізації контейнеризації, є простори імен (namespaces), контрольні групи (cgroups), привілеї (capabilities) та фільтрація системних викликів (seccomp) [1].

Однією з найбільш небезпечних загроз для контейнерної ізоляції є атака типу container escape, тобто вихід процесу за межі ізольованого середовища контейнера з подальшим отриманням доступу до ресурсів хостової системи або інших контейнерів. У дослідженні [2] відокремлено три основні джерела атак типу container escape: небезпечні конфігурації, уразливості компонентів контейнерної інфраструктури та вразливості ядра Linux.

Небезпечні конфігурації послаблюють фактичну ізоляцію контейнера та можуть створювати умови для несанкціонованого доступу до ресурсів хоста або підвищення привілеїв. Для їх виявлення використовують засоби перевірки декларативних конфігурацій контейнерної інфраструктури, зокрема Docker Bench for Security, kube-bench, Checkov і Conftest. Зниження ризику базується на застосуванні принципу найменших привілеїв: відмові від privileged-режиму, мінімізації capabilities, застосуванні user namespace, обмеженні host mounts, runtime-сокетів і ресурсів через cgroups.

Уразливості компонентів контейнерної інфраструктури виникають через помилки в коді runtime-компонентів або використання вразливих залежностей в образах. Показовим прикладом є вразливість CVE-2024-21626 у runc, яка могла дозволити процесу контейнера отримати доступ до файлової системи хоста [3]. Ризики, що надходять з рівня контейнерних образів, можуть створювати умови для подальшої компрометації хостової системи, використовуючи застарілі пакети, вбудовані секрети, недовірені базові образи, скомпрометовані або шкідливі образи в реєстрах, а також небезпечні інструкції збірки. Для протидії атакам, що базуються на уразливостях компонентів контейнерної інфраструктури, застосовують регулярне оновлення runtime-компонентів, використання довірених реєстрів і базових образів, підписування

та перевірку цілісності образів, SBOM, а також сканування за допомогою Trivy, Clair, Anchore, Grype та Snyk Container.

Уразливості ядра Linux є особливо небезпечними для контейнерних середовищ через спільне використання ядра контейнерами та хостовою системою. Прикладом є вразливість CVE-2022-0847 Dirty Pipe [4], яка дозволяла непривілейованому локальному користувачу записувати дані у сторінки page cache, пов'язані з read-only файлами, і таким чином підвищувати привілеї в системі. Основним засобом захисту від подібних атак є своєчасне оновлення ядра. Додатковим рівнем захисту може бути runtime-моніторинг системних викликів і подій ядра за допомогою Falco, Tracee та Tetragon.

Небезпечні конфігурації, уразливості компонентів контейнерної інфраструктури та уразливості ядра Linux можуть взаємно посилювати наслідки одне одного, створюючи умови для порушення меж ізоляції. Наприклад, у кампанії, описаній Aqua Security, атака поєднувала використання зловмисного контейнерного образу з небезпечною конфігурацією контейнера. Така комбінація дозволила шкідливому скрипту всередині контейнера використати механізм cgroup release_agent для виконання коду на хості [5].

Проведений аналіз показав, що зниження ризику атак типу container escape потребує комплексного підходу. Поєднання визначення фактичної ізоляції контейнера на рівні механізмів ядра Linux із runtime-моніторингом системних викликів і подій ядра на основі eBPF в єдиному програмному рішенні є перспективним напрямом формування комплексної оцінки поточного рівня захищеності контейнеризованого середовища. Такий підхід дозволяє враховувати не лише формальні параметри запуску контейнера, а й його фактичний стан під час виконання: належність процесів до просторів імен, обмеження cgroups, набір capabilities, режим seccomp, а також поведінкові ознаки, що проявляються через системні виклики та події ядра.

1. Sultan S., Ahmad I., Dimitriou T. Containers' Security: Issues, Challenges, and Road Ahead. IEEE Access. 2019. Vol. 7. P. 52976–52996. DOI: 10.1109/ACCESS.2019.2911732.
2. Chen K., Zhao Y., Guo J., Gu Z., Han L., Tang K. A Container Escape Detection Method Based on a Dependency Graph. Electronics. 2024. Vol. 13, № 23. Article 4773. DOI: 10.3390/electronics13234773.
3. National Vulnerability Database. CVE-2024-21626 Detail. URL: https://nvd.nist.gov/vuln/detail/CVE-2024-21626 (дата звернення: 04.05.2026).
4. National Vulnerability Database. CVE-2022-0847 Detail. URL: https://nvd.nist.gov/vuln/detail/CVE-2022-0847 (дата звернення: 04.05.2026).
5. Eitani A. Threat Actors Using release_agent Container Escape. Aqua Security. 03.11.2021. URL: https://www.aquasec.com/blog/threat-alert-container-escape/ (дата звернення: 04.05.2026).