КІБЕРБЕЗПЕКА: освіта, наука, техніка
CYBERSECURITY: EDUCATION, SCIENCE, TECHNIQUE
№ 4 (32), 2026
ISSN 2663 – 4023
DOI 10.28925/2663-4023.2026.32.1188
УДК: 004.7;004.4

Дарієнко Дмитро Геннадійович
аспірант кафедри захисту інформації
Національний університет "Львівська політехніка", Львів, Україна

Когут Назарій Мирославович
аспірант кафедри захисту інформації
Національний університет "Львівська політехніка", Львів, Україна

РОЗШИРЕННЯ ФУНКЦІЙ OPEN POLICY AGENT

Анотація. У роботі розглянуто підходи до інтеграції системи виявлення загроз у реальному часі Falco з механізмом політик Open Policy Agent (OPA) у середовищі Kubernetes. Проаналізовано три основні моделі взаємодії: безпосереднє надсилання подій через Falcosidekick у HTTP API OPA, застосування OPA як admission controller для блокування небезпечних конфігурацій на етапі створення ресурсів, а також використання проміжного сервісу як зв'язувальної ланки між Falco та OPA з можливістю складної обробки подій. Дослідження демонструє, що комбінування механізмів сенсорного контролю та політико-орієнтованого ухвалення рішень дозволяє досягти вищого рівня безпеки, автоматизації реагування та відповідності принципам Zero Trust. Поєднання Falco та Open Policy Agent формує багаторівневу модель безпеки, де рантайм-виявлення загроз доповнюється гнучким механізмом політик. Варіант прямого надсилання подій з Falco через Falcosidekick до HTTP-інтерфейсу OPA забезпечує оперативну реакцію і мінімальну затримку між фіксацією інциденту та ухваленням рішення, що робить його доцільним для кластерів із підвищеними вимогами до часу відгуку. Використання OPA у ролі admission controller посилює превентивний захист, оскільки блокує небажані об'єкти ще до їхнього розгортання, однак потребує постійного коригування Rego-політик на основі знань, що їх надає Falco. Сценарій із проміжним сервісом узгоджує переваги обох підходів: він дає можливість виконувати складну бізнес-логіку, вмикає обмежену ізоляцію підозрілих робочих навантажень і водночас не перевантажує OPA надмірною кількістю подій. Усі три схеми підтверджують, що інтеграція сенсорних та політико-орієнтованих компонентів суттєво підвищує рівень захисту контейнеризованих середовищ, сприяє реалізації принципів Zero Trust і створює передумови для самовідновної інфраструктури з автоматизованим контролем відповідності.

Ключові слова: kubernetes; динамічний доступ; контекстна безпека; OPA; Falco; інформаційна безпека, захист інформації.

ВСТУП

У сучасних хмарних інфраструктурах, зокрема в середовищах на базі Kubernetes, забезпечення динамічного, гнучкого та ефективного контролю безпеки набуває критично важливого значення. Зростання складності мікросервісних архітектур, частота змін у середовищі виконання та розширення векторів атак зумовлюють потребу у впровадженні багаторівневих підходів до захисту, які поєднують виявлення загроз і динамічне застосування політик доступу. Одним із найбільш перспективних підходів є концепція Zero Trust, що передбачає повну відмову від довіри до будь-яких компонентів системи без постійної перевірки їх дій. У цьому контексті інтеграція системи рантайм-

моніторингу Falco з механізмом політик Open Policy Agent відкриває нові можливості для створення адаптивного середовища безпеки. Falco забезпечує виявлення аномальної або небезпечної поведінки на рівні ядра операційної системи, тоді як OPA дозволяє формалізувати правила контролю доступу та реакції на події за допомогою мови політик Rego. Їхнє поєднання дозволяє реалізувати як реактивну, так і превентивну безпеку в Kubernetes-кластерах.

Метою даного дослідження є аналіз можливих моделей взаємодії між Falco та OPA, оцінка їх ефективності в умовах динамічного кластерного середовища та обґрунтування доцільності впровадження інтеграційного підходу для підтримки принципів Zero Trust.

Аналіз досліджень і публікацій. У сучасних інформаційних системах, зокрема в середовищах контейнеризації та хмарних обчислень, формалізація та автоматизація політик безпеки стає необхідною умовою забезпечення контрольованого доступу та динамічного реагування на загрози [4]. Політики розглядаються як декларативно задані правила, що визначають дозволені або заборонені дії в системі, а в межах підходу Policy-as-Code – як об'єкти програмного управління, що інтегруються у процеси розгортання та експлуатації інфраструктури [4, 5]. Це дозволяє здійснювати централізовану перевірку відповідності та автоматизоване управління поведінкою користувачів і сервісів за наперед визначеними правилами.

Актуальним є питання адаптивності політик, зокрема їх здатності реагувати на зміни контексту, поведінки або ризику. Хоча значна частина механізмів контролю реалізується на етапі обробки запитів до системи, цього недостатньо для повноцінного захисту в динамічних середовищах [1, 7]. Моніторинг поведінки на рівні системних викликів або подій виконання (runtime) дозволяє виявляти загрози, що виникають уже після проходження етапів автентифікації чи створення об'єктів. У поєднанні з механізмами оцінки політик, реалізованими через незалежні рушії (наприклад, OPA), така модель дозволяє здійснювати не лише фіксацію, а й автоматизоване реагування – шляхом видалення, ізоляції або зміни об’єктів [4, 5, 7].

У науковій літературі простежується тенденція до розробки політик, що ґрунтуються на багатовимірних критеріях: атрибутах, відносинах, історії дій та оцінці ризиків [1, 2, 3]. Однак найбільш актуальним у контексті побудови захищених кластерних систем є питання розподілу відповідальності між компонентами системи безпеки. Моделі, у яких прийняття рішення відокремлене від виконання дії, а політики реалізуються автономно від основного коду додатку, вважаються більш стійкими до порушень та гнучкими до змін [4, 6]. Такий підхід забезпечує реалізацію захисту без надання надмірних привілеїв компонентам моніторингу та прийняття рішень, що є критично важливим у хмарних середовищах і системах з високим рівнем розподілення.

Останні дослідження у сфері моніторингу інформаційної безпеки демонструють зміщення фокусу від реактивного виявлення загроз до впровадження проактивних та контекстно-обізнаних механізмів, що забезпечують безперервне спостереження за станом систем. У роботах розглядаються як загальні стратегії постійного моніторингу для захисту критичних активів, так і специфічні архітектури, орієнтовані на контейнеризовані середовища [8, 10]. Акцент робиться на необхідності забезпечення спостереження у реальному часі, підтримки актуальності політик безпеки та інтеграції аналітичних модулів, здатних виявляти відхилення від допустимої поведінки в момент їх виникнення.

Окремі підходи спрямовані на оцінювання ефективності програм моніторингу в організаціях та формалізацію критеріїв такої оцінки [11]. Ініціативи, орієнтовані на

практики управління політиками в контейнерних середовищах, зосереджуються на реалізації механізмів попереджувального реагування, де політики застосовуються ще до моменту порушення [8]. У специфічних галузях, зокрема в морській інфраструктурі, наголошується на важливості розробки інструментів, здатних адаптуватися до доменної специфіки, забезпечуючи одночасно як технічний, так і процедурний моніторинг [9]. Таким чином, в літературі поступово формується уявлення про моніторинг як про постійно діючий, адаптивний та інтегрований процес, що охоплює як технічні події, так і політичні аспекти забезпечення безпеки. Також існують розробки пов’язані з виявленням загроз за допомогою штучного інтелекту [12].

Варто наголосити, що виявлення загроз це лише частина процесу захисту середовища. Динамічні загрози, потребують негайного застосування політик безпеки, здатних блокувати, ізолювати або обмежувати дії, які не відповідають очікуваній поведінці. Тому впровадження механізмів, що дозволяють не лише виявляти інциденти, а й автоматизовано реагувати на них згідно з наперед визначеними політиками, є критично важливим для забезпечення цілісності, доступності та контрольованості сучасних інформаційних систем.

Постановка проблеми. Для реалізації етапу прийняття рішень до архітектури включається компонент Open Policy Agent (OPA). Цей інструмент забезпечує можливість декларативного визначення політик безпеки за допомогою мови Rego та реалізує так звану модель "Policy Decision Point" (PDP). У межах запропонованої постановки задачі OPA приймає на вхід подію і аналізує її відповідність одному або кільком правилам, які описують допустиму поведінку в системі [13]. Як генератор подій можна використовувати Falco - систему моніторингу, що аналізує події на рівні ядра операційної системи та виявляє поведінкові відхилення відповідно до заздалегідь визначених правил. Виявлена подія, яка відповідає шаблону загрози, генерується у вигляді структурованого повідомлення у форматі JSON [14]. Falco не виконує жодних дій самостійно, а передає ці події далі у зовнішній сервіс для обробки.

Open Policy Agent виконує перевірку на рівні запитів до API Kubernetes. Він діє як , тобто перевіряє запити користувача або системи на створення чи зміну об’єктів ще до того, як Kubernetes їх прийме [15]. Наприклад, якщо користувач намагається створити pod з увімкненим режимом привілейованого доступу, OPA може заборонити таку операцію на основі визначених політик, написаних на мові Rego. Таким чином, OPA дозволяє контролювати безпеку на рівні декларативної конфігурації кластера. Проте після створення ресурсу, коли він уже працює, OPA більше не спостерігає за його поведінкою. Він не бачить, що відбувається всередині контейнера або які дії виконує процес, який було дозволено на етапі створення [16].

Falco, навпаки, діє на рівні ядра операційної системи та здійснює моніторинг у режимі реального часу. Він відстежує системні виклики, активність процесів, взаємодію з файлами, мережею і подіями Kubernetes. Завдяки цьому Falco здатен виявляти небезпечну або аномальну поведінку, яка не описується в YAML-файлах і не перевіряється політиками OPA. Наприклад, якщо всередині вже створеного контейнера хтось виконує команду bash, встановлює пакети за допомогою пакетного менеджера або відкриває нестандартне з’єднання з мережею, Falco фіксує цю активність і може відреагувати на неї. Це стосується також виконання kubectl exec або kubectl port-forward, які технічно не є ресурсами [17].

Інтеграція Falco та Open Policy Agent можлива через кілька підходів, які відрізняються тим, на якому рівні вони працюють і як реагують на виявлені події.

Перший спосіб полягає у використанні Falcosidekick для пересилання подій, які виявляє Falco, безпосередньо в HTTP API, який обробляється OPA. Таким чином Falco виступає як генератор подій, а OPA ухвалює рішення на основі вхідної інформації. Цей варіант підходить для побудови реактивних механізмів, де OPA може реєструвати порушення, відправляти додаткові повідомлення. Такий підхід виключає можливість автоматичної реакції на подію.

Другий варіант передбачає використання OPA у вигляді admission controller у кластері. У цьому випадку політики, написані на Rego, блокують створення ресурсів ще до того, як вони з'являться в кластері. Однак тут прямої інтеграції з Falco немає. Але можна реалізувати взаємодію через спільну логіку: наприклад, Falco виявляє небезпечну поведінку, на основі якої адміністратор змінює політики в OPA, щоб у майбутньому такі дії блокувалися вже на рівні admission. Таким чином, Falco виконує роль сенсора, що підказує, як удосконалити політики в OPA. Такий підхід не є автоматичним.

Третій підхід – це обробка подій від Falco за допомогою проміжного сервісу, який виступає як сполучна ланка між Falco та OPA. У цьому сценарії події від Falco надсилаються через Falcosidekick до зовнішнього сервісу, який виконує HTTP-запит до OPA, отримує рішення і на його основі виконує певну дію, наприклад створення Kubernetes Event, оновлення політики або ізоляцію pod'a. Такий підхід є найбільш гнучким і дозволяє побудувати комплексну систему реакції.

РЕЗУЛЬТАТИ ДОСЛІДЖЕННЯ

Falco та Open Policy Agent виконують різні ролі у захисті Kubernetes-середовищ, і хоча вони можуть здаватися подібними за метою, насправді працюють на різних рівнях системної безпеки. Open Policy Agent або його інтеграція через Gatekeeper виконує перевірку на рівні запитів до API Kubernetes. Він діє як admission controller, тобто перевіряє запити користувача або системи на створення чи зміну об'єктів ще до того, як Kubernetes їх прийме. Наприклад, якщо користувач намагається створити pod з увімкненим режимом привілейованого доступу, OPA може заборонити таку операцію на основі визначених політик, написаних на мові Rego. Таким чином, OPA дозволяє контролювати безпеку на рівні декларативної конфігурації кластера. Проте після створення ресурсу, коли він уже працює, OPA більше не спостерігає за його поведінкою. Він не бачить, що відбувається всередині контейнера або які дії виконує процес, який було дозволено на етапі створення.

Falco, навпаки, діє на рівні ядра операційної системи та здійснює моніторинг у режимі реального часу. Він відстежує системні виклики, активність процесів, взаємодію з файлами, мережею і подіями Kubernetes. Завдяки цьому Falco здатен виявляти небезпечну або аномальну поведінку, яка не описується в YAML-файлах і не перевіряється політиками OPA. Наприклад, якщо всередині вже створеного контейнера хтось виконує команду bash, встановлює пакети за допомогою пакетного менеджера або відкриває нестандартне з'єднання з мережею, Falco фіксує цю активність і може відреагувати на неї. Це стосується також виконання kubectl exec або kubectl port-forward, які технічно не є ресурсами Kubernetes, а отже не проходять через admission контролери. У таких випадках саме Falco забезпечує виявлення постфактум, коли політики вже не діють.

Таким чином, OPA та Falco працюють у різних точках життєвого циклу подій. OPA перевіряє відповідність політикам до моменту створення ресурсу, а Falco аналізує те, що реально відбувається після створення. Їхня одночасна присутність у кластері

створює модель багаторівневого захисту, де один інструмент забезпечує запобігання помилкам або порушенням політик на рівні конфігурації, а інший контролює виконання і реагує на реальні інциденти. Такий підхід узгоджується з принципом глибокої оборони, коли різні механізми безпеки працюють у взаємодії та перекривають один одного в критичних точках.

На рисунку 1 зображено повний ланцюг взаємодії компонентів системи виявлення та реагування на потенційні загрози у Kubernetes-кластері. Дана діаграма ілюструє сценарій, у якому користувач або процес ініціює небажану дію, наприклад, спробу отримання привілейованого доступу або виконання команди всередині контейнера. Цей ланцюг взаємодії побудований на принципі реактивного аналізу, із залученням Falco, Open Policy Agent та додаткового логічного рівня – decision handler.

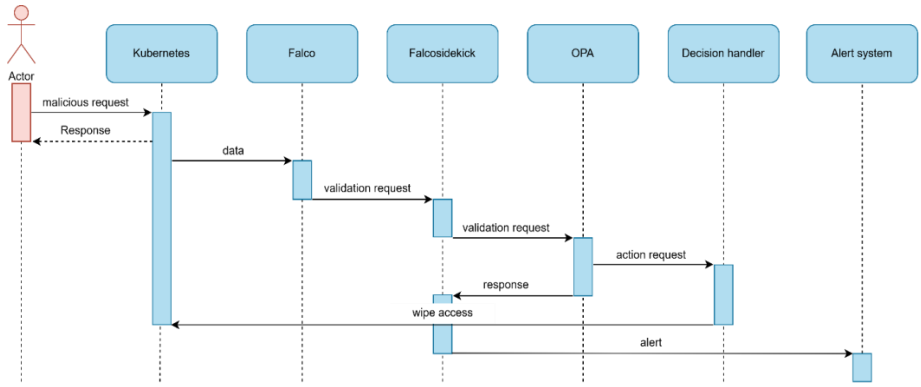

Рисунок 1. Діаграма послідовності реакції на шкідливий запит

Усе починається з того, що в кластері Kubernetes відбувається певна активність, яка потрапляє під моніторинг Falco. Це може бути системний виклик, що порушує встановлені правила, наприклад, exec у контейнері. Falco, виявивши подію, генерує повідомлення у форматі JSON, яке пересилається до Falcosidekick.

Falcosidekick приймає подію і передає її до зовнішнього API-сервісу – компоненту Open Policy Agent. Цей компонент виконує політичний аналіз, використовуючи визначені Rego-політики, й оцінює, чи є подія порушенням безпекових норм. У разі виявлення порушення OPA повертає відповідь, яка містить рішення про неприйнятність або небезпечність дії.

Далі рішення OPA інтерпретується decision handler – окремим компонентом, який відповідає за реалізацію технічної реакції. Якщо подія визнана загрозливою, decision handler ініціює дію, наприклад, видалення доступів або ізоляцію ресурсу, що видно на діаграмі як "wipe access".

Після завершення реакції система передає фінальний результат або повідомлення у зовнішню підсистему сповіщення, наприклад, систему алертів, SIEM чи канал сповіщень команди безпеки. Таким чином, команда або оператор отримує інформацію про те, що відбулася загрозлива активність, яку було проаналізовано й оброблено відповідно до заданої політики.

Цей підхід демонструє повний життєвий цикл події: від виявлення у середовищі до автоматичного рішення та сповіщення, забезпечуючи при цьому чітке розділення обов’язків між компонентами та мінімізацію ризиків через обмеження прав доступу кожної частини системи.

Представлена схема роботи системи сформована на основі принципу мінімальних привілеїв, що є ключовим підходом у побудові безпечної архітектури в Kubernetes.

Головна ідея полягає в тому, щоб кожен компонент системи виконував лише ті функції, які йому безпосередньо призначені, і мав лише той рівень доступу, який необхідний для реалізації цих функцій – не більше.

Falco виконує роль пасивного сенсора, який здійснює моніторинг поведінки контейнерів на рівні ядра. Для цього йому не потрібен доступ до Kubernetes API, а лише доступ до системних подій та логів. Такий підхід дозволяє запускати Falco з обмеженими привілеями або в ізольованому середовищі, зменшуючи ризик його компрометації.

Falcosidekick, який приймає події від Falco, також не має доступу до управління кластером. Його єдина роль – пересилати події у зовнішні сервіси. Це дозволяє використовувати його як логічний маршрутизатор без надмірних прав.

Основну логіку ухвалення рішення виконує Open Policy Agent. Він отримує подію у вигляді запиту і, на основі наперед визначених Rego-політик, повертає відповідь. Важливо, що OPA сам по собі не має і не потребує жодного доступу до Kubernetes API. Він не виконує жодних змін у кластері, а лише аналізує та формулює рішення.

Реальне втручання в роботу кластера виконується лише одним компонентом – Decision handler. Саме він, отримавши рішення від OPA, може ініціювати конкретні дії, наприклад, видалення ресурсу або сповіщення. І лише цей компонент має необхідні права на запис до Kubernetes API. Така централізація виконання дозволяє точно контролювати зону ризику та застосовувати додатковий аудит або моніторинг саме до цього сервісу.

Насамперед така модель є реактивною, а не превентивною. Це означає, що система не здатна запобігти події до її виникнення – вона лише реагує на те, що вже відбулося [18]. Наприклад, якщо користувач всередині контейнера виконає підозрілу команду, наприклад curl на зовнішню адресу, Falco її зафіксує, але це трапиться після того, як команда вже була виконана. Відповідно, навіть якщо OPA визнає цю дію порушенням політики, шкода може бути вже завдана, і рішення, що надходить після події, не здатне її скасувати.

Іншим обмеженням є затримка між подією та дією. Оскільки ланцюжок проходить через кілька систем – Falco, Falcosidekick, decision handler, OPA – кожна з них додає невелику затримку. Це може становити проблему у сценаріях, де важлива швидкість реагування, наприклад, при виявленні експлуатації вразливостей у реальному часі. Час затримки залежить від обчислювальної потужності системи, а також кількості подій, що необхідно обробити [19].

Ще один ризик – відсутність гарантії доставки та обробки подій. У разі збою Falcosidekick, неправильного форматування події або перевантаження decision handler, окремі події можуть бути втрачені або не оброблені. Це особливо критично в системах, які повинні мати повний і точний аудит активності.

Крім того, система залежить від зовнішніх інтеграцій, які мають бути правильно налаштовані й доступні: наприклад, якщо OPA тимчасово недоступний, decision handler не зможе отримати рішення і може не виконати необхідну дію або навпаки – виконати її без перевірки.

І, нарешті, важливо розуміти, що такий підхід не захищає від помилок конфігурації на рівні Kubernetes API. Якщо, наприклад, шкідливий pod було створено з некоректними правами доступу, але він ще нічого не зробив, така система його не зафіксує. Для таких випадків потрібно використовувати інші інструменти – наприклад, Gatekeeper як admission controller, який здатен блокувати дії до їх виконання [20].

ВИСНОВКИ ТА ПЕРСПЕКТИВИ ПОДАЛЬШИХ ДОСЛІДЖЕНЬ

У підсумку проведене дослідження демонструє, що поєднання Falco та Open формує багаторівневу модель безпеки, де рантайм-виявлення загроз доповнюється гнучким механізмом політик. Варіант прямого надсилання подій з Falco через Falcosidekick до HTTP-інтерфейсу OPA забезпечує оперативну реакцію і мінімальну затримку між фіксацією інциденту та ухваленням рішення, що робить його доцільним для кластерів із підвищеними вимогами до часу відгуку.

Використання OPA у ролі admission controller посилює превентивний захист, оскільки блокує небажані об’єкти ще до їхнього розгортання, однак потребує постійного коригування Rego-політик на основі знань, що їх надає Falco. Сценарій із проміжним сервісом узгоджує переваги обох підходів: він дає можливість виконувати складну бізнес-логіку, вмикає обмежену ізоляцію підозрілих робочих навантажень і водночас не перевантажує OPA надмірною кількістю подій.

Усі три схеми підтверджують, що інтеграція сенсорних та політико-орієнтованих компонентів суттєво підвищує рівень захисту контейнеризованих середовищ, сприяє реалізації принципів Zero Trust і створює передумови для самовідновної інфраструктури з автоматизованим контролем відповідності.

Тому подальшим розвитком проведеного дослідження буде їх інтенсивна практична реалізація та імплементація для підвищення рівня захисту інформації в контейнеризованих середовищах.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Chen, G., Gao, X., Xu, W., Liu, J., & Xu, X. (2025). MAC-UAE: Multi-level access control based on updateable attribute encryption of secure data in mobile cloud center. *Mobile Networks and Applications*. https://doi.org/10.1007/s11036-025-02451-y
2. Chakraborty, S., & Sandhu, R. (2021). Formal analysis of ReBAC policy mining feasibility. In *Proceedings of the 11th ACM Conference on Data and Application Security and Privacy (CODASPY ’21)* (pp. 197–207). ACM. https://doi.org/10.1145/3422337.3447828
3. Cheng, Y., Bijon, K., & Sandhu, R. (2016). Extended ReBAC administrative models with cascading revocation and provenance support. In *Proceedings of the 21st ACM Symposium on Access Control Models and Technologies (SACMAT 2016)*. ACM. https://doi.org/10.1145/2914642.2914655
4. Vijayaraghavan, S. K. J. (2025). Policy as code: A paradigm shift in infrastructure security and governance. *World Journal of Advanced Research and Reviews, 26*(1), 3399–3405. https://doi.org/10.30574/wjarr.2025.26.1.1441
5. Yaqub, N., Ullah, S., Jalil, A., & Khan, I. (2025). Blockchain-enabled policy-based access control mechanism to restrict unauthorized access to electronic health records. *PeerJ Computer Science, 11*, e2647. https://doi.org/10.7717/peerj-cs.2647
6. Merlec, M. M., & In, H. P. (2024). SC-CAAC: A smart contract-based context-aware access control scheme for blockchain-enabled IoT systems. *IEEE Internet of Things Journal*. https://doi.org/10.1109/JIOT.2024.3371504
7. Lee, B., Almutairi, A., & Barka, E. (2017). Situational awareness-based risk-adaptable access control in enterprise networks. In *Proceedings of the 2nd International Conference on Internet of Things, Big Data and Security*. https://doi.org/10.5220/0006363404000405
8. Kermabon-Bobinnec, H., et al. (2022). ProSPEC: Proactive security policy enforcement for containers. In *Proceedings of the 12th ACM Conference on Data and Application Security and Privacy (CODASPY ’22)*. ACM. https://doi.org/10.1145/3508398.3511515
9. Vaarandi, R., et al. (2025). A systematic literature review of cyber security monitoring in maritime. *IEEE Access*. https://doi.org/10.1109/ACCESS.2025.3567385
10. Ebute, M. (2025). Continuous monitoring and assessment mechanisms in cybersecurity: Best practices for sustained protection of critical assets. SSRN. https://ssrn.com/abstract=4912624
11. Dempsey, K., et al. (2021). *ISCMA: An information security continuous monitoring program assessment* (NIST IR 8212). National Institute of Standards and Technology. https://doi.org/10.6028/NIST.IR.8212

12. Pitkar, H. (2025). Enhancing Kubernetes security with AI: Anomaly detection for cloud-based workloads. *International Scientific Journal of Engineering and Management, 4*(4), 1–9. https://doi.org/10.55041/isjem02746
13. Open Policy Agent. (n.d.). *Overview & architecture*. https://www.openpolicyagent.org/docs/latest/kubernetes-introduction/
14. Falco Project. (n.d.). *Falco documentation*. https://falco.org/docs/
15. Ali, A., et al. (2024). Implementation of new security features in CMSWEB Kubernetes cluster at CERN. *EPJ Web of Conferences*, *295*, 07026. https://doi.org/10.1051/epjconf/202429507026
16. Open Policy Agent. (n.d.). *REST API*. https://www.openpolicyagent.org/docs/latest/rest-api/
17. Falco Project. (n.d.). *Alerts forwarding*. https://falco.org/docs/concepts/outputs/forwarding/
18. Kermabon-Bobinnec, H., et al. (2024). PerfSPEC: Performance profiling-based proactive security policy enforcement for containers. *IEEE Transactions on Dependable and Secure Computing*, 1–18. https://doi.org/10.1109/TDSC.2024.3420712
19. Falco Project. (n.d.). *Falco performance testing*. https://falco.org/blog/falco-performance-testing/
20. Zhang, R., et al. (2019). OPA Gatekeeper: Policy and governance for Kubernetes. *Kubernetes Blog*. https://kubernetes.io/blog/2019/08/06/opa-gatekeeper-policy-and-governance-for-kubernetes/

КІБЕРБЕЗПЕКА: освіта, наука, техніка
CYBERSECURITY: EDUCATION, SCIENCE, TECHNIQUE
№ 4 (32), 2026
ISSN 2663 – 4023

Dmytro Darienko
Postgraduate student of the Department of Information Protection
National University "Lviv Polytechnic", Lviv, Ukraine
ORCID: 0009-0001-7080-531X
dmytro.h.dariienko@lpnu.ua

Nazariy Kohut
Postgraduate student of the Department of Information Protection
National University "Lviv Polytechnic", Lviv, Ukraine
ORCID: 0009-0000-4466-7868
nazarii.m.kohut@lpnu.ua

EXTENSION OF OPEN POLICY AGENT FUNCTIONS USING FALCO

Abstract. The paper considers approaches to integrating the real-time threat detection system Falco with the Open Policy Agent (OPA) policy mechanism in the Kubernetes environment. Three main interaction models are analyzed: direct event sending via Falcosidekick to the OPA HTTP API, using OPA as an admission controller to block dangerous configurations at the resource creation stage, and using an intermediate service as a connecting link between Falco and OPA with the ability to process complex events. The study demonstrates that the combination of sensor control mechanisms and policy-oriented decision-making allows for a higher level of security, automated response, and compliance with Zero Trust principles. The combination of Falco and Open Policy Agent forms a multi-layered security model, where runtime threat detection is complemented by a flexible policy mechanism. The option of direct event sending from Falco via Falcosidekick to the OPA HTTP interface provides a prompt response and minimal delay between incident detection and decision-making, which makes it appropriate for clusters with increased response time requirements. Using OPA as an admission controller enhances proactive protection by blocking unwanted objects before they are deployed, but requires constant adjustment of Rego policies based on the knowledge provided by Falco. The scenario with an intermediate service combines the advantages of both approaches: it allows for complex business logic, enables limited isolation of suspicious workloads, and at the same time does not overload OPA with an excessive number of events. All three schemes confirm that the integration of sensor and policy-oriented components significantly increases the level of protection of containerized environments, contributes to the implementation of Zero Trust principles, and creates the prerequisites for a self-healing infrastructure with automated compliance control.

Keywords: kubernetes; dynamic access; contextual security; OPA; Falco; information security, information protection.

REFERENCES (TRANSLATED AND TRANSLITERATED)

1. Chen, G., Gao, X., Xu, W., Liu, J., & Xu, X. (2025). MAC-UAE: Multi-level access control based on updateable attribute encryption of secure data in mobile cloud center. *Mobile Networks and Applications*. https://doi.org/10.1007/s11036-025-02451-y
2. Chakraborty, S., & Sandhu, R. (2021). Formal analysis of ReBAC policy mining feasibility. In *Proceedings of the 11th ACM Conference on Data and Application Security and Privacy (CODASPY '21)* (pp. 197–207). ACM. https://doi.org/10.1145/3422337.3447828
3. Cheng, Y., Bijon, K., & Sandhu, R. (2016). Extended ReBAC administrative models with cascading revocation and provenance support. In *Proceedings of the 21st ACM Symposium on Access Control Models and Technologies (SACMAT 2016)*. ACM. https://doi.org/10.1145/2914642.2914655
4. Vijayaraghavan, S. K. J. (2025). Policy as code: A paradigm shift in infrastructure security and governance. *World Journal of Advanced Research and Reviews, 26*(1), 3399–3405. https://doi.org/10.30574/wjarr.2025.26.1.1441
5. Yaqub, N., Ullah, S., Jalil, A., & Khan, I. (2025). Blockchain-enabled policy-based access control mechanism to restrict unauthorized access to electronic health records. *PeerJ Computer Science, 11*, e2647. https://doi.org/10.7717/peerj-cs.2647

6. Merlec, M. M., & In, H. P. (2024). SC-CAAC: A smart contract-based context-aware access control scheme for blockchain-enabled IoT systems. *IEEE Internet of Things Journal*. https://doi.org/10.1109/JIOT.2024.3371504
7. Lee, B., Almutairi, A., & Barka, E. (2017). Situational awareness-based risk-adaptable access control in enterprise networks. In *Proceedings of the 2nd International Conference on Internet of Things, Big Data and Security*. https://doi.org/10.5220/0006363404000405
8. Kermabon-Bobinnec, H., et al. (2022). ProSPEC: Proactive security policy enforcement for containers. In *Proceedings of the 12th ACM Conference on Data and Application Security and Privacy (CODASPY '22)*. ACM. https://doi.org/10.1145/3508398.3511515
9. Vaarandi, R., et al. (2025). A systematic literature review of cyber security monitoring in maritime. *IEEE Access*. https://doi.org/10.1109/ACCESS.2025.3567385
10. Ebute, M. (2025). Continuous monitoring and assessment mechanisms in cybersecurity: Best practices for sustained protection of critical assets. SSRN. https://ssrn.com/abstract=4912624
11. Dempsey, K., et al. (2021). *ISCMA: An information security continuous monitoring program assessment* (NIST IR 8212). National Institute of Standards and Technology. https://doi.org/10.6028/NIST.IR.8212
12. Pitkar, H. (2025). Enhancing Kubernetes security with AI: Anomaly detection for cloud-based workloads. *International Scientific Journal of Engineering and Management, 4*(4), 1–9. https://doi.org/10.55041/isjem02746
13. Open Policy Agent. (n.d.). *Overview & architecture*. https://www.openpolicyagent.org/docs/latest/kubernetes-introduction/
14. Falco Project. (n.d.). *Falco documentation*. https://falco.org/docs/
15. Ali, A., et al. (2024). Implementation of new security features in CMSWEB Kubernetes cluster at CERN. *EPJ Web of Conferences, 295*, 07026. https://doi.org/10.1051/epjconf/202429507026
16. Open Policy Agent. (n.d.). *REST API*. https://www.openpolicyagent.org/docs/latest/rest-api/
17. Falco Project. (n.d.). *Alerts forwarding*. https://falco.org/docs/concepts/outputs/forwarding/
18. Kermabon-Bobinnec, H., et al. (2024). PerfSPEC: Performance profiling-based proactive security policy enforcement for containers. *IEEE Transactions on Dependable and Secure Computing*, 1–18. https://doi.org/10.1109/TDSC.2024.3420712
19. Falco Project. (n.d.). *Falco performance testing*. https://falco.org/blog/falco-performance-testing/
20. Zhang, R., et al. (2019). OPA Gatekeeper: Policy and governance for Kubernetes. *Kubernetes Blog*. https://kubernetes.io/blog/2019/08/06/opa-gatekeeper-policy-and-governance-for-kubernetes/

Отримано редакцією журналу / Received: 06.01.26
Прорецензовано / Revised: 21.02.26
Схвалено до друку / Accepted: 26.03.26