КІБЕРБЕЗПЕКА: освіта, наука, техніка
CYBERSECURITY: EDUCATION, SCIENCE, TECHNIQUE
№ 1 (29), 2025
ISSN 2663 - 4023
DOI 10.28925/2663-4023.2025.29.820
УДК 004.056

Скіп Андрій Володимирович
аспірант кафедри захисту інформації
Вінницький національний технічний університет, Вінниця, Україна
ORCID ID: 0009-0009-3334-4145
phd@askip.me

Баришев Юрій Володимирович
кандидат технічних наук, доцент кафедри захисту інформації
Вінницький національний технічний університет, Вінниця, Україна
ORCID ID: 0000-0001-8324-8869
yuriy.baryshev@vntu.edu.ua

АНАЛІЗ БЕЗПЕКИ ЛАНЦЮГІВ ПОСТАЧАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Анотація У статті досліджено актуальну проблему забезпечення безпеки програмних інтерфейсів (API) у хмарних середовищах в контексті захисту ланцюгів постачання програмного забезпечення. Враховуючи поширення мікросервісних архітектур та відкритих REST/GraphQL API, питання захисту API від атак набуло критичної важливості. Наведено формалізацію ланцюга постачання програмного забезпечення. На основі цієї моделі проведено аналіз потенційних векторів атак на кожному з етапів програмного ланцюга постачання — від компрометації репозиторіїв, залежностей і CI/CD-серверів до сховищ артефактів і середовищ розгортання. Особливу увагу приділено аналізу двох ключових механізмів захисту — Web Application Firewall (WAF) і API Gateway. У статті представлено порівняльний аналіз їх функціональності, підходів до обробки запитів, рівнів безпеки, методі в журналювання та інтеграції у хмарну інфраструктуру. Визначено, що WAF ідентифікує та блокує атаки на HTTP-рівні, зокрема SQL-ін'єкції, XSS, CSRF, на основі сигнатур або поведінкових правил. Натомість API Gateway виконує роль шлюзу, керує маршрутизацією, автентифікацією та авторизацією запитів, впроваджує політики доступу та надає контроль над API-викликами. Результати дослідження демонструють, що WAF та API Gateway не є взаємозамінними, а повинні застосовуватись як взаємодоповнюючі компоненти. Запропоновано комбіновану модель захисту API в хмарі, де WAF забезпечує периметральний контроль, а API Gateway — логічний контроль доступу. У статті також наведено методи оцінювання ефективності цих механізмів: емпіричні випробування в тестових середовищах, порівняльні бенчмарки з різними конфігураціями, а також аналітичне моделювання загроз. У результаті сформульовано низку нерозв'язаних задач і напрямів для подальших наукових досліджень у сфері захисту API та CI/CD у хмарних середовищах.

Ключові слова: кібербезпека; захист API; CI/CD; ланцюг постачання ПЗ; аналіз загроз; захист web-застосунків.

ВСТУП

З-поміж атак на застосунки вирізняються атаки на ланцюги постачання (supply chains) цих застосунків до користувачів. Особливість протидії цим атакам полягає у різнорідності векторів атак. Це можуть бути атаки на репозиторії з вихідним кодом, налаштування автоматизованих процесів постачання коду (pipeline), тестові та виробничі сервери, а також програмні інтерфейси застосунків (API). Останні стали фундаментальними для інтеграції та взаємодії між сервісами. Водночас, попри численні переваги, API створюють серйозні виклики для кібербезпеки [1]. Дослідження [2] показують, що API вже стали пріоритетною мішенню зловмисників. Проблема особливо актуальна для хмарних середовищ, де компанії масово розгортають мікросервіси та веб-

застосунки, надаючи публічні REST і GraphQL API для зовнішніх та внутрішніх споживачів. Відкритість і масштабованість хмари збільшують кількість векторів атак, тому питання захисту таких API є критично важливим.

Серед ключових механізмів захисту веб-застосунків та API сьогодні найпоширенішими є Web Application Firewall (WAF) та API Gateway (шлюз API). WAF та шлюзи API часто розгортаються в хмарній інфраструктурі та покликані запобігати несанкціонованим запитам, атакам та іншим загрозам. Однак їх різнорідність обумовлює актуальне завдання порівняльного їх аналізу. Метою цієї роботи є удосконалення безпеки ланцюгів постачання програмного забезпечення шляхом порівняльного аналізу WAF та API Gateway у забезпеченні безпеки API в хмарних середовищах та визначення шляхів їх удосконалення.

Постановка проблеми. Попри активне впровадження WAF і API Gateway у хмарних архітектурах, відсутність чітких рекомендацій щодо їх узгодженого використання створює значні ризики. Часто ці засоби впроваджуються ізольовано, що ускладнює досягнення повноцінного рівня безпеки. Зокрема, WAF зазвичай не враховує контекст бізнес-логіки API, тоді як API Gateway не виконує глибокий аналіз запитів на наявність шкідливого вмісту. Це призводить до виникнення «сірих зон» у захисті, які можуть бути використані зловмисниками для компрометації ланцюгів постачання програмного забезпечення. Додатково, поява нових форматів API (наприклад, GraphQL [3], [4]) ще більше ускладнює задачу адаптації наявних засобів захисту. У зв'язку з цим постає потреба у цілісному аналізі взаємодії WAF та API Gateway з позиції забезпечення надійного захисту API в хмарному середовищі.

Аналіз останніх досліджень і публікацій. Проблематика захисту API у хмарних середовищах дедалі більше привертає увагу науковців і практиків, що підтверджується низкою сучасних досліджень [1] – [3]. Зокрема, у роботі [1] підкреслюється, що традиційні методи захисту API недостатньо ефективні в умовах гібридних і мультихмарних архітектур. Автор наголошує на необхідності впровадження принципів Zero Trust, централізованої ідентифікації, багатофакторної автентифікації, а також обов'язкової перевірки політик доступу для кожного мікросервісу. Це положення перегукується з основною ідеєю даної статті — необхідністю комбінованого використання WAF і API Gateway як взаємодоповнюючих засобів захисту.

Інше дослідження [2], проведене у Linköping University (Швеція), містить порівняльний аналіз функцій API Gateway і Application Gateway у контексті хмарних сервісів. Дослідники вказують на обмеженість традиційних WAF у виявленні сучасних атак, спрямованих на бізнес-логіку API, та наголошують на потребі динамічної фільтрації трафіку. Ці висновки підтримують у роботі й автори статті, що аналізується, зокрема в розділі про «сірі зони» — області, які залишаються незахищеними при ізольованому використанні лише одного з механізмів (WAF чи API Gateway).

У статті [3] розглядається API як «хребет цифрової трансформації» і наголошується, що будь-які вразливості на цьому рівні можуть вплинути на всю CI/CD-інфраструктуру. Автор детально описує потенційні ризики, пов'язані з публікацією відкритих REST/GraphQL API, та висуває ідею про застосування поведінкової аналітики до вхідного трафіку, що може бути інтегровано в сучасні API Gateway. Саме ця потреба у глибшому аналізі запитів — поза межами сигнатурного фільтрування — є однією з раніше невирішених частин проблеми, на яку прямо вказується в даній роботі (розділ «Постановка проблеми»).

Огляд літератури також свідчить про дефіцит досліджень, присвячених узгодженій роботі WAF і API Gateway в умовах реальних хмарних CI/CD-pipelines. Зазвичай

механізми аналізуються ізольовано або без урахування практичних аспектів впровадження в розподілених середовищах. Таким чином, нерозв'язаними залишаються такі частини загальної проблеми:

- відсутність формалізованих моделей взаємодії WAF та API Gateway з урахуванням специфіки мікросервісної архітектури та особливостей ланцюгів постачання ПЗ;
- недостатня видимість "тіньових API", які можуть бути недокументованими або самовільно розгорнутими, що ускладнює захист через API Gateway;
- відсутність адаптивних засобів аналізу бізнес-логіки запитів на рівні WAF, що не дозволяє своєчасно виявляти націлені атаки на API;
- потреба у динамічному моніторингу та аналізі загроз у реальному часі в контексті CI/CD.

У контексті цих викликів, подальші наукові дослідження мають бути спрямовані на інтеграцію механізмів глибокої інспекції, оркестрації політик доступу та спільного журналювання подій безпеки в єдину платформу захисту API.

Мета статті. Метою цієї роботи є удосконалення безпеки ланцюгів постачання шляхом порівняльного аналізу WAF та API Gateway у забезпеченні безпеки API в хмарних середовищах та визначення шляхів їх удосконалення. Для досягнення мети необхідно виконати такі завдання:

- формалізувати ланцюг постачання програмних засобів;
- визначити функціональне призначення WAF та API Gateway, а також визначити їх відмінності;
- виявити основні переваги та виклики використання цих технологій для захисту API;
- ідентифікувати нерозв'язані задачі захисту ланцюгів постачання при використанні цих засобів;
- формалізувати завдання щодо подальших наукових досліджень в напрямі захисту ланцюгів постачання програмних засобів.

ТЕОРЕТИЧНІ ОСНОВИ ДОСЛІДЖЕННЯ

Ланцюг постачання програмного забезпечення починається з розробників, які створюють сирцевий код. Результати їх роботи в подальшому інтегруються за допомогою репозиторіїв для систем керування версіями, як-то GitHub, SVN або Bitbucket. З репозиторію цей сирцевий код за допомогою автоматизованих процесів забезпечення неперервності бізнес-процесів розробки компілюється, тестується та збирається відповідно до технології розробки. Також на цьому етапі до програмного продукту включається код програмних бібліотек, який використовували розробники. Оскільки за результатами компілювання через наявність дефектів в програмному коду тестування або, навіть, компілювання можуть бути невдалими, автоматизовано формуються артефакти виконання цих дій, які містять результати їх перебігу. Результатом збірки є артефакт який завантажується в сховище, після чого артефакт розгортається в середовищі виконання (CD-pipeline) [5] – [8]. На рис. 1 наведено спрощену схему ланцюга постачання програмного забезпечення, яка включає основні етапи розроблення програм [9] – [11].

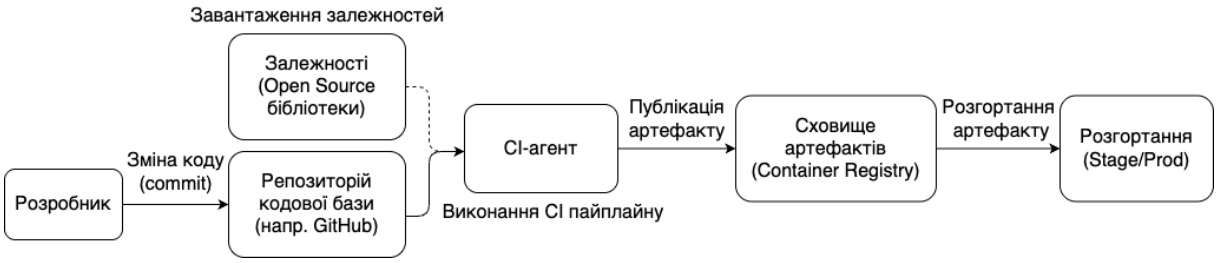

Рис. 1. Спрощена структура програмного ланцюга постачання з компонентами CI/CD

Як видно із схеми, кожен етап ланцюга постачання є окремим елементом, що потребує захисту від стороннього втручання. Дуже часто кожен етап CI/CD процесу виконується в окремому середовищі (віртуальна машина, датацентр, хмарна платформа). Сегментація компонентів за їх призначенням дозволяє ізолювати потенційну вразливість в одному місці, але кількість таких елементів і необхідність побудови мережі між ними, навпаки — збільшує площу атаки. Запропонована нижче схема (рис. 2) описує механізми захисту ланцюга постачання для кожного елементу.

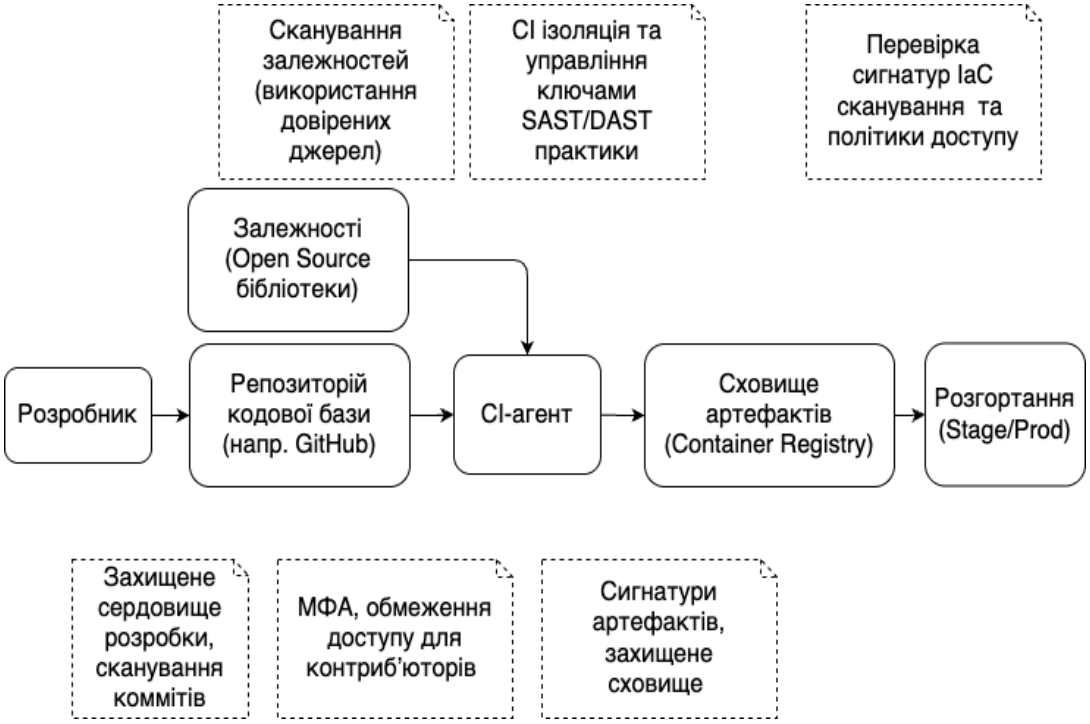

Рис. 2. Механізми безпеки на кожному етапі ланцюга постачання ПЗ.

На рівні розробника — безпечне середовище (IDE), code-review [6] – [8] — аналіз коду на відповідність до стандартів безпеки. На рівні репозиторію - контроль доступу, дозволяє змінювати код лише авторизованим користувачам і лише після аналізу наявність паролів, токенів, сертифікатів тощо, у відкритому вигляді. Ізолювання агентів збірки, дозволяє отримати повністю закрите середовище, де відбувається перевірка залежностей ще під час збірки і перед публікацією артефакту до сховища. Така процедура дозволяє впевнитись що лише артефакти, що відповідають критеріям безпеки, можуть бути присутніми в сховищі і використовуватись в процесі розгортання.

Попри наявність процесів захисту на ланцюг постачання можуть бути здійсненими різноманітні атаки. На рис. 3 наведено результати їх аналізу.

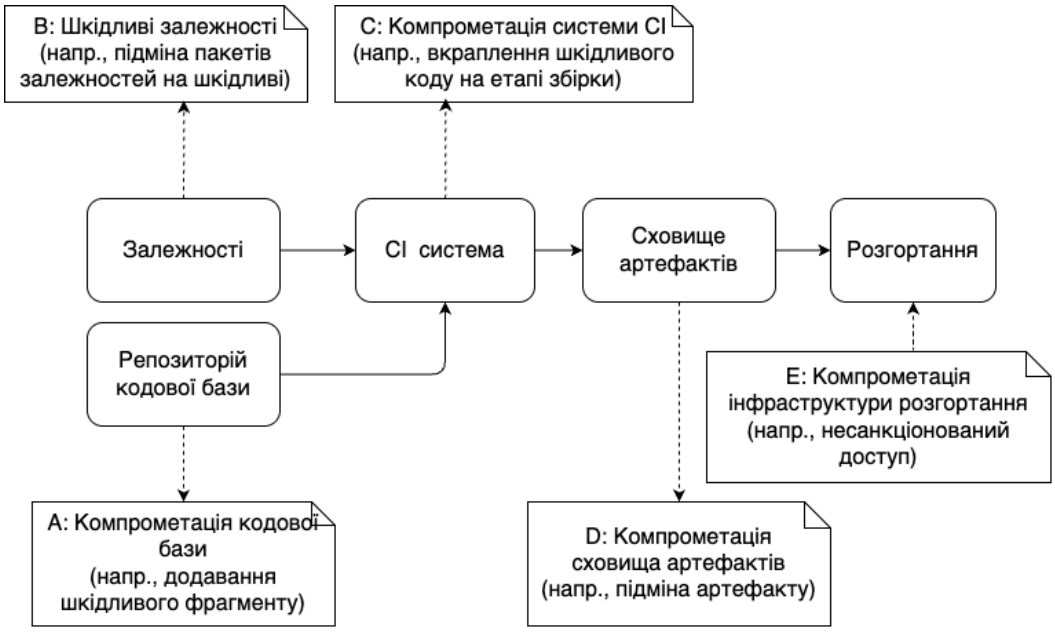

Рис. 3. Потенційні вектори атак на програмний ланцюг постачання

Як видно з рис. 3 зловмисник може:

- (A) скомпрометувати репозиторій вихідного коду (через вкрадений обліковий запис розробника або уразливість платформи SCM) і внести шкідливий код;
- (B) змінити залежність — опублікувати чи змінити сторонню бібліотеку, що використовується проектом, додавши в неї бекдор;
- (C) атакувати CI/CD сервер або pipeline конфігурацію, щоб впровадити свій код у процес збірки;
- (D) отримати несанкціонований доступ до сховища артефактів і підмінити там пакет/образ на скомпрометований;
- (E) скомпрометувати середовище розгортання (наприклад, Kubernetes-кластер) чи секрети пов’язані із запуском програмних засобів, щоб вплинути на результат — програмний засіб, який буде поставлено [7], [8].

РЕЗУЛЬТАТИ ДОСЛІДЖЕННЯ

Як випливає з виконаного аналізу, актуальним питанням захисту ланцюгів постачання є дослідження засобів їх захисту. Для випадку веб-застосунків мікросервісної архітектури [10] такими засобами є WAF та API Gateway.

Web Application Firewall (WAF) — це міжмережевий екран рівня веб-застосунків, що фільтрує та контролює HTTP(S)-трафік між клієнтами та сервером. WAF працює як проксі-сервер, аналізуючи вміст запитів та відповідей і порівнюючи їх з набором правил безпеки [12]. Основне призначення WAF — виявляти і блокувати шкідливі запити, зокрема атаки типу SQL-ін’єкцій, міжсайтового скриптингу (XSS), підробки міжсайтових запитів (CSRF) та інші поширені веб-вразливості [3]. Таким чином, WAF

захищає веб-застосунки від відомих шаблонів атак, використовуючи сигнатури та евристики (наприклад, правила на основі OWASP Top 10 [13]). WAF були розроблені для відповідності вимогам безпеки (наприклад PCI DSS 6.6) і існують набагато довше, ніж сучасні API-сервіси [1]. Сьогодні WAF доступні як у вигляді апаратних/програмних шлюзів, так і у вигляді хмарних сервісів (наприклад, AWS WAF, Azure WAF, Cloudflare WAF тощо) [12].

API Gateway — це шлюз для програмних інтерфейсів, який виступає єдиною точкою входу для зовнішніх запитів до набору внутрішніх сервісів (зазвичай організованих як мікросервіси або функції). Основні функції API Gateway — маршрутизація викликів до відповідних сервісів, агрегування та оркестрування запитів, а також керування доступом і протоколами [14]. На відміну від WAF, що зосереджений саме на фільтрації шкідливого трафіку, API Gateway більшою мірою відповідає за адміністрування API: автентифікацію викликів (наприклад, перевірка токенів або ключів API), авторизацію доступу до ресурсів, застосування політик лімітування запитів (rate limiting) і квот, забезпечення балансування навантаження та перетворення даних (наприклад, формату запитів/відповідей JSON в XML тощо) [5]. Таким чином, API Gateway діє як проксі-сервер на рівні прикладної логіки, що поєднує можливості зворотного проксі/балансувальника навантаження з базовими засобами безпеки – такими як автентифікація та журналювання [1]. Відзначимо, що шлюзи API набули популярності саме з ростом використання веб-API.

Хоча обидва рішення працюють на межі між клієнтом і сервером, їх фокус та підходи різняться за такими параметрами:

- рівнем захисту;
- методом реагування на загрозу;
- зоною видимості та логування;
- функціональністю.

WAF діє на рівні HTTP-трафіку, аналізуючи вміст кожного запиту щодо наявності шкідливих шаблонів або аномалій. Натомість API Gateway працює на рівні бізнес-логіки API, виконуючи контроль доступу та маршрутизацію, але зазвичай не проводить детальний аналіз вмісту щодо атак. Інакше кажучи, WAF захищає від відомих веб-експлойтів, а шлюз API захищає точки входу через механізми автентифікації й авторизації.

Відповідно до методу реагування WAF має набір правил (сигнатурних або поведінкових), які автоматично блокують або фільтрують шкідливі запити до того, як вони досягнуть застосунка. На противагу, API Gateway — за замовчуванням пропускає запити до сервісів, якщо ті відповідають визначеній конфігурації (шляхи, методи) і містять актуальні облікові дані. Він може відхиляти запити, що порушують встановлені політики (відсутність авторизації, перевищення лімітів тощо), але не обов’язково розпізнає ін’єкції чи інші атаки, якщо вони виглядають як формально коректні запити [7].

За критерієм видимості та охоплення API Gateway оперує зареєстрованими у ньому маршрутами API. Тобто він ефективний для документованих та відомих кінцевих точок API, але може не бачити «тіньових API» (невідомих або несанкціоновано розгорнутих сервісів) [3], [15]. WAF натомість може захищати будь-який HTTP(S) трафік, навіть, якщо якийсь API не був явно доданий в конфігурацію, — проте WAF не «розуміє» контексту кожного окремого API, тлумачачи трафік більш узагальнено. З іншого боку, WAF надає розширену видимість у трафік веб-застосунку загалом (збір метрик, логів на рівні запитів тощо), тоді як шлюз API здебільшого логує звернення в розрізі визначених сервісів і може надавати аналітику використання API.

За функціональністю шлюзи API, особливо хмарні рішення, часто інтегровані з іншими сервісами безпеки та розвитку: підтримують OAuth 2.0/OIDC для автентифікації, можуть перевіряти підписи JWT токенів, працювати з сертифікатами клієнтів, забезпечувати CORS-політики тощо. WAF же зосереджений саме на веб-трафіку, протидії ботам і веб-вразливостями: сучасні WAF можуть включати модулі для захисту від ботів, запобігання DDoS на рівні застосунку, виявлення аномалій протоколу HTTP, сканування вмісту щодо вразливостей тощо [8]. Таким чином, API Gateway більше відповідає за управління та довіреність трафіку, а WAF — за перевірку його безпечності.

Отримані результати аналізу скомпоновано в табл. 1.

Таблиця 1

Відмінності між WAF та API Gateway

Критерій	WAF (Web Application Firewall)	API Gateway (шлюз API)
Рівень захисту	– Спрямований на атаки на веб-рівні: атаки з OWASP Top 10, спільне відвідування веб-сайтів; – Виявляє відомі вразливості за сигнатурними правилами; – Не враховує бізнес-логіку API.	– Захист доступу до API: автентифікація, ліміти, CORS; – Не орієнтований на пошук ін'єкцій чи аномалій даних.
Метод реагування	– Блокування або фільтрація HTTP-запитів на основі правил; – Може повністю блокувати трафік при спрацьовування правила (HTTP 403/406) і логувати атаки; – Правила оновлюються періодично.	– Управляє маршрутизацією; – Відхиляє неавторизовані виклики (HTTP 401/403); – Реагує на основі налаштованої політики а не аналізу вмісту.
Функціональність	– Аналіз HTTP(S), розбір заголовків, URI, тіла запиту; – Часто містить модулі анти-DDoS, анти-бот, геофільтрацію; – Не «розуміє» API-схему чи формат даних; – Оперує описаними схемами запитів та regexp .	– Орієнтований на управління API: маршрутизація до сервісів, трансформація запитів/відповідей; – Моніторинг продуктивності; – Інтегрується у архітектуру застосунку як компонент платформи; – Підтримує можливості автентифікації.
Видимість та логування	– Журнал подій безпеки: які атаки відбулись, з яких IP-адрес, які правила спрацьовували; – Не відстежує логіку запитів; – Не має інформації про внутрішні ідентифікатори користувачів, лише про самі запити.	– Журнали використання API: які методи викликались, час відгуку, токен користувача; – Відстежує потік легітимних запитів; – Не аналізує тіло запиту щодо атаки.
Розгортання	– Як правило на периметрі (cloud service або перед застосунком); – Одна інсталяція WAF може обслуговувати декілька сервісів (платформ) одразу; – Вимагає налаштування правил під конкретний застосунок, щоб зменшити хибні спрацювання (false-positive).	– Вбудовується як точка входу для користувацького трафіку; – Розгортається для кожного набору API; – Вимагає опис усіх API-методів, схем, авторизації. Інтегрований у процес розробки.

Аналіз табл. 1 показує, що WAF і API Gateway слід розглядати не як взаємовиключні, а як компоненти що доповнюють один одного в архітектурі безпеки.

ВИСНОВКИ ТА ПЕРСПЕКТИВИ ПОДАЛЬШИХ ДОСЛІДЖЕНЬ

Аналіз ланцюгів постачання веб-застосунків показав актуальність їх захисту та дозволив виявити потенційні вектори атак. Для зменшення впливу цих атак на програмні засоби, виконано порівняльний аналіз засобів захисту цих ланцюгів постачання.

У результаті проведеного дослідження визначено, що забезпечення безпеки API у хмарних середовищах є ключовим елементом захисту ланцюгів постачання програмного забезпечення. Аналіз функціональності та архітектурного застосування двох основних механізмів — Web Application Firewall (WAF) і API Gateway — показав, що ці інструменти мають комплементарні властивості й не можуть розглядатись як взаємозамінні. WAF ефективно протидіє класичним атакам на веб-рівні (SQLi, XSS, CSRF), тоді як API Gateway забезпечує логічне керування доступом до API, автентифікацію, авторизацію, облік викликів та інтеграцію з CI/CD-процесами.

Саме тому доцільно впроваджувати комбіновану модель використання WAF і API Gateway, яка дозволяє досягти більш високого рівня безпеки шляхом об’єднання фільтрування трафіку та контролю логіки викликів API.

Разом із тим, у межах цього дослідження залишаються відкритими такі проблеми, які потребують подальшого вивчення: відсутність формалізованої моделі узгодженої взаємодії між WAF та API Gateway з урахуванням специфіки мікросервісних архітектур, CI/CD-конвеєрів і динамічної конфігурації сервісів у хмарі.

Обмеженість традиційних WAF у контексті бізнес-логіки запитів, що ускладнює виявлення цілеспрямованих атак на критичні функції API; Нерозв'язана проблема захисту «тіньових» API (недокументованих або несанкціоновано розгорнутих), які часто не покриваються централізованими шлюзами; Відсутність єдиної системи журналювання безпекових подій між WAF, API Gateway та іншими компонентами безпеки (SIEM, EDR), що ускладнює кореляцію інцидентів;

Брак ефективних методів оцінювання впливу конфігурацій WAF і API Gateway на продуктивність та безпеку в реальних CI/CD-сценаріях; Необхідність в адаптивних механізмах реагування на нові формати API, зокрема GraphQL, які потребують специфічних правил фільтрації і моніторингу.

Подальші дослідження мають бути спрямовані на побудову формалізованих моделей інтеграції засобів захисту API, розробку методів поведінкової аналітики на рівні запитів та створення систем адаптивного захисту в умовах змінного хмарного середовища.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Türetken B. Enhancing security with cloud-based API management : Dissertation. Stockholm, 2024. 83 p. URL: https://www.diva-portal.org/smash/get/diva2:1903654/FULLTEXT01.pdf (date of access: 07.05.2025).
2. Dasher J. Why do I need API security if I have a WAF and API gateway?. cequence.ai. URL: https://www.cequence.ai/blog/api-security/why-do-i-need-api-security-if-i-have-a-waf-and-api-gateway (date of access: 07.05.2025).
3. McNaught B. API security best practices. *ISC2*. URL: https://www.isc2.org/Insights/2023/08/api-security-best-practices-in-the-hybrid-multi-cloud-digital-world#:~:text=were%20four%20key%20approaches:%20Web,going%20to%20be%20the%20converse” (date of access: 07.05.2025).
4. Cosgrove J., Andreev I. A. Protecting GraphQL APIs from Malicious Queries. The Cloudflare Blog. URL: https://blog.cloudflare.com/protecting-graphql-apis-from-malicious-queries/ (date of access: 07.05.2025).
5. What is a CI/CD pipeline?. Red Hat - We make open source technologies for the enterprise. URL: https://www.redhat.com/en/topics/devops/what-cicd-pipeline (date of access: 07.05.2025).

6. GitLab. What is a code review?. *The most-comprehensive AI-powered DevSecOps platform | GitLab*. URL: https://about.gitlab.com/topics/version-control/what-is-code-review/ (date of access: 07.05.2025).
7. Ханас М. Л., Барановський О. М. Сучасний стан безпеки Kubernetes: Використання існуючих інструментів та підходів для тестування на проникнення. XXI Всеукраїнська науково-практична конференція студентів, аспірантів та молодих вчених «Теоретичні і прикладні проблеми фізики, математики та інформатики» : матеріали конф., м. Київ, 11–12 трав. 2023 р. Київ, 2023. С. 322-326.
8. Alghawli A. S. A., Radivilova T. Resilient cloud cluster with DevSecOps security model, automates a data analysis, vulnerability search and risk calculation. Alexandria Engineering Journal. 2024. Vol. 107. P. 136–149. URL: https://doi.org/10.1016/j.aej.2024.07.036 (date of access: 23.05.2025).
9. Design of mechanisms for ensuring the execution of tasks in project planning / O. Mulesa et al. Eastern-European Journal of Enterprise Technologies. 2023. Vol. 2, no. 4 (122). P. 16–22. URL: https://doi.org/10.15587/1729-4061.2023.277585 (date of access: 23.05.2025).
10. Development of Secure Containerized Applications with a Microservices Architecture / S. Spasiteleva et al. Cybersecurity: Education, Science, Technique. 2023. Vol. 1, no. 21. P. 193–210. URL: https://doi.org/10.28925/2663-4023.2023.21.193210 (date of access: 23.05.2025).
11. Новікова Д. О. Дослідження методів забезпечення безпеки ланцюжків поставок програмного забезпечення : пояснювальна записка до кваліфікаційної роботи здобувача вищої освіти на другому (магістерському) рівні, спеціальність 125 Кібербезпека / Д. О. Новікова ; М-во освіти і науки України, Харків. нац. ун-т радіоелектроніки. Харків, 2024. 90 с.
12. Kivilis Y., Sadhu S. Preventing API Breaches Using Salt Security with AWS WAF and Amazon API Gateway | Amazon Web Services. URL: https://aws.amazon.com/blogs/apn/preventing-api-breaches-using-salt-security-with-aws-waf-and-amazon-api-gateway (date of access: 07.05.2025).
13. OWASP top 10 API security risks – 2023 - OWASP API security top 10. OWASP Foundation, the Open Source Foundation for Application Security | OWASP Foundation. URL: https://owasp.org/API-Security/editions/2023/en/0x11-t10/ (date of access: 07.05.2025).
14. Ponaka K. R. API security - protecting the backbone of digital transformation. Journal of mathematical & computer applications. 2023. P. 1–5. URL: https://doi.org/10.47363/jmca/2024(2)e129 (date of access: 07.05.2025).
15. Anand V. API Security with Apigee and Google Cloud Armor | Google Cloud Blog. URL: https://cloud.google.com/blog/products/api-management/api-security-with-apigee-and-google-cloud-armor (date of access: 07.05.2025).

КІБЕРБЕЗПЕКА: освіта, наука, техніка
CYBERSECURITY: EDUCATION, SCIENCE, TECHNIQUE
№ 4 (28), 2025
ISSN 2663 - 4023

Andrii Skip
Postgraduate Student of Information Protection Department
Vinnytsia National Technical University,Vinnytsia, Ukraine
ORCID ID: 0009-0009-3334-4145
phd@askip.me

Yurii Baryshev
PhD (Eng), Associate Professor of Information ProtectionDepartment
Vinnytsia National Technical University,Vinnytsia, Ukraine
ORCID ID: 0000-0001-8324-8869
yuriy.baryshev@vntu.edu.ua

SOFTWARE SUPPLY CHAIN SECURITY ANALYSIS

Abstract. This article addresses the pressing issue of securing application programming interfaces (APIs) in cloud environments within the broader context of software supply chain protection. Given the widespread adoption of microservice architectures and open REST/GraphQL APIs, the need to defend APIs against attacks has become critically important. The paper presents a formalization of the software supply chain model, which serves as the basis for analyzing potential attack vectors at each stage of the chain — from the compromise of repositories, dependencies, and CI/CD servers to artifact storage and deployment environments. Special attention is given to two key security mechanisms: the Web Application Firewall (WAF) and the API Gateway. A comparative analysis is provided covering their functionality, request handling approaches, security levels, logging methods, and integration into cloud infrastructures. The study identifies that WAFs detect and block HTTP-level attacks — such as SQL injections, XSS, and CSRF — based on signatures or behavioral rules, while API Gateways act as intermediaries managing request routing, authentication, authorization, access policies, and API call control. The research findings demonstrate that WAFs and API Gateways are not interchangeable but should be used as complementary components. The article proposes a combined cloud-based API protection model in which the WAF provides perimeter control and the API Gateway enforces logical access control. Additionally, it presents methods for evaluating the effectiveness of these mechanisms, including empirical testing in sandboxed environments, comparative benchmarking under various configurations, and threat modeling. Finally, the paper outlines several unresolved challenges and directions for further research in API and CI/CD security within cloud environments.

Keywords: cybersecurity; API protection; CI/CD; software supply chain; threat analysis; web-application protection.

REFERENCES (TRANSLATED AND TRANSLITERATED)

1. Türetken B. Enhancing security with cloud-based API management : Dissertation. Stockholm, 2024. 83 p. URL: https://www.diva-portal.org/smash/get/diva2:1903654/FULLTEXT01.pdf (date of access: 07.05.2025).
2. Dasher J. Why do I need API security if I have a WAF and API gateway?. cequence.ai. URL: https://www.cequence.ai/blog/api-security/why-do-i-need-api-security-if-i-have-a-waf-and-api-gateway (date of access: 07.05.2025).
3. McNaught B. API security best practices. *ISC2*. URL: https://www.isc2.org/Insights/2023/08/api-security-best-practices-in-the-hybrid-multi-cloud-digital-world#:~:text=were%20four%20key%20approaches:%20Web,going%20to%20be%20the%20converse” (date of access: 07.05.2025).
4. Cosgrove J., Andreev I. A. Protecting GraphQL APIs from Malicious Queries. The Cloudflare Blog. URL: https://blog.cloudflare.com/protecting-graphql-apis-from-malicious-queries/ (date of access: 07.05.2025).
5. What is a CI/CD pipeline?. Red Hat - We make open source technologies for the enterprise. URL: https://www.redhat.com/en/topics/devops/what-cicd-pipeline (date of access: 07.05.2025).
6. GitLab. What is a code review?. *The most-comprehensive AI-powered DevSecOps platform | GitLab*. URL: https://about.gitlab.com/topics/version-control/what-is-code-review/ (date of access: 07.05.2025).

7. Khanas M. L., Baranovskyi O. M. The Current State of Kubernetes Security: Utilization of Existing Tools and Approaches for Penetration Testing. XXI All-Ukrainian Scientific and Practical Conference for Students, Postgraduates, and Young Scientists "Theoretical and Applied Problems of Physics, Mathematics, and Informatics": conference proceedings, Kyiv, May 11–12, 2023. Kyiv, 2023.
8. Alghawli A. S. A., Radivilova T. Resilient cloud cluster with DevSecOps security model, automates a data analysis, vulnerability search and risk calculation. Alexandria Engineering Journal. 2024. Vol. 107. P. 136–149. URL: https://doi.org/10.1016/j.aej.2024.07.036 (date of access: 23.05.2025).
9. Design of mechanisms for ensuring the execution of tasks in project planning / O. Mulesa et al. Eastern-European Journal of Enterprise Technologies. 2023. Vol. 2, no. 4 (122). P. 16–22. URL: https://doi.org/10.15587/1729-4061.2023.277585 (date of access: 23.05.2025).
10. Development of Secure Containerized Applications with a Microservices Architecture / S. Spasiteleva et al. Cybersecurity: Education, Science, Technique. 2023. Vol. 1, no. 21. P. 193–210. URL: https://doi.org/10.28925/2663-4023.2023.21.193210 (date of access: 23.05.2025).
11. Novikova D. O. Research on Methods for Securing Software Supply Chains: explanatory note to the qualification thesis of the second (master's) level higher education applicant, specialty 125 Cybersecurity / D. O. Novikova; Ministry of Education and Science of Ukraine, Kharkiv National University of Radio Electronics. – Kharkiv, 2024. – 90 p.
12. Kivilis Y., Sadhu S. Preventing API breaches using salt security with AWS WAF and amazon API gateway | amazon web services. Amazon Web Services. URL: https://aws.amazon.com/blogs/apn/preventing-api-breaches-using-salt-security-with-aws-waf-and-amazon-api-gateway (date of access: 07.05.2025).
13. OWASP top 10 API security risks – 2023 - OWASP API security top 10. OWASP Foundation, the Open Source Foundation for Application Security | OWASP Foundation. URL: https://owasp.org/API-Security/editions/2023/en/0x11-t10/ (date of access: 07.05.2025).
14. Ponaka K. R. API security - protecting the backbone of digital transformation. Journal of mathematical & computer applications. 2023. P. 1–5. URL: https://doi.org/10.47363/jmca/2024(2)e129 (date of access: 07.05.2025).
15. Anand V. API Security with Apigee and Google Cloud Armor | Google Cloud Blog. URL: https://cloud.google.com/blog/products/api-management/api-security-with-apigee-and-google-cloud-armor (date of access: 07.05.2025).