DOI 10.34185/1562-9945-5-166-2026-24
УДК 004.056.5:004.738

А.А. Кравчук, М.В. Онай

ПРОЄКТУВАННЯ АРХІТЕКТУРИ ПРОГРАМНОЇ СИСТЕМИ ВИЯВЛЕННЯ ТА БЛОКУВАННЯ HTTP LR-DDOS АТАК НА ПЛАТФОРМІ KUBERNETES

Анотація. У статті запропоновано та реалізовано розподілену архітектуру захисту мікросервісів від низькошвидкісних DDoS-атак (LR-DDoS) прикладного рівня у середовищі Kubernetes, без модифікації коду цільових застосунків та без відчутних затримок легітимного трафіку. Архітектурне рішення базується на розділенні площини маршрутизації (Data Plane) та аналізу (Control Plane): атрибути прикладного рівня збираються розширенням EnvoyFilter після дешифрування TLS-трафіку на вхідному шлюзі Istio, а мережеві атрибути – засобами eBPF/XDP, після чого асинхронно передаються до брокера Apache Kafka. Розроблені мовою Go модуль класифікації з горизонтальним масштабуванням та контролер на базі шаблону Kubernetes Operator зберігають джерела DDoS-атак у Redis із механізмом TTL для видалення застарілих даних та оновлюють правила блокування в ресурсі AuthorizationPolicy. Це забезпечує три сценарії реагування на атаку: повну класифікацію невідомих сесій, прискорену перевірку за збереженими сигнатурами та блокування запитів на шлюзі зі складністю O(1). Експериментальні дослідження підтвердили високу продуктивність системи: затримка від EnvoyFilter становить лише 2,933 мс, а час реакції – 3,15 с для нових джерел, 0,8 с для відомих сигнатур та 5,674 мс для блокування на рівні шлюзу. Модуль класифікації обробляє 4000 сесій за 32,5 мс із використанням 3,09 МБ оперативної пам'яті. Порівняно з брандмауером вебзастосунків Coraza WAF система демонструє у 12,3 раза меншу затримку (3,667 мс проти 45 мс) та значно вищу пропускну здатність (1973 проти 473 запитів/с).

Ключові слова: DDoS, низькошвидкісні DDoS-атаки, LR-DDoS, програмна система, інженерія програмного забезпечення, мікросервісна архітектура, Kubernetes, Istio, захист інформації, криптографія.

Постановка проблеми. Розподілені атаки на відмову в обслуговуванні (distributed denial-of-service, DDoS) залишаються однією з найпоширеніших причин порушення безперебійної роботи вебсервісів, причому кількість таких інцидентів продовжує стрімко зростати [1]. Особливу небезпеку в цьому контексті становлять низькошвидкісні атаки прикладного рівня (low-rate DDoS, LR-DDoS), спрямовані на протоколи HTTP/HTTPS. На відміну від об'ємних (volumetric) DDoS-атак, які перевантажують канали зв'язку безперервним потоком мережевих пакетів даних із високою частотою [2], LR-DDoS атаки (наприклад, Slowloris [3], RUDY [4], Slow Read [5]) застосовують принципово інший підхід: вони штучно утримують мережеві сесії відкритими за допо-

© Кравчук А.А., Онай М.В., 2026
Copyright for this paper by its authors. Use permitted under License CC BY 4.0.
ISSN 1562-9945 (Print)
ISSN 2707-7977 (Online)

могою навмисного затягування процесу передачі або зчитування фрагментів HTTP-запитів, що призводить до вичерпання пулу TCP-з'єднань, переповнення черг очікування, блокування потоків операційної системи та надмірного споживання оперативної пам'яті цільового сервера [6; 7]. Через малу інтенсивність генерації пакетів трафік LR-DDoS атак за своїми параметрами є подібним до активності легітимних користувачів із нестабільним мережевим з'єднанням, що значно ускладнює їх ідентифікацію за допомогою статистичних методів та звичайних механізмів контролю трафіку (зокрема, обмеження інтенсивності запитів), оскільки швидкість надходження зловмисних запитів зазвичай не перевищує встановлених порогових значень [8].

Сучасні програмні системи дедалі частіше проєктуються на основі мікросервісної архітектури [9] та працюють у розподілених хмарних середовищах із застосуванням технологій контейнеризації, де загальноприйнятим стандартом автоматизації та оркестрації є платформа Kubernetes [10]. Розгортання програмних компонентів у межах кластера Kubernetes передбачає об'єднання обчислювальних вузлів у спільний пул ресурсів, де керування контейнерами, балансування навантаження та маршрутизація трафіку між ними здійснюються динамічно [10]. Подібний підхід суттєво підвищує гнучкість та горизонтальну масштабованість систем, проте водночас має ризики для їхньої загальної відмовостійкості. Зокрема, через взаємозв'язки між елементами системи, тимчасове обмеження доступності або вичерпання обчислювальних ресурсів одного критичного компонента (наприклад, централізованого сервісу автентифікації) внаслідок DDoS-атаки здатне спровокувати лавиноподібну деградацію продуктивності всього ланцюга обробки запитів. Окрім того, процес побудови систем захисту від атак ускладнюється зазвичай притаманною мікросервісним рішенням гетерогенністю (що полягає у використанні різних технологій та інструментарію програмування), де окремі компоненти створюються різними командами розробників із застосуванням різних мов програмування, архітектурних шаблонів та вебфреймворків. За таких умов реалізація механізмів протидії LR-DDoS атакам всередині прикладного коду кожного сервісу є неефективною, оскільки вона призводить до розосередження алгоритмів захисту, надмірного дублювання вихідного коду та суттєвого ускладнення централізованого керування політиками фільтрації мережевого трафіку.

З огляду на вищезазначене, актуальною проблемою є проєктування уніфікованої архітектури програмної системи, яка забезпечувала б виявлення та блокування HTTP LR-DDoS атак у кластері Kubernetes без внесення змін до вихідного коду окремих мікросервісів, гарантуючи при цьому збереження їхньої автономності, масштабованості та відмовостійкості. Складність вирішення цієї проблеми полягає в необхідності одночасного здійснення комплексного аналізу вхідного інтернет-трафіку та забезпечення високої швидкості обробки HTTP-запитів. З одного боку, забезпечення високої точності виявлення LR-DDoS атак прикладного рівня зазвичай вимагає використання даних як мережевого, так і прикладного рівнів мережевої моделі OSI [11]. Враховуючи широке використання протоколу HTTPS у сучасних системах, необхідною передумовою є отримання доступу до вмісту HTTP-запитів у дешифрованому вигляді до їхньої передачі на

ISSN 1562-9945 (Print)
ISSN 2707-7977 (Online)

оброблення в цільовий мікросервіс або застосунок. З іншого боку, інтеграція методів виявлення DDoS-атак безпосередньо у вихідний код мікросервісів є неефективною, оскільки це призводить до суттєвого зростання затримок обробки запитів та спричиняє додаткове навантаження на обчислювальні ресурси, що загрожує зниженням загальної продуктивності всієї системи. Відтак, постає задача виокремлення механізмів захоплення трафіку, його аналізу для виявлення DDoS-атак та подальшого блокування, що передбачає їхню реалізацію у вигляді уніфікованого інфраструктурного рішення на рівні кластера. Практична цінність такого підходу полягає в забезпеченні захисту системи без втручання у вихідний код мікросервісів.

Аналіз останніх досліджень і публікацій. Сучасні дослідження, присвячені проблемі виявлення LR-DDoS атак, пропонують широкий спектр методів та алгоритмічних рішень, серед яких найбільшого поширення набули метод статистичного аналізу ентропії [12], метод на основі прогнозування часових рядів [13; 14], а також методи на основі машинного навчання [15], зокрема нейронних мереж [16]. Водночас для підвищення точності ідентифікації таких атак у межах цих підходів можуть залучатися додаткові атрибути прикладного рівня, що своєю чергою передбачає аналіз вмісту HTTP-запитів у дешифрованому вигляді. Зокрема, подібний підхід покладено в основу модифікованого методу на основі часових рядів, який передбачає перехід від аналізу суто мережевих показників до атрибутів прикладного рівня [17], водночас зберігаючи можливість комплексного поєднання даних обох рівнів. Проте більшість розглянутих робіт зосереджена виключно на аналізі ефективності самих методів виявлення DDoS-атак, тоді як питання проєктування програмної архітектури для їхнього впровадження у розподілені мікросервісні системи залишається відкритим.

Аналіз наявних варіантів розгортання засобів захисту від DDoS-атак дозволяє згрупувати їх у три основні категорії. Перша з них базується на використанні традиційних брандмауерів для вебзастосунків (Web Application Firewall, WAF), які розгортаються на зовнішніх точках входу до системи, зокрема на вхідних мережевих шлюзах [18; 19]. Такі програмні рішення, зокрема на основі ModSecurity та OWASP Core Rule Set [20], перехоплюють і послідовно аналізують кожен запит, перед тим як передати його до цільового сервісу. Основний недолік такого підходу полягає в тому, що вхідний шлюз перетворюється на критичний вузол інфраструктури, перевантаження або відмова якого зупиняє роботу всієї системи. Оскільки виявлення LR-DDoS атак потребує ресурсоємного аналізу вмісту та поведінкових характеристик HTTP-запитів, раптове зростання шкідливого трафіку може перевантажити потужності вузла мережевого шлюзу, спричиняючи зростання затримок під час обробки легітимного трафіку та зумовлюючи високий рівень хибнопозитивних спрацьовувань.

Друга категорія передбачає використання вбудованих модулів безпеки або проміжного програмного забезпечення (middleware) [21], коли механізми виявлення DDoS-атак інтегруються безпосередньо у вихідний код мікросервісів. Хоча це нівелює ризик перевантаження вхідного мережевого шлюзу, такий підхід вимагає модифікації вихідного коду вебфреймворків та створює додаткову залежність для бізнес-логіки самого застосунку від впровадженої системи захисту. У системах, де мікросервіси реалізовані з

використанням різних мов програмування та технологічних стеків, супровід таких програмних модулів стає складним завданням. Крім того, цей підхід порушує базовий принцип розділення відповідальності (separation of concerns) [22], оскільки перекладає завдання виявлення і блокування DDoS-атак на рівень прикладного коду застосунків.

Третя категорія охоплює рішення на основі асинхронного аналізу збережених копій захопленого мережевого трафіку або журналів доступу до вебресурсів, сформованих вебсерверами чи мережевими шлюзами для фіксації параметрів HTTP-запитів [23]. За такого сценарію аналіз атрибутів запитів здійснюється окремими програмними компонентами поза основним ланцюгом обробки, що дозволяє уникнути додаткових затримок у роботі мікросервісів. Проте суттєвим недоліком асинхронного підходу є часова затримка між початком атаки та моментом її виявлення. Оскільки аналіз даних виконується у відкладеному режимі, реагування на атаку зазвичай здійснюється із запізненням, що є критичним під час LR-DDoS атак, коли несвоєчасне блокування джерел зловмисних запитів призводить до перевантаження ресурсів системи та недоступності вебзастосунків для легітимних користувачів.

З огляду на проведений аналіз, невирішеною частиною загальної проблеми залишається відсутність цілісної програмної архітектури, яка б, по-перше, отримувала атрибути прикладного рівня безпосередньо після дешифрування HTTPS-запитів без втручання у вихідний код застосунків; по-друге, виконувала ресурсоємний аналіз в асинхронному режимі, мінімізуючи вплив на легітимний трафік, але водночас гарантуючи мінімальний час виявлення DDoS-атак; і по-третє, забезпечувала оперативне блокування джерел зловмисних запитів ще до того, як вони досягнуть цільових вебзастосунків. Проєктуванню та програмній реалізації такої архітектури і присвячено цю статтю.

Мета дослідження. Метою дослідження є проєктування та програмна реалізація розподіленої архітектури системи виявлення і блокування HTTP LR-DDoS атак для Kubernetes-кластерів, яка забезпечує захист від атак без модифікації вихідного коду цільових мікросервісів. Ключовою вимогою до розроблюваної системи є зменшення часу реагування (від моменту початку атаки до блокування її джерел) порівняно з наявними системами виявлення DDoS-атак, причому додаткова затримка в обробленні та маршрутизації легітимного трафіку від вхідного шлюзу до цільового мікросервісу не повинна перевищувати 10 мс.

Для досягнення поставленої мети необхідно вирішити такі завдання: спроєктувати розподілену архітектуру системи захисту від DDoS-атак; здійснити її програмну реалізацію для кластерного середовища Kubernetes; розробити методику експериментального оцінювання; провести тестування розробленої системи та порівняти показники її ефективності з наявними аналогами.

Викладення основного матеріалу дослідження. Зважаючи на ключову вимогу щодо забезпечення захисту від DDoS-атак без втручання у вихідний код мікросервісів, базовим інфраструктурним середовищем для реалізації архітектури обрано технологію сервісної сітки (Service Mesh) [24; 25]. Її застосування передбачає винесення функцій управління мережевим трафіком між мікросервісами в окремий шар інфраструктури,

ISSN 1562-9945 (Print)
ISSN 2707-7977 (Online)

який реалізується за допомогою мережі розподілених проксі-серверів. Одним із найбільш поширених рішень для реалізації цього підходу в кластерах Kubernetes є платформа Istio [26], яка забезпечує централізоване керування трафіком між сервісами, автентифікацію та авторизацію, а також гнучке налаштування політик безпеки та правил маршрутизації. Головною перевагою Istio в контексті цього дослідження є використання високопродуктивних проксі-серверів Envoy [27] для кожного мікросервісу та вхідних шлюзів. Завдяки механізму програмних розширень EnvoyFilter [28], сервісна сітка Istio надає можливість перехоплювати та аналізувати трафік безпосередньо на вхідних мережевих шлюзах кластера вже після дешифрування HTTPS-запитів, що забезпечує доступ до їхніх атрибутів в умовах захищеного з'єднання без внесення змін до вихідного коду застосунків.

Архітектура запропонованої програмної системи. В основу запропонованої архітектури покладено принцип розділення площини захоплення трафіку (Data Plane), яка здійснює його маршрутизацію, та площини аналізу і прийняття рішень (Control Plane). Перехоплення HTTP-запитів виконується безпосередньо в проксі-сервері Envoy на вхідному шлюзі кластера . На цьому етапі із запитів відбираються виключно необхідні атрибути, які без додаткових трансформацій асинхронно надсилаються до наступного компонента системи. Це дозволяє уникнути додаткових затримок під час маршрутизації трафіку до мікросервісів. Оскільки механізм виявлення DDoS-атак повністю винесено в окремий модуль, часові витрати на оброблення легітимних запитів є мінімальними.

На вхідному шлюзі Istio Ingress Gateway, де здійснюється дешифрування HTTPS-трафіку, розгортається розширення EnvoyFilter, реалізоване скриптовою мовою програмування Lua. Воно отримує параметри кожного запиту (HTTP-метод, URI, значення заголовків User-Agent, Accept-Language, Accept-Encoding, перелік імен всіх наявних заголовків у запиті, а також IP-адресу та порт джерела) у дешифрованому вигляді ще до моменту перенаправлення запиту у внутрішню мережу кластера. Важливо зазначити, що реалізований EnvoyFilter виконує виключно функцію експорту: за допомогою асинхронного виклику httpCall дані надсилаються до відповідного сервісу, який збирає атрибути запитів, у фоновому режимі (без очікування на результат обробки). Завдяки цьому критичний шлях оброблення та маршрутизації трафіку до цільових застосунків не залежить від процесу виявлення DDoS-атак.

Оскільки запропоноване розширення EnvoyFilter після дешифрування HTTPS-трафіку не має доступу до атрибутів мережевого та транспортного рівнів (зокрема значення TTL, SYN, розміру вікна TCP), які є важливими для виявлення LR-DDoS атак типу Slow Read [5], до архітектури розроблюваної системи інтегровано додатковий модуль аналізу мережевого стека на рівні протоколів TCP та UDP. Цей модуль реалізовано в середовищі Kubernetes як об'єкт DaemonSet [29], що гарантує його глобальне розгортання на всіх робочих вузлах кластера. Завдяки використанню технологій eBPF/XDP [30] програмний модуль перехоплює мережевий трафік безпосередньо на рівні мережевих інтерфейсів вузлів. Перехоплені пакети групуються за унікальною па-

рою ідентифікаторів (IP-адреса та номер порту відправника), після чого зібрані дані експортуються до іншого сервісу для подальшої обробки.

Архітектура розробленої системи ґрунтується на модульному принципі та використовує механізми просторів імен (namespaces) середовища Kubernetes для логічної ізоляції компонентів [29]. Зокрема, процеси безпосереднього перехоплення трафіку та блокування джерел LR-DDoS атак виконуються у системному просторі імен istio-system на основі вхідного мережевого шлюзу. Усі процеси обробки даних інкапсульовано у виділеному просторі імен ddos-system, який включає захоплення HTTP-запитів і мережевих пакетів, їх оброблення, класифікацію, а також збереження результатів. Для дотримання принципу єдиної відповідальності функції автоматизованого реагування на DDoS-атаки винесено в окремий простір імен ddos-operator-system. Такий підхід гарантує повну незалежність захисних механізмів від просторів імен цільових застосунків, для захисту яких створюється ця система.

Узагальнену UML-діаграму розгортання запропонованої архітектури в кластері Kubernetes та інформаційні потоки між її складовими наведено на рис. 1.

Відповідно до наведеної діаграми (рис. 1), ключові компоненти системи виконують такі функції:

– Istio Ingress Gateway – вхідний мережевий шлюз на основі проксі-сервера Envoy, в якому налаштовані розширення EnvoyFilter для виокремлення та експорту метаданих і заголовків (атрибутів) HTTP-запитів та AuthorizationPolicy для блокування шкідливого трафіку від DDoS-атак.

– HTTP Attributes Collector – сервіс, що отримує експортовані атрибути HTTP-запитів від розширення EnvoyFilter, формує з них структуровані об’єкти клієнтських сесій та надсилає їх брокеру повідомлень для подальшого аналізу модулем DDoS Classifier.

– eBPF Packet Collector – сервіс, що здійснює захоплення мережевих пакетів та надсилає атрибути пакетів на рівні мережевих та транспортних протоколів до брокера повідомлень, які потім будуть оброблені модулем DDoS Classifier.

– Apache Kafka – брокер повідомлень, що забезпечує асинхронну потокову передачу даних і усуває жорстку архітектурну залежність між компонентами системи.

– DDoS Classifier – програмний модуль, який відповідає за реалізацію методу виявлення DDoS-атак: він зчитує з брокера повідомлень атрибути прикладного та мережевого рівнів, здійснює їхню класифікацію для ідентифікації джерел атак і передає виявлені ідентифікатори до сховища Redis.

– Redis – високошвидкісне сховище даних типу “ключ-значення”, призначене для збереження ідентифікованих джерел DDoS-атак (IP-адреса та порт) із механізмом автоматичного видалення застарілих записів (TTL).

– DDoS Protection Operator – контролер, який забезпечує початкове розгортання інфраструктури системи на кластері Kubernetes, а також безперервне зчитування виявлених джерел атак зі сховища та автоматичне оновлення правил блокування AuthorizationPolicy на вхідному шлюзі.

ISSN 1562-9945 (Print)
ISSN 2707-7977 (Online)

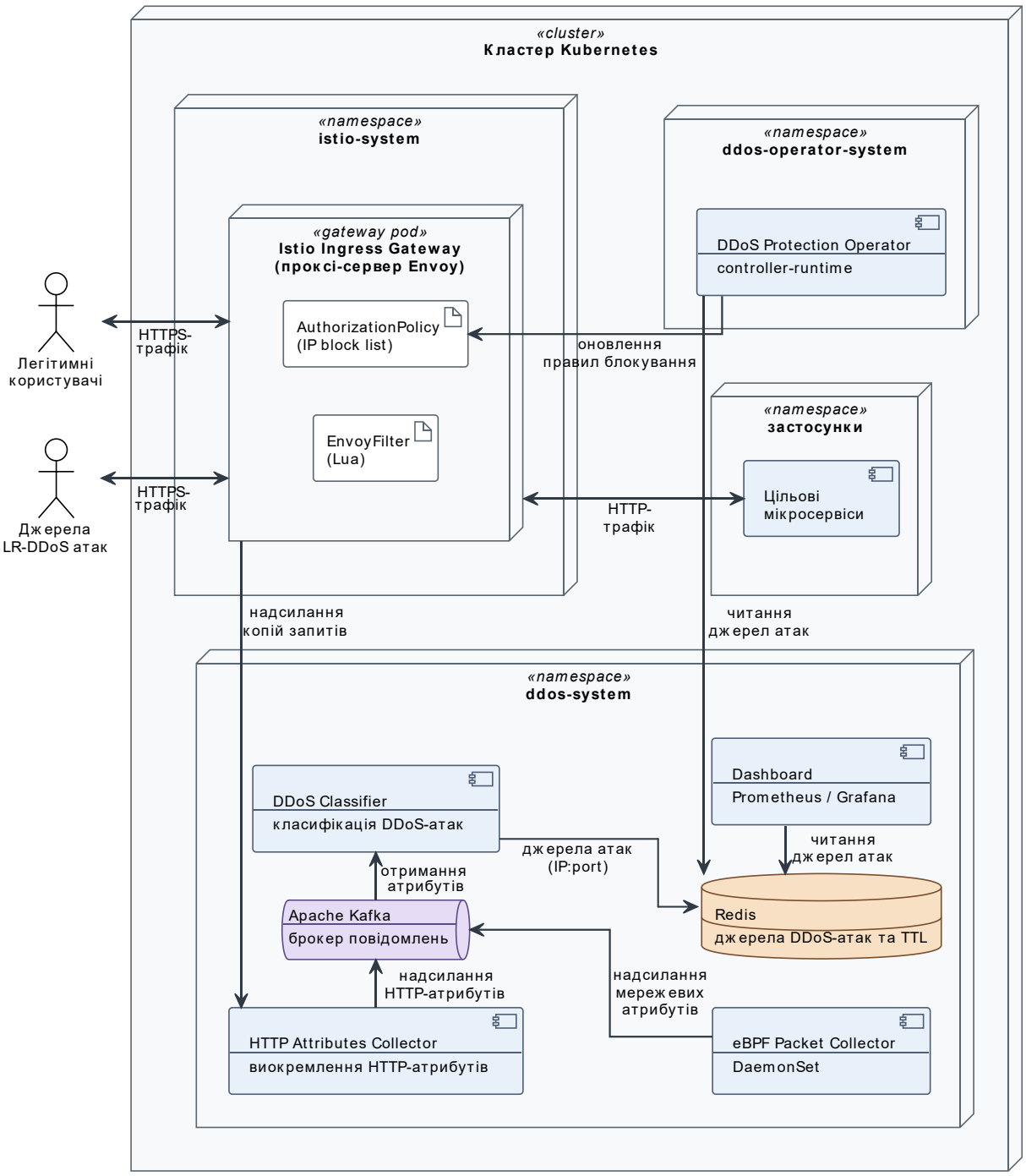


Рисунок 1 – UML-діаграма розгортання програмної системи в кластері Kubernetes

У межах описаної архітектури ресурсоємні сервіси, до яких належать HTTP Attributes Collector та DDoS Classifier, підтримують механізм горизонтального масштабування Horizontal Pod Autoscaler (HPA) для забезпечення стабільної роботи під навантаженням. Застосування брокера повідомлень Apache Kafka запроваджує подійно-орієнтовану взаємодію за шаблоном “публікація-підписка” (publish-subscribe). Це створює слабку зв’язність компонентів і гарантує, що процес швидкого зчитування атрибутів запитів не блокуватиметься тривалим аналізом навіть за умов масованих атак. Також варто зазначити, що використання архітектурного шаблону Kubernetes Operator у

контролері DDoS Protection Operator автоматизує адміністрування розробленої програмної системи та перетворює її на нативне розширення кластера [31].

Особливості програмної реалізації запропонованої системи. Ключові програмні компоненти системи, що відповідають за захоплення мережевих пакетів, оброблення HTTP-запитів, класифікацію DDoS-атак та встановлення правил фільтрації зловмисного трафіку, розроблені мовою Go. Вибір цієї мови обґрунтовується її вбудованими механізмами багатопотоковості, автоматичного керування пам'яттю (garbage collection) та помірним використанням апаратних ресурсів. Для того щоб швидкість аналізу не обмежувала загальну пропускну здатність системи під час DDoS-атак, для оброблення кожної сесії використовуються легкі потоки (goroutines), адже вони не вимагають виділення значних ресурсів, на відміну від потоків на рівні операційної системи. Це дозволяє одночасно аналізувати тисячі клієнтських сесій, а також гарантує стабільну роботу Control Plane без надмірного споживання оперативної пам'яті навіть за екстремальних навантажень.

Розгортання розробленої системи на кластері Kubernetes відбувається за допомогою контролера DDoS Protection Operator, який реалізовано на основі фреймворку controller-runtime [32]. Інтерфейсом конфігурування системи виявлення та блокування DDoS-атак виступає об'єкт для визначення користувацького ресурсу (Custom Resource Definition, CRD) під назвою DDoSProtection. Цей об'єкт розширює API кластера Kubernetes новим класом ресурсів і формалізує структуру даних: як конфігураційні параметри, що можуть бути задані в об'єкті, так і поля для відображення поточного стану системи. На його основі створюється декларативний екземпляр (Custom Resource, CR), який інкапсулює всі налаштування розроблюваної системи: параметри розгортання власних або підключення до зовнішніх екземплярів брокера повідомлень Kafka та сховища даних Redis, назви мережевих інтерфейсів для захоплення трафіку, назву вхідного шлюзу Istio, посилання на ресурс AuthorizationPolicy, інтервали синхронізації для оновлення правил блокування зі сховища Redis тощо. Основним завданням контролера є реалізація циклу узгодження (reconciliation loop) – ідемпотентної функції, яка виключно за декларативним описом ресурсу автоматично генерує та підтримує в актуальному стані всі залежні об'єкти кластера (Deployment, DaemonSet, Service), необхідні для роботи системи. Застосування механізмів локального кешування, наданих фреймворком, дозволяє суттєво знизити навантаження на API-сервер Kubernetes під час моніторингу стану цих об'єктів.

Окрім керування життєвим циклом розробленої системи, контролер DDoS Protection Operator безперервно оновлює правила фільтрації зловмисних джерел у ресурсі AuthorizationPolicy. Цей механізм винесено в окремий паралельний процес, незалежний від головного циклу узгодження компонентів системи. Така ізоляція зумовлена різним характером подій, що ініціюють ці процеси: якщо основний цикл узгодження об'єктів системи реагує на зміну стану розгорнутих ресурсів у кластері, то оновлення правил блокування потребує зчитування нових ідентифікаторів загроз зі сховища Redis, як тільки вони там з'являються. Об'єднання цих завдань у межах єдиного циклу узго-

ISSN 1562-9945 (Print)
ISSN 2707-7977 (Online)

дження унеможливило б миттєве блокування джерел DDoS-атак, оскільки для актуалізації фільтрів трафіку контролеру довелося б очікувати на виникнення подій зміни стану розгорнутих компонентів. Цей паралельний процес передбачає періодичне зчитування поточного переліку заблокованих IP-адрес зі сховища Redis, обчислення бажаного стану ресурсу AuthorizationPolicy та порівняння його з фактичним станом у кластері. Модифіковані правила блокування джерел DDoS-атак застосовуються до об'єкта AuthorizationPolicy лише в разі виявлення розбіжностей, що мінімізує кількість операцій запису до API-сервера Kubernetes. Для взаємодії з ресурсами Istio в кластері використовуються типізовані структури даних із відповідних програмних бібліотек, що забезпечує коректність модифікації об'єктів Istio.

Особливістю реалізації модуля класифікації трафіку DDoS Classifier є використання легких потоків (goroutines), що дозволяє обробляти отримані дані з високою продуктивністю. Кожна клієнтська сесія, що визначається унікальною парою з IP-адреси джерела та номера порту, аналізується в окремому потоці. У межах кожної сесії здійснюється одночасний аналіз атрибутів мережевого та прикладного рівнів, які надходять від сервісів eBPF Packet Collector та HTTP Attributes Collector відповідно. Для аналізу даних мережевого рівня реалізовано метод на основі прогнозування часових рядів [14], а для прикладного рівня – реалізовано модифіковану версію цього методу [17]. Аналіз даних на обох рівнях передбачає бінарну класифікацію клієнтської сесії для її ідентифікації як легітимної або зловмисної. Після завершення роботи паралельних потоків результати класифікації з обох рівнів об'єднуються за допомогою примітива синхронізації sync.WaitGroup. Якщо хоча б один із рівнів класифікував DDoS-атаку, пара з IP-адреси джерела та номера порту цієї клієнтської сесії вноситься до бази сигнатур атак. Формування та перевірка бази сигнатур здійснюються за допомогою структури sync.Map, яка є безпечною для використання в багатопотоковому середовищі, а також оптимізованою для інтенсивного одночасного читання та запису без взаємних блокувань. Це дозволяє виконувати швидку перевірку наявності сигнатури поточної сесії у переліку відомих джерел DDoS-атак.

Також у процесі аналізу в модулі DDoS Classifier кожній клієнтській сесії ставиться у відповідність набір сигнатур, які обчислюються на основі атрибутів мережевого та прикладного рівнів. Використання сигнатур обґрунтовується тим, що під час DDoS-атак вузли одного ботнету часто генерують трафік зі схожими характеристиками, але з різних IP-адрес. Щоб уникнути ресурсоємної класифікації трафіку від нових IP-адрес раніше виявленого ботнету та пришвидшити блокування таких джерел, модуль класифікації спочатку обчислює сигнатури з атрибутів обох рівнів і перевіряє, чи наявні вони в базі сигнатур відомих атак. У разі виявлення збігу результат класифікації повертається миттєво, визначаючи таку сесію як зловмисну. Якщо ж збігів немає, виконується класифікація атрибутів трафіку на обох рівнях. У випадку, коли за результатами класифікації виявлено DDoS-атаку, до бази сигнатур вноситься не лише пара з IP-адреси та номера порту, а й обчислені сигнатури атрибутів обох рівнів, що відповідають цій парі. Такий підхід дозволяє системі значно швидше виявляти та блокувати нові джерела атак за наявності схожих атрибутів трафіку.

Захоплення інтернет-трафіку здійснюється двома спеціалізованими модулями, які працюють паралельно та незалежно один від одного. Сервіс eBPF Packet Collector, реалізований за допомогою бібліотеки cilium/ebpf, перехоплює пакети за технологією XDP (eXpress Data Path) безпосередньо на рівні драйвера мережевого інтерфейсу. На відміну від стандартних засобів захоплення трафіку (як-от tcpdump), це дозволяє зчитувати та аналізувати атрибути мережевого і транспортного рівнів без створення додаткового навантаження на процесор та без повного копіювання вмісту пакетів [30]. На прикладному рівні HTTP Attributes Collector отримує дані на кінцеву точку (endpoint) API /ingest від розширення EnvoyFilter, яке здійснює асинхронний експорт атрибутів HTTP-запитів. Цей сервіс обробляє отримані заголовки та інші атрибути запитів для того, щоб трансформувати їх у структуровані об’єкти клієнтських сесій. Підготовлені дані з обох сервісів публікуються у відповідні теми (topics) брокера повідомлень Kafka за допомогою бібліотеки segmentio/kafka-go для подальшого аналізу.

UML-діаграму компонентів розробленої системи наведено на рис. 2.

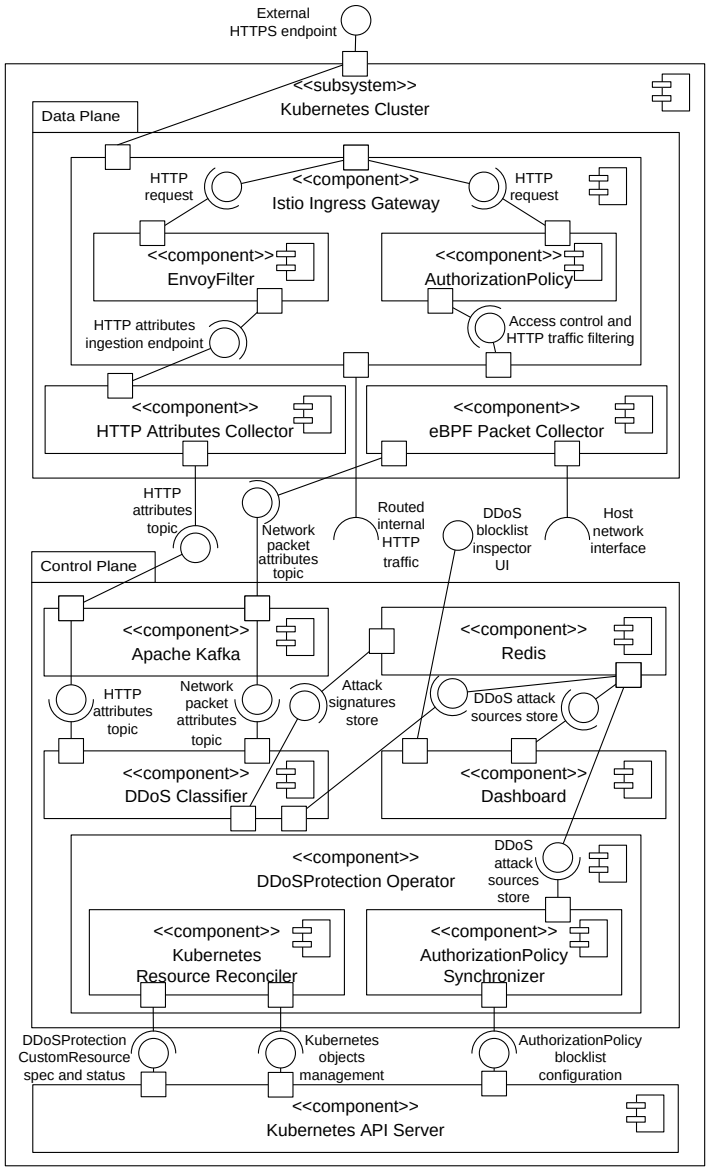


Рисунок 2 – UML-діаграма компонентів системи виявлення і блокування DDoS-атак

ISSN 1562-9945 (Print)
ISSN 2707-7977 (Online)

Сценарії оброблення трафіку та оцінка часу реакції на атаки. Оброблення та маршрутизація вхідного трафіку в запропонованій системі здійснюються за одним із трьох сценаріїв, вибір якого залежить від статусу ідентифікації джерела HTTP-запитів та наявності його сигнатури у базі відомих атак. Такий підхід дозволяє оптимально використовувати обчислювальні ресурси та мінімізувати затримки: ресурсоємна класифікація виконується виключно для принципово нових джерел атак, тоді як ідентифікація вузлів відомих ботнетів пришвидшується, а вже заблоковані джерела DDoS-атак відхиляються миттєво.

Для нових, ще не класифікованих клієнтських сесій із невідомими сигнатурами трафіку застосовується сценарій "повного холодного шляху" (full cold path), що передбачає повний первинний аналіз. У межах цього сценарію атрибути асинхронно експортуються до модуля класифікації атак, тоді як сам трафік безперешкодно маршрутизується до цільового застосунку. Модуль класифікації накопичує отримані дані у межах заданого вікна спостереження T_{win}, після чого обчислює сигнатури поточної сесії. Не знайшовши збігів у базі відомих атак, модуль виконує класифікацію запитів клієнтської сесії за допомогою відповідних методів. У разі класифікації джерела як шкідливого його IP-адреса, номер порту та обчислені сигнатури записуються до сховища Redis із визначеним часом життя (TTL), після завершення якого вони автоматично видаляються. Якщо ж аномалій не виявлено, сесія вважається легітимною, і процес її перевірки завершується. Згодом компонент AuthorizationPolicy Synchronizer у контролері DDoS Protection Operator під час чергового циклу синхронізації зчитує ці дані та оновлює правила в AuthorizationPolicy на вхідному шлюзі. Цей процес є повністю асинхронним і не додає затримок під час оброблення легітимних запитів. Однак він потребує скінченного часу, включаючи збирання атрибутів клієнтських сесій у межах вікна спостереження, що необхідно для виявлення LR-DDoS атак прикладного рівня. Загальний час реакції системи T_{mit}^{full} для цього сценарію становить:

$$T_{mit}^{full} = T_{win} + T_{fp} + T_{class} + T_{redis} + T_{sync} + T_{prop}, \quad (1)$$

де T_{mit}^{full} – час реакції розробленої системи на DDoS-атаку для сценарію "повного холодного шляху"; T_{win} – тривалість вікна спостереження; T_{fp} – час обчислення сигнатур клієнтської сесії; T_{class} – час класифікації запитів клієнтської сесії; T_{redis} – час запису адреси та сигнатур джерела DDoS-атаки у сховищі Redis; T_{sync} – інтервал циклу перевірки контролером нових джерел атак у сховищі Redis; T_{prop} – час застосування оновленої конфігурації правил блокування у проксі-сервері.

Для нових IP-адрес, трафік яких генерується вже виявленим ботнетом, застосовується сценарій "прискореного холодного шляху" (fast-track cold path). Оскільки перший запит від нового джерела пропускається вхідним шлюзом, його атрибути так само асинхронно експортуються до модуля класифікації та накопичуються протягом того самого вікна спостереження T_{win}. Після його завершення модуль обчислює сигнатури сесії за час T_{fp} і знаходить збіг у локальній копії базі сигнатур ще до запуску класифікації. У

цьому разі сесія ідентифікується як шкідлива миттєво – без виконання ресурсоємної класифікації даних мережевого та прикладного рівнів, тобто $T_{class} = 0$. Відповідна IP-адреса, номер порту та сигнатури одразу зберігаються у сховищі Redis для подальшого блокування оператором. Для цього сценарію час реакції T_{mit}^{sig} скорочується через вилучення найважчої обчислювальної складової T_{class}:

$$T_{mit}^{sig} = T_{win} + T_{fp} + T_{redis} + T_{sync} + T_{prop}, \quad (2)$$

де T_{mit}^{sig} – час реакції розробленої системи на DDoS-атаку для сценарію “прискореного холодного шляху”.

Для вже виявлених зловмисних джерел запитів застосовується сценарій “гарячого шляху” (hot path), який забезпечує їхнє миттєве блокування. Компонент Authorization-Policy Synchronizer у контролері DDoS Protection Operator періодично зчитує заблоковані адреси зі сховища Redis та оновлює ресурс AuthorizationPolicy, що працює на рівні вхідного шлюзу. Після цього кожен наступний запит від ідентифікованого джерела DDoS-атак відхиляється проксі-сервером із кодом відповіді HTTP 403. Обчислювальна складність такого блокування становить $O(1)$, оскільки воно виконується без залучення модуля класифікації та без звернень до бази даних. Час реакції для цього сценарію є мінімальним, оскільки з нього виключаються абсолютно всі складові аналізу, класифікації та синхронізації:

$$T_{mit}^{hot} = O(1), \quad (3)$$

де T_{mit}^{hot} – час реакції розробленої системи на DDoS-атаку для сценарію “гарячого шляху”.

Отже, система здійснює ресурсоємну класифікацію профілю атаки лише один раз на повному холодному шляху (1), потім використовує обчислені сигнатури для прискореного виявлення інших IP-адрес ботнету (2), після чого фільтрує всі їхні подальші запити на рівні шлюзу за сталий час (3). Послідовність взаємодій компонентів для всіх трьох сценаріїв наведено на рис. 3.

Аналіз отриманих результатів. Експериментальну оцінку розробленої системи проведено в локальному кластері Kind (Kubernetes 1.36, Go 1.26.2) з одним вузлом, розгорнутому на комп’ютері з процесором Intel Core i5-10210U (1,60 ГГц, 4 ядра) та 8 ГБ оперативної пам’яті. У кластері запущено розроблений програмний комплекс системи захисту і блокування DDoS-атак разом із тестовим вебсервісом httpbin [33]. Оцінювання системи здійснювалося за такими метриками: затримка оброблення HTTP-запитів, пропускна здатність, час реакції на загрози та споживання оперативної пам’яті. Легітимний трафік генерувався утилітою Fortio [34], запущеною всередині кластера, що надсилала HTTP-запити до Istio Ingress Gateway. Трафік із характеристиками LR-DDoS атак формувався за допомогою спеціальних утиліт та скриптів.

ISSN 1562-9945 (Print)
ISSN 2707-7977 (Online)

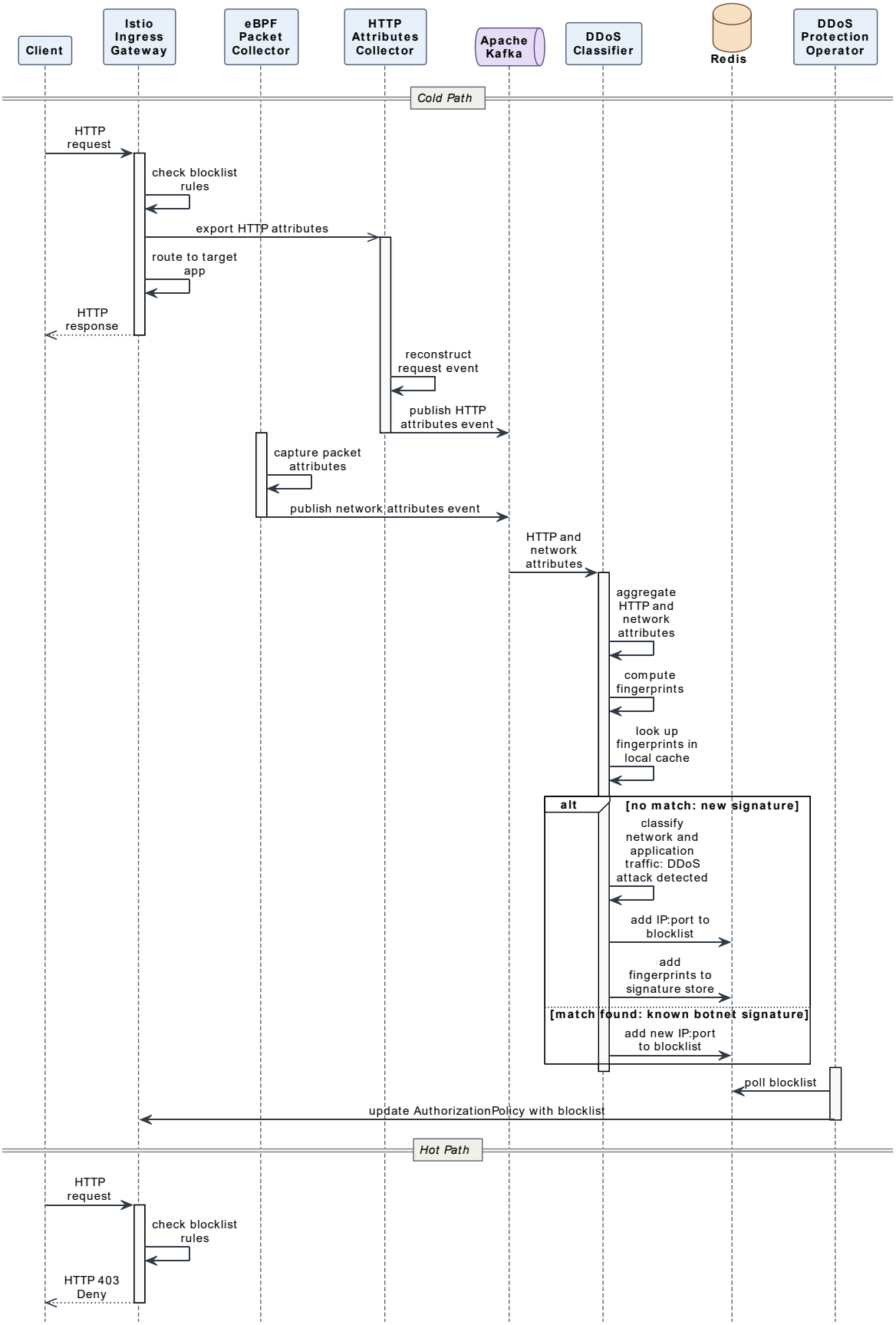


Рисунок 3 – UML-діаграма послідовності взаємодії компонентів для трьох сценаріїв

Зокрема, атака Slow Read відтворювалася за допомогою спеціалізованої утиліти [35], що відкривала TCP-з'єднання з малим приймальним буфером та штучно затримувала зчитування об'ємної відповіді – це дозволило перевірити здатність системи фіксувати аномальне зменшення розміру вікна TCP на транспортному рівні. Окрім цього, розроблено скрипти, які формували серію запитів за допомогою curl із заголовками автоматизованих ботів, а також відтворювали частоту запитів та їхню структуру, типові для LR-DDoS атак. Для оцінки сценарію "повного холодного шляху" скрипт надсилав серію з шести запитів через одне HTTP-з'єднання, при цьому модуль класифікації перед кожним тестовим прогоном запускався із порожньою базою сигнатур. Для сценарію "прискореного холодного шляху" застосовувався автоматизований скрипт, що надсилав запити, параметри яких відповідали попередньо збереженій сигнатурі відомої атаки. Для мінімізації впливу фонових коливань під час вимірювання пропускної здатності та затримок легітимних запитів кожен тестовий запуск тривав 15 секунд і повторювався тричі, а як результуюче значення використовувалася медіана.

Час реакції розробленої системи на DDoS-атаку визначався як часовий інтервал від моменту ініціалізації DDoS-атаки до отримання першої відповіді від вхідного шлюзу з кодом HTTP 403. Для кожного з досліджуваних сценаріїв оброблення трафіку виконано по 10 успішних тестових запусків. Отримані результати вимірювань у вигляді середнього арифметичного значення, а також 50-го (p50, медіана) та 90-го (p90) перцентилів наведено в табл. 1.

Таблиця 1

Час реакції системи на DDoS-атаку для різних сценаріїв оброблення трафіку

Час на DDoS-атаку, с / Сценарій реагування	Full cold path	Fast-track cold path	Hot path
Середнє значення	3,53	1,53	0,005641
p50	3,15	0,8	0,005674
p90	5,22	3,33	0,006576

Аналіз отриманих даних в табл. 1 свідчить, що сценарій "повного холодного шляху" демонструє найбільший час реакції, оскільки перед застосуванням блокування система здійснює ресурсоємну класифікацію раніше невідомої клієнтської сесії. Натомість застосування сценарію "прискореного холодного шляху" дозволяє виключити цей етап завдяки збігу обчисленої сигнатури із базою відомих сигнатур, що забезпечує зниження медіанного часу реакції (p50) майже в 4 рази – до 0,8 с. Мінімальна затримка очікувано спостерігається під час застосування "гарячого шляху", де блокування зловмисного джерела відбувається безпосередньо на рівні вхідного шлюзу без залучення модуля класифікації, забезпечуючи медіанний час блокування запиту на рівні 5,674 мс.

Окремо досліджувалися затримка, пропускна здатність та обчислювальні витрати компонентів Data Plane та модуля класифікації. Для оцінки роботи Istio Ingress Gateway

ISSN 1562-9945 (Print)
ISSN 2707-7977 (Online)

виконано порівняння оброблення ідентичних HTTP-запитів у двох станах: без EnvoyFilter та з увімкненим розширенням EnvoyFilter для експорту атрибутів. Щоб виключити додаткові затримки цільового вебсервісу на результати вимірювань та уникнути пересилання трафіку вглиб кластера, на шлюзі налаштовано генерацію локальної відповіді HTTP 200 для URI /benchmark/local. Завдяки цьому виміряний час відгуку відображає виключно затримку оброблення запиту на рівні шлюзу (зокрема, витрати на виконання EnvoyFilter). Під час експерименту контролер DDoS Protection Operator залишався неактивним, а ресурс AuthorizationPolicy містив адресу-заглушку, що гарантувало проходження всіх запитів без залучення механізмів блокування. Навантаження на шлюз генерувалося утилітою Fortio (сеансами по 15 с): за фіксованої інтенсивності 500 запитів за секунду фіксувалася медіанна затримка, а за максимальної інтенсивності – гранична пропускна здатність. Час виконання та споживання оперативної пам'яті модулем класифікації оцінювалися за допомогою окремого тесту на вибірці з 4000 клієнтських сесій. Результати цих вимірювань наведено в табл. 2.

Таблиця 2

Оцінка накладних витрат шлюзу та модуля класифікації

Метрика	Значення
Медіанна затримка оброблення запиту на шлюзі (без EnvoyFilter), мс	1,71
Медіанна затримка оброблення запиту на шлюзі (з EnvoyFilter), мс	4,643
Максимальна пропускна здатність шлюзу (без EnvoyFilter), запитів/с	6063
Максимальна пропускна здатність шлюзу (з EnvoyFilter), запитів/с	2158
Час класифікації 4000 сесій у модулі класифікації T_{class}, мс	32,5
Обсяг оперативної пам'яті для класифікації 4000 сесій, МБ	3,09

Наведені дані в табл. 2 демонструють обчислювальні накладні витрати на експорт атрибутів на рівні шлюзу та безпосередню класифікацію клієнтських сесій. На розгорнутому тестовому кластері активація EnvoyFilter призвела до збільшення медіанної затримки на 2,933 мс та зниження пропускної здатності шлюзу на 64,4 %. Важливо зазначити, що ці показники відображають виключно витрати на виокремлення та асинхронне відправлення HTTP-атрибутів до модуля збору даних та не включають час на оброблення даних кінцевим вебзастосунком. Значення 32,5 мс та 3,09 МБ характеризують одноразове виконання класифікації для набору даних із 4000 сесій усередині процесу модуля класифікації, а не оброблення однієї сесії чи споживання пам'яті контейнером загалом. Час роботи реалізованих методів виявлення DDoS-атак у модулі класифікації T_{class} для масиву з 4000 сесій становить 32,5 мс. Отримані значення часу та використання оперативної пам'яті (лише 3,09 МБ) є незначними для такого обсягу даних, що підтверджує високу обчислювальну ефективність рішення та мінімізує його навантаження на ресурси середовища Kubernetes.

Для здійснення порівняльного аналізу розробленої системи з наявними аналогами на тому самому вхідному шлюзі розгорнуто рішення Coraza – WAF (Web Application Firewall), що сумісний із ModSecurity та містить повний набір правил OWASP Core Rule

Set (CRS), реалізований як WASM-розширення для проксі-сервера Envoy [36]. Експеримент передбачав дослідження двох альтернативних сценаріїв оброблення запитів: запропонованої системи з асинхронним експортом атрибутів через EnvoyFilter та програми Coraza із синхронною перевіркою правил (без використання EnvoyFilter). Обидва варіанти тестувалися на локальному URI /benchmark/local в шлюзі, що повністю виключало вплив цільового застосунку. Як і в попередніх дослідженнях, кожне вимірювання тривало 15 с і повторювалося тричі, а за підсумковий показник бралося медіанне значення. Важливо зазначити, що медіанну затримку визначали за зниженої інтенсивності навантаження, а саме 10 запитів за секунду. Це зумовлено тим, що при 500 запитів за секунду Coraza вичерпує обчислювальні ресурси, через що до основного часу оброблення додається затримка очікування запитів у черзі, що спотворило б результати порівняння. Водночас гранична пропускна здатність вимірювалася окремо в умовах максимального навантаження. Результати порівняльного оцінювання наведено в табл. 3.

Таблиця 3

Порівняння затримки та пропускної здатності розробленої системи з Coraza WAF

Система захисту	Медіанна затримка оброблення запиту, мс	Максимальна пропускна здатність, запитів/с
Розроблена система	3,667	1973
Coraza WAF	45	473

Наведені в табл. 3 дані демонструють порівняння ключових показників обох рішень за однакових умов експерименту. Відмінність абсолютних значень затримки та пропускної здатності для розробленої системи порівняно з результатами, поданими в табл. 2, пояснюється виключно зниженою інтенсивністю тестового навантаження (10 запитів/с замість 500 запитів/с). Аналіз даних показує, що порівняно із запропонованим рішенням застосування Coraza WAF збільшує медіанну затримку на 41,333 мс (у 12,3 раза) та знижує граничну пропускну здатність на 76 %. Отже, за умов проведеного експерименту синхронна перевірка трафіку з повним набором правил CRS генерує суттєво більші накладні витрати на рівні вхідного шлюзу, ніж розроблений архітектурний підхід на базі асинхронного експорту атрибутів.

Під час аналізу отриманих результатів необхідно враховувати низку обмежень проведених експериментів. По-перше, вимірювання виконувалися на тестовому кластері з одним вузлом за помірної інтенсивності трафіку, тому в розподілених середовищах з декількома вузлами під час масивних атак показники мережевих затримок можуть відрізнятися. По-друге, обчислювальна складова T_{class} безпосередньо залежить від застосованих методів виявлення атак. Оцінка їхньої точності не входила в межі цього дослідження, проте впровадження складніших методів неминуче збільшить час класифікації. Крім того, зі збільшенням кількості виявлених зловмисників зростатиме обсяг списку блокування в ресурсі AuthorizationPolicy, що підвищить споживання оператив-

ISSN 1562-9945 (Print)
ISSN 2707-7977 (Online)

ної пам’яті на шлюзі. Однак цей вплив може ефективно нівелюватися механізмом TTL, який забезпечує автоматичне очищення застарілих записів блокування.

Окремо слід відзначити архітектурну специфіку порівняння розробленого рішення з рішеннями класу WAF (зокрема Coraza). Зазвичай WAF виконує миттєву синхронну перевірку за статичними правилами, що створює значні накладні витрати на кожен запит і потребує ручного налаштування під нові вектори атак. Натомість запропонована система виносить складну логіку класифікації LR-DDoS (яка потребує тривалого вікна спостереження) в окремий асинхронний конвеєр. Через це повний цикл первинного виявлення атаки (cold path) триває довше, ніж перевірка вже наявного правила у WAF. Проте загальна ефективність системи ґрунтується на автоматичній класифікації та оптимізованому способу оброблення запитів: після першої класифікації переважна більшість шкідливого трафіку блокується миттєво безпосередньо на шлюзі (hot path), що робить таку архітектуру оптимальною для захисту застосунків. У подальших дослідженнях планується розширити експериментальну базу для тестування системи в масштабованих середовищах.

Висновки. У статті запропоновано та реалізовано розподілену архітектуру захисту мікросервісів від HTTP LR-DDoS атак у середовищі Kubernetes, яка не потребує модифікації вихідного коду цільових застосунків. Запропоноване архітектурне рішення передбачає виокремлення атрибутів HTTP-запитів після дешифрування TLS-трафіку у вхідному шлюзі Istio Ingress Gateway, їхню асинхронну класифікацію поза основним потоком оброблення та блокування виявлених джерел атак за допомогою Istio-ресурсу AuthorizationPolicy.

Головною особливістю розробленої архітектури є поєднання асинхронного експорту HTTP-атрибутів, збору мережевих атрибутів за допомогою технологій eBPF/XDP та реалізації трьох сценаріїв реагування системи на DDoS-атаки. Завдяки цьому неідентифікований трафік підлягає аналізу за допомогою реалізованих методів класифікації, запити з відомими сигнатурами розпізнаються без повторних обчислень, а HTTP-запити від уже заблокованих джерел атак відхиляються безпосередньо вхідним шлюзом зі сталою обчислювальною складністю $O(1)$.

Результати проведених експериментальних досліджень підтвердили досягнення поставленої мети та виконання всіх встановлених вимог. Застосування розширення EnvoyFilter збільшило затримку оброблення запитів лише на 2,933 мс, що повністю відповідає встановленій вимозі (не більше 10 мс) та не створює відчутних затримок під час маршрутизації легітимного трафіку. Час реагування системи на атаку склав 3,15 с для сценарію “full cold path” (класифікація нових джерел запитів), 0,8 с для “fast-track cold path” (розпізнавання відомих сигнатур) та 5,674 мс для “hot path” (блокування на рівні шлюзу). Модуль класифікації виконав аналіз 4000 сесій за 32,5 мс із використанням 3,09 МБ оперативної пам’яті. У порівнянні з Coraza WAF розроблена система продемонструвала у 12,3 раза меншу медіанну затримку (3,667 мс проти 45 мс) та вищу граничну пропускну здатність (1973 запити за секунду проти 473 запитів за секунду).

Отримані експериментальні дані отримано на тестовому кластері з одним вузлом, тому вони відображають особливості розробленого рішення, а не параметри його роботи у повноцінному промисловому середовищі. Напрямами подальших досліджень є тестування системи у розподіленій конфігурації з багатьма вузлами, налаштування динамічного масштабування класифікатора на основі розміру черги запитів у Kafka та перевірка ефективності функціонування всієї системи на наборах реального мережевого трафіку.

ЛІТЕРАТУРА

1. Yoachimik O., Pacheco J. 2025 Q4 DDoS threat report: A record-setting 31.4 Tbps attack caps a year of massive DDoS assaults. *The Cloudflare Blog*.
URL: https://blog.cloudflare.com/ddos-threat-report-2025-q4/ (date of access: 16.04.2026).

2. Singh Samra R., Barcellos M. DDoS2Vec: flow-level characterisation of volumetric DDoS attacks at scale. *Proceedings of the ACM on networking*. 2023. Vol. 1, CoNEXT3. 13. DOI: 10.1145/3629135.

3. Sabri S., Ismail N., Hazzim A. Slowloris DoS attack based simulation. *IOP conference series: materials science and engineering*. 2021. Vol. 1062, no 1. 012029.
DOI: 10.1088/1757-899x/1062/1/012029.

4. Hands-on denial of service lab exercises using SlowLoris and RUDY / E. Damon et al. *The 2012 information security curriculum development conference*, Kennesaw, Georgia, 12–13 October 2012. P. 21–29. DOI: 10.1145/2390317.2390321.

5. Shafieian S., Zulkernine M., Haque A. CloudZombie: launching and detecting slow-read distributed denial of service attacks from the cloud. *2015 IEEE international conference on computer and information technology; ubiquitous computing and communications; dependable, autonomic and secure computing; pervasive intelligence and computing*, Liverpool, 26–28 October 2015. P. 1733–1740. DOI: 10.1109/cit/iucc/dasc/picom.2015.261.

6. Low-Rate DoS attacks, detection, defense, and challenges: a survey / W. Zhijun et al. *IEEE access*. 2020. Vol. 8. P. 43920–43943. DOI: 10.1109/access.2020.2976609.

7. Tripathi N., Hubballi N. Application layer denial-of-service attacks and defense mechanisms. *ACM computing surveys*. 2021. Vol. 54, no. 4. 86. DOI: 10.1145/3448291.

8. Detection and mitigation of low-rate denial-of-service attacks: a survey / V. D. M. Rios et al. *IEEE access*. 2022. Vol. 10. P. 76648–76668. DOI: 10.1109/access.2022.3191430.

9. Velepucha V., Flores P. A survey on microservices architecture: principles, patterns and migration challenges. *IEEE access*. 2023. Vol. 11. P. 88339–88358.
DOI: 10.1109/access.2023.3305687.

10. Vasireddy I., Ramya G., Kandi P. Kubernetes and docker load balancing: state-of-the-art techniques and challenges. *International journal of innovative research in engineering and management*. 2023. Vol. 10, no. 6. P. 49–54. DOI: 10.55524/ijirem.2023.10.6.7.

11. Deressu T. F., Beyene A. M., Gemechu A. Y. A survey of application layer distributed denial of service detection: approaches, models, and datasets. *Artificial intelligence review*. 2026. Vol. 59, no. 6. 139. DOI: 10.1007/s10462-026-11528-3.

ISSN 1562-9945 (Print)
ISSN 2707-7977 (Online)

12. Tian Q., Miyata S. A DDoS attack detection method using conditional entropy based on SDN traffic. *IoT*. 2023. Vol. 4, no. 2. P. 95–111. DOI: 10.3390/iot4020006.
13. Qasim S. S., Nsaif S. M. Advancements in time series-based detection systems for distributed denial-of-service (DDoS) attacks: a comprehensive review. *Babylonian journal of networking*. 2024. Vol. 2024. P. 9–17. DOI: 10.58496/bjn/2024/002.
14. Detection of slow DDoS attacks based on user's behavior forecasting / V. Savchenko et al. *International journal of emerging trends in engineering research*. 2020. Vol. 8, no. 5. P. 2019–2025. DOI: 10.30534/ijeter/2020/90852020.
15. A survey of low rate DDoS detection techniques based on machine learning in software-defined networks / A. A. Alashhab et al. *Symmetry*. 2022. Vol. 14, no. 8. 1563.
DOI: 10.3390/sym14081563.
16. Ilango H. S., Ma M., Su R. A feedforward–convolutional neural network to detect low-rate DoS in IoT. *Engineering applications of artificial intelligence*. 2022. Vol. 114. 105059. DOI: 10.1016/j.engappai.2022.105059.
17. Онай М. В., Кравчук А. А. Вдосконалення та аналіз ефективності методів виявлення DDoS-атак на основі протоколу HTTP. *Прикладна математика та комп'ютинг ПМК' 2024* : зб. тез доп. Сімнадцятої конф. магістрантів та аспірантів, м. Київ, 20–22 листоп. 2024 р. С. 149–153. URL: https://ela.kpi.ua/handle/123456789/71110.
18. Web application firewall for detecting and mitigation of based DDoS attacks using machine learning and blockchain / E. Leka et al. *TEM journal*. 2024. Vol. 13, no. 4. P. 2802–2811. DOI: 10.18421/tem134-17.
19. Dawadi B. R., Adhikari B., Srivastava D. K. Deep learning technique-enabled web application firewall for the detection of web attacks. *Sensors*. 2023. Vol. 23, no. 4. 2073.
DOI: 10.3390/s23042073.
20. OWASP ModSecurity Core Rule Set (CRS): project documentation. *OWASP Foundation*. URL: https://coreruleset.org/docs/ (date of access: 20.04.2026).
21. Alasri A., Sulaiman R. Protection of XML-based denail-of-service and HTTP flooding attacks in web services using the middleware tool. *International journal of engineering and technology*. 2018. Vol. 7, no. 4.7. P. 322–329. DOI: 10.14419/ijet.v7i4.7.20570.
22. Berezhna M. Separation of concerns: a fundamental principle in software development. *DEV Community*. URL: https://dev.to/marie_berezhna/separation-of-concerns-a-fundamental-principle-in-software-development-oaj (date of access: 20.04.2026).
23. Detecting distributed denial-of-service (DDoS) attacks through the log consolidation processing (LCP) framework / M. A. Khairuddin et al. *International journal on informatics visualization*. 2024. Vol. 8, no. 3. P. 1064–1073. DOI: 10.62527/joiv.8.3.2184.
24. Service mesh overview and comparison / A. Kammer et al. *INNOQ*.
URL: https://servicemesh.es/ (date of access: 28.04.2026).
25. Singh P., Ayyasamy S. Exploring, analyzing and tuning service mesh performance: a literature review. 2023. 10 p. (Preprint. TechRxiv). DOI: 10.36227/techrxiv.22776119.v1.
26. Priya Patharlagadda P. Kubernetes traffic management using Istio. *Journal of media & management*. 2022. Vol. 4, no. 1. DOI: 10.47363/jmm/2022(4)e101.
27. Architecture overview and what is Envoy – Envoy documentation. *Envoy proxy*.

URL: https://www.envoyproxy.io/docs/envoy/v1.38.2/intro/what_is_envoy (date of access: 30.04.2026).
28. Envoy Filter: traffic management in Istio – Istio documentation. *Istio*.
URL: https://istio.io/latest/docs/reference/config/networking/envoy-filter/ (date of access: 30.04.2026).
29. Kubernetes documentation: fundamental concepts. *The Kubernetes Authors*.
URL: https://kubernetes.io/docs/concepts/ (date of access: 04.05.2026).
30. Fast Packet Processing with eBPF and XDP / M. A. M. Vieira et al. *ACM computing surveys*. 2020. Vol. 53, no. 1. 16. DOI: 10.1145/3371038.
31. Kubernetes documentation: operator pattern. *The Kubernetes Authors*. URL: https://kubernetes.io/docs/concepts/extend-kubernetes/operator/ (date of access: 04.05.2026).
32. Kubernetes controller-runtime project – Go Packages. *The Go Project*. URL: https://pkg.go.dev/sigs.k8s.io/controller-runtime (date of access: 04.05.2026).
33. Reitz K. PostmanLabs/httpbin: HTTP request & response service, written in python + flask. *GitHub, Inc*. URL: https://github.com/postmanlabs/httpbin (date of access: 04.05.2026).
34. Fortio load testing library, command line tool, advanced echo server in go. *Fortio*.
URL: https://fortio.org/ (date of access: 04.05.2026).
35. SlowHttp – Slow Read DDoS simulator written as a Python script. *GitHub, Inc*.
URL: https://github.com/ojardila/slowhttp (date of access: 04.05.2026).
36. Coraza – open source, enterprise-grade, high performance web application firewall. *Coraza*. URL: https://www.coraza.io/docs/tutorials/introduction/ (date of access: 04.05.2026).

REFERENCES

1. Yoachimik, O., & Pacheco, J. (2026). 2025 Q4 DDoS threat report: A record-setting 31.4 Tbps attack caps a year of massive DDoS assaults. *The Cloudflare Blog*. https://blog.cloudflare.com/ddos-threat-report-2025-q4/
2. Singh Samra, R., & Barcellos, M. (2023). DDoS2Vec: Flow-level characterisation of volumetric DDoS attacks at scale. *Proceedings of the ACM on Networking*, 1(CoNEXT3), Article 13. https://doi.org/10.1145/3629135
3. Sabri, S., Ismail, N., & Hazzim, A. (2021). Slowloris DoS attack based simulation. *IOP Conference Series: Materials Science and Engineering*, 1062(1), Article 012029.
https://doi.org/10.1088/1757-899x/1062/1/012029
4. Damon, E., Dale, J., Laron, E., Mache, J., Land, N., & Weiss, R. (2012). Hands-on denial of service lab exercises using SlowLoris and RUDY. In *The 2012 information security curriculum development conference* (pp. 21–29). ACM Press.
https://doi.org/10.1145/2390317.2390321
5. Shafieian, S., Zulkernine, M., & Haque, A. (2015). CloudZombie: Launching and detecting slow-read distributed denial of service attacks from the cloud. In *2015 IEEE international conference on computer and information technology; ubiquitous computing and communications; dependable, autonomic and secure computing; pervasive intelligence and computing* (pp. 1733–1740). IEEE. https://doi.org/10.1109/cit/iucc/dasc/picom.2015.261

ISSN 1562-9945 (Print)
ISSN 2707-7977 (Online)

6. Zhijun, W., Wenjing, L., Liang, L., & Meng, Y. (2020). Low-Rate DoS attacks, detection, defense, and challenges: A survey. *IEEE Access*, *8*, 43920–43943. https://doi.org/10.1109/access.2020.2976609
7. Tripathi, N., & Hubballi, N. (2021). Application layer denial-of-service attacks and defense mechanisms. *ACM Computing Surveys*, *54*(4), Article 86. https://doi.org/10.1145/3448291
8. Rios, V. D. M., Inacio, P. R. M., Magoni, D., & Freire, M. M. (2022). Detection and mitigation of low-rate denial-of-service attacks: A survey. *IEEE Access*, *10*, 76648–76668. https://doi.org/10.1109/access.2022.3191430
9. Velepucha, V., & Flores, P. (2023). A survey on microservices architecture: Principles, patterns and migration challenges. *IEEE Access*, *11*, 88339–88358. https://doi.org/10.1109/access.2023.3305687
10. Vasireddy, I., Ramya, G., & Kandi, P. (2023). Kubernetes and docker load balancing: State-of-the-art techniques and challenges. *International Journal of Innovative Research in Engineering and Management*, *10*(6), 49–54. https://doi.org/10.55524/ijirem.2023.10.6.7
11. Deressu, T. F., Beyene, A. M., & Gemechu, A. Y. (2026). A survey of application layer distributed denial of service detection: Approaches, models, and datasets. *Artificial Intelligence Review*, *59*(6), Article 139. https://doi.org/10.1007/s10462-026-11528-3
12. Tian, Q., & Miyata, S. (2023). A DDoS attack detection method using conditional entropy based on SDN traffic. *IoT*, *4*(2), 95–111. https://doi.org/10.3390/iot4020006
13. Qasim, S. S., & Nsaif, S. M. (2024). Advancements in time series-based detection systems for distributed denial-of-service (DDoS) attacks: A comprehensive review. *Babylonian Journal of Networking*, *2024*, 9–17. https://doi.org/10.58496/bjn/2024/002
14. Savchenko, V., Ilin, O., Hnidenko, N., Tkachenko, O., Laptiev, O., & Lehominova, S. (2020). Detection of slow DDoS attacks based on user's behavior forecasting. *International Journal of Emerging Trends in Engineering Research*, *8*(5), 2019–2025. https://doi.org/10.30534/ijeter/2020/90852020
15. Alashhab, A. A., Zahid, M. S. M., Azim, M. A., Daha, M. Y., Isyaku, B., & Ali, S. (2022). A survey of low rate DDoS detection techniques based on machine learning in software-defined networks. *Symmetry*, *14*(8), Article 1563. https://doi.org/10.3390/sym14081563
16. Ilango, H. S., Ma, M., & Su, R. (2022). A feedforward–convolutional neural network to detect low-rate DoS in IoT. *Engineering Applications of Artificial Intelligence*, *114*, Article 105059. https://doi.org/10.1016/j.engappai.2022.105059
17. Onai, M. V., & Kravchuk, A. A. (2024). Vdoskonalennia ta analiz efektyvnosti metodiv vyiavlennia DDoS-atak na osnovi protokolu HTTP. In *Prykladna matematyka ta kompiutynh PMK' 2024* (pp. 149–153). https://ela.kpi.ua/handle/123456789/71110
18. Leka, E., Lamani, L., Aliti, A., & Hoxha, E. (2024). Web application firewall for detecting and mitigation of based DDoS attacks using machine learning and blockchain. *TEM Journal*, *13*(4), 2802–2811. https://doi.org/10.18421/tem134-17
19. Dawadi, B. R., Adhikari, B., & Srivastava, D. K. (2023). Deep learning technique-enabled web application firewall for the detection of web attacks. *Sensors*, *23*(4), Article 2073. https://doi.org/10.3390/s23042073

20. OWASP ModSecurity Core Rule Set (CRS): Project Documentation. (2026). *OWASP Foundation*. https://coreruleset.org/docs/
21. Alasri, A., & Sulaiman, R. (2018). Protection of XML-based denail-of-service and HTTP flooding attacks in web services using the middleware tool. *International Journal of Engineering and Technology*, *7*(4.7), 322–329. https://doi.org/10.14419/ijet.v7i4.7.20570
22. Berezhna, M. (2025). Separation of concerns: A fundamental principle in software development. *DEV Community*. https://dev.to/marie_berezhna/separation-of-concerns-a-fundamental-principle-in-software-development-oaj
23. Khairuddin, M. A., Mohd Isa, M. R., Mohd Shukran, M. A., Ismail, M. N., & Maskat, K. (2024). Detecting distributed denial-of-service (DDoS) attacks through the log consolidation processing (LCP) framework. *International Journal on Informatics Visualization*, *8*(3), 1064–1073. https://doi.org/10.62527/joiv.8.3.2184
24. Kammer, A., Koppelt, C., Müller, J., Prinz, H., Schmidt, C., Steinke, M., & Wolff, E. (2024). Service mesh overview and comparison. *INNOQ*. https://servicemesh.es/
25. Singh, P., & Ayyasamy, S. (2023). Exploring, analyzing and tuning service mesh performance: A literature review. *TechRxiv*. https://doi.org/10.36227/techrxiv.22776119.v1
26. Priya Patharlagadda, P. (2022). Kubernetes traffic management using Istio. *Journal of Media & Management, 4*(1). https://doi.org/10.47363/jmm/2022(4)e101
27. Architecture overview and what is Envoy – Envoy documentation. (2026). *Envoy proxy*. https://www.envoyproxy.io/docs/envoy/v1.38.2/intro/what_is_envoy
28. Envoy Filter: Traffic management in Istio – Istio documentation. (2026). *Istio*. https://istio.io/latest/docs/reference/config/networking/envoy-filter/
29. Kubernetes documentation: Fundamental concepts. (2026). *The Kubernetes Authors*. https://kubernetes.io/docs/concepts/
30. Vieira, M. A. M., Castanho, M. S., Pacífico, R. D. G., Santos, E. R. S., Júnior, E. P. M. C., & Vieira, L. F. M. (2020). Fast Packet Processing with eBPF and XDP. *ACM Computing Surveys*, *53*(1), Article 16. https://doi.org/10.1145/3371038
31. Kubernetes documentation: Operator pattern. (2025). *The Kubernetes Authors*. https://kubernetes.io/docs/concepts/extend-kubernetes/operator/
32. Kubernetes controller-runtime project – Go Packages. (2026). *The Go Project*. https://pkg.go.dev/sigs.k8s.io/controller-runtime
33. Reitz, K. (2026). PostmanLabs/httpbin: HTTP request & response service, written in python + flask. *GitHub, Inc*. https://github.com/postmanlabs/httpbin
34. Fortio load testing library, command line tool, advanced echo server in go. (2026). *Fortio*. https://fortio.org/
35. SlowHttp – Slow Read DDoS simulator written as a Python script. (2026). *GitHub, Inc*. https://github.com/ojardila/slowhttp
36. Coraza – open source, enterprise-grade, high performance web application firewall. (2026). *Coraza*. https://www.coraza.io/docs/tutorials/introduction/

Received 29.06.2026.
Accepted 31.08.2026.
Published 30.10.2026

ISSN 1562-9945 (Print)
ISSN 2707-7977 (Online)

Designing the architecture of a software system for detecting and blocking HTTP LR-DDoS attacks on the Kubernetes platform

This paper proposes and implements a distributed architecture for protecting microservices against application-layer low-rate DDoS (LR-DDoS) attacks in a Kubernetes environment, without modifying the target applications' code and without introducing noticeable latency to legitimate traffic. The architectural solution is based on separating the routing plane (Data Plane) and the analysis plane (Control Plane): application-layer attributes are collected by the EnvoyFilter extension after TLS decryption at the Istio ingress gateway, and network attributes are gathered using eBPF/XDP technologies, after which they are asynchronously forwarded to the Apache Kafka broker. The horizontally scalable classification module, developed in Go, and the controller based on the Kubernetes Operator pattern store DDoS attack sources in Redis with a TTL mechanism to remove stale data and update blocking rules in the AuthorizationPolicy resource. This enables three attack mitigation scenarios: full classification of unknown sessions, accelerated verification against preserved signatures, and request blocking at the gateway with O(1) complexity. Experimental studies have confirmed the system's high performance: the latency overhead from the EnvoyFilter is only 2.933 ms, and the response time is 3.15 s for new sources, 0.8 s for known signatures, and 5.674 ms for gateway-level blocking. The classification module processes 4,000 sessions in 32.5 ms using 3.09 MB of RAM. Compared to the Coraza Web Application Firewall (WAF), the proposed system demonstrates 12.3 times lower latency (3.667 ms versus 45 ms) and significantly higher throughput (1,973 versus 473 requests/s).

Keywords: DDoS, low-rate DDoS attacks, LR-DDoS, software system, software engineering, microservice architecture, Kubernetes, Istio, information security, cryptography.

Кравчук Аркадій Андрійович – аспірант кафедри програмного забезпечення комп'ютерних систем, Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського», Київ, Україна.
ORCID: https://orcid.org/0000-0002-6128-206X
Онай Микола Володимирович – к.т.н., доцент, доцент кафедри програмного забезпечення комп'ютерних систем, Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського», Київ, Україна.
ORCID: https://orcid.org/0000-0002-4938-8355

Kravchuk Arkadii – PhD Student at the Department of Computer System Software, National Technical University of Ukraine "Igor Sikorsky Kyiv Polytechnic Institute", Kyiv, Ukraine.
ORCID: https://orcid.org/0000-0002-6128-206X
Onai Mykola – PhD in Computer Systems and Components, Associate Professor, Associate Professor at the Department of Computer System Software, National Technical University of Ukraine "Igor Sikorsky Kyiv Polytechnic Institute", Kyiv, Ukraine.
ORCID: https://orcid.org/0000-0002-4938-8355