КІБЕРБЕЗПЕКА: освіта, наука, техніка
CYBERSECURITY: EDUCATION, SCIENCE, TECHNIQUE
№ 1 (33), 2026
ISSN 2663 – 4023
DOI 10.28925/2663-4023.2026.33.1257
УДК 004.056.5

Молнар Віталій Васильович
аспірант кафедри захисту інформації
Національний Університет «Львівська Політехніка», Львів, Україна
ORCID: 0009-0001-3183-0117
vitalii.v.molnar@lpnu.ua

Сабодашко Дмитро Володимирович
доктор філософії, старший викладач кафедри захисту інформації
Національний Університет «Львівська Політехніка», Львів, Україна
ORCID: 0000-0003-1675-0976
dmytro.v.sabodashko@lpnu.ua

ДЕТЕРМІНОВАНЕ СТРИМУВАННЯ ПОШИРЕННЯ ПОДІЙ У ПОДІЙНО-КЕРОВАНИХ БЕЗСЕРВЕРНИХ СИСТЕМАХ

Анотація. Подійно-керовані безсерверні системи забезпечують гнучке масштабування хмарних застосунків, але водночас можуть породжувати надмірне поширення подій, коли невелика кількість вхідних повідомлень спричиняє непропорційно великий обсяг подальших виконань через розгалуження, повторні спроби або циклічну обробку. Таке явище є важливим з погляду кібербезпеки, оскільки збільшує операційний радіус ураження, підвищує ризики доступності та може призводити до фінансового виснаження в середовищах з оплатою за фактичне використання. У статті проаналізовано проблему обмеженості наявних платформових механізмів, зокрема повторних спроб, черг недоставлених повідомлень і обмеження одночасних виконань, які переважно діють після початку поширення або лише регулюють його темп. Метою роботи є експериментальна оцінка того, чи може мінімальна детермінована точка вхідного контролю зменшувати подальше поширення подій у безсерверному конвеєрі. Методика дослідження ґрунтується на відтворюваному A/B-експерименті в AWS із використанням AWS Lambda та Amazon SQS. Порівнюються базова архітектура, у якій події безпосередньо надходять до основної черги, та захищена архітектура з вхідним шлюзом оцінювання ризику і маршрутизацією частини подій до карантинної черги. Для оцінювання використано коефіцієнт ампліфікації AF, зміни стану карантинної черги та DLQ, кількість викликів функцій і час стабілізації конвеєра. Результати показали, що за нормального навантаження обидві архітектури зберігають поведінку «один до одного» (AF = 1,0), тоді як отруйні повідомлення спричиняють збільшення виконань через повторні спроби з подальшою ізоляцією в DLQ. Найбільший ефект вхідного контролю зафіксовано для навантаження з розгалуженням: у базовому режимі AF становив 16,6, за повного блокування зменшувався до 0,0, а в селективній конфігурації за $\tau = 0{,}25$ знижувався до 4,28 лише за рахунок шлюзу та до 3,0 у поєднанні з обмеженням розгалуження на рівні обробника. Для циклічного профілю AF залишався рівним 5,0, що вказує на обмежене поширення та іншу семантику карантину як термінального маркування. Отримані результати свідчать, що детермінований вхідний контроль може суттєво зменшувати радіус ураження для високоризикових шаблонів поширення подій без використання навчених моделей. Водночас його застосування потребує довіреного периметра виробників подій і подальшої перевірки на змішаних реалістичних навантаженнях.

Ключові слова: безсерверна безпека; подійно-керовані архітектури; AWS Lambda; Amazon SQS; ампліфікація подій; радіус ураження; карантинна маршрутизація.

ВСТУП

Подійно-керовані безсерверні архітектури стали одним з основних проєктних підходів до побудови масштабованих та економічно ефективних хмарних застосунків. Завдяки розділенню виробників і споживачів подій через асинхронний обмін повідомленнями такі системи підвищують гнучкість масштабування та спрощують операційне обслуговування. Однак ті самі властивості, які забезпечують

масштабованість, водночас породжують специфічний системний ризик – ампліфікацію подій, коли одна вхідна подія спричиняє непропорційно велику кількість подальших виконань через ланцюги розгалуження, повторні спроби або циклічні шаблони поширення. На практиці ампліфікація подій є не лише питанням надійності, а й безпековим ризиком, оскільки її можна використати для зниження доступності та породження ефекту фінансового виснаження через штучне нарощування обсягу обчислень, операцій у чергах і викликів функцій. Як наслідок, обмеження поширення подій стає одним з ключових прикладних завдань під час забезпечення безпеки подійно-керованих безсерверних систем.

Постановка проблеми. Наявні платформові механізми, зокрема політики повторних спроб, черги недоставлених повідомлень (DLQ) та обмеження одночасних виконань, забезпечують лише часткове пом'якшення цього ризику. DLQ та політики повторних спроб за своєю природою є реактивними: вони обмежують вплив окремих невдалих повідомлень уже після поширення, але не запобігають тому, щоб зовні нешкідливі події розгорталися у значний обсяг подальших виконань через розгалуження ще до настання збою. Крім того, зарезервована одночасність працює синхронно на рівні точки входу обробника: вона здатна обмежувати миттєву пропускну спроможність, проте не має необхідної деталізації для розрізнення низько- та високоризикових подій. До того ж цей механізм не зменшує сукупного обсягу подій, що підлягають обробленню, а лише регулює темп їхнього виконання. Зазначені обмеження обумовлюють потребу в експериментальному дослідженні того, чи можна квантифікувати ампліфікацію подій у відтворюваний спосіб і чи можна її стримати за допомогою точки вхідного контролю на основі політик.

Аналіз останніх досліджень і публікацій. Безсерверні обчислення суттєво змінили підхід до проєктування хмарно-орієнтованих застосунків, уможлививши дрібнозернисте, подійно-кероване виконання без потреби в постійному управлінні серверами; разом з тим ця зміна формує специфічну позицію в питаннях безпеки, що відрізняється від традиційних монолітних або довготривалих сервісних розгортань [1]. Нещодавній систематичний огляд механізмів безпеки безсерверних обчислень показує, що дослідницька галузь рухається у бік багатошарового мислення в безпеці, де середовищний захист, керування на рівні оркестрації та засоби, чутливі до подій, дедалі частіше розглядаються як окремі, але взаємозалежні домени захисту [2]. З погляду управління та керування ризиками такі моделі відмов та зловживань, зумовлені поширенням, узгоджуються із загальнішою потребою керувати кіберризиками та зменшувати їх у хмарно-орієнтованих системах [3].

Це розрізнення є важливим для подійно-керованих архітектур, оскільки безпекові наслідки роботи системи більше не визначаються лише захистом окремих функцій – вони залежать також від семантики поширення, яку формують тригери, черги та ланцюги викликів [1]. Як наслідок, аналіз безпеки безсерверних систем має враховувати не лише пряму компрометацію окремих функцій, а й непрямі шляхи ампліфікації, через які зовні правомірні події можуть розгортатися у непропорційно великий обсяг подальшої активності [2]. Один із основних напрямів досліджень останніх років зосереджено на економічному впливі та впливі на доступність зловмисних шаблонів виклику у безсерверних середовищах з оплатою за фактичне використання [4]. Концепція «відмови гаманця» (Denial of Wallet) формалізує ідею, що для завдання істотної шкоди зловмиснику не обов'язково повністю припиняти обслуговування, оскільки головним механізмом збитку може стати примусове надмірне виконання, яке призводить до завищеного рахунку та надмірного споживання ресурсів. У подійно-керованих конвеєрах цей ризик посилюється механізмами поширення – розгалуженням, повторними спробами та ланцюжковим обробленням, – які здатні перетворити невелику множину шкідливих або некоректно сформованих вхідних подій на непропорційно великий обсяг подальших виконань. Xiong та співавтори [5] продовжили цю лінію досліджень, показавши, що специфічна для безсерверних систем поведінка відмови в обслуговуванні може виникати з характеристик платформи, що значною мірою непомітні на рівні застосунку; це підкреслює потребу в засобах контролю, які діють ще до того, як поширення стане некерованим. Shen та співавтори [6] запропонували внутрішнє виявлення «відмови гаманця» для безсерверних платформ і показали, що легковагові методи на основі моніторингу здатні виявляти зловживання без надмірних накладних витрат. Kelly та співавтори [7] далі розвинули цей напрям, класифікувавши трафік атак на гаманець на рівні застосункового трафіку та продемонструвавши, що категорії атак у безсерверних системах можна виокремити з легітимних потоків запитів у контрольованих умовах.

Нещодавній огляд атак типу «відмова гаманця» консолідував ці результати та підкреслив, що галузі досі бракує достатньо стандартизованих експериментальних методологій для порівняння впливу атак та ефективності засобів пом'якшення для різних шаблонів поширення [8]. Цей пробіл є особливо важливим для досліджень, орієнтованих на ампліфікацію подій, оскільки значна частина наявних робіт описує зловживання у термінах витрат або позначок аномалій, тоді як значно менше досліджень

оцінюють, скільки саме подальшої активності фактично породжує задана подія та яку частину цієї активності можна запобігти за допомогою явного стримування [8].

Паралельно низка робіт досліджувала виявлення безпекових інцидентів, спостережуваність, реагування на інциденти та моніторинг ланцюгів безсерверних задач [9-13]. Ці дослідження підтверджують, що потоки подій, виклики функцій і провайдерські сигнали моніторингу можуть бути корисними об'єктами безпекового аналізу. Водночас вони переважно зосереджені на виявленні аномалій, контролі цілісності або реагуванні після події, тоді як питання кількісного вимірювання подальшого поширення подій та його раннього детермінованого стримування залишається менш дослідженим. Іншою важливою лінією досліджень є вивчення безпеки через структуру взаємодій у безсерверних системах, а не через ізольовану поведінку окремих функцій. Polinsky та співавтори [14] моделюють поведінку авторизації у безсерверних системах як задачу досяжності у графі та показують, що посилення політик можна сформулювати як зменшення непередбаченої досяжності у графах політик безпеки. Це безпосередньо стосується концепції радіуса ураження у подійно-керованих системах, оскільки подальше поширення можна також розуміти як явище досяжності, коли одна допущена подія може уможливити цілу множину наступних переходів, побічних ефектів та активацій функцій. На ширшому архітектурному рівні Lazzari та Farias [15] звертають увагу на брак достатніх емпіричних доказів того, як топології подій, стратегії моніторингу та архітектурна стабільність взаємодіють у сучасних подійно-орієнтованих системах.

Література з повторних спроб, автоматичних розмикачів (circuit breakers) та стратегій відновлення також становить важливе концептуальне підґрунтя для дослідження ампліфікації подій. Saleh Sedghpour та співавтори [16] показали, що конфігурації повторних спроб та автоматичних розмикачів здатні суттєво змінювати поведінку системи в умовах збою, у деяких випадках підвищуючи стійкість, а в інших – спричиняючи неочікувані побічні ефекти залежно від налаштувань та характеристик навантаження. mmad [17] далі узагальнив повторні спроби, автоматичні розмикачі, протитиск (backpressure), ідемпотентність та спостережуваність як ключові механізми обмеження каскадних відмов у розподілених системах. Хоча ці роботи здебільшого розглядаються у площині інженерії стійкості, а не безпеки безсерверних систем, вони чітко підтверджують ідею, що стримування слід оцінювати як функцію керування поширенням, а не лише як функцію локального оброблення помилок.

Для частини дослідження, пов'язаної зі специфікою хмарної реалізації, релевантною є офіційна документація AWS, оскільки вона визначає семантику використаних компонентів. Lambda може опрацьовувати повідомлення з Amazon SQS через відображення джерел подій, DLQ слугує механізмом ізоляції повідомлень після невдалого оброблення, а CloudWatch надає метрики викликів, помилок, обмежень одночасності та тривалості виконання [18-20]. Дослідження вартості, тарифікації та продуктивності безсерверних систем додатково показують, що фінансовий і операційний вплив таких архітектур залежить від характеристик навантаження та режимів виконання [21-23]. Додатково дослідження каскадних відмов у мікросервісних системах показують, що ефекти поширення можна реконструювати за журналами, що підтримує акцент цієї роботи на вимірюванні поширення подій [24].

Загалом, нинішні дослідження формують міцну основу у сферах безсерверної безпеки, «відмови гаманця», виявлення аномалій, спостережуваності, досяжності в політиках та механізмів відновлення, однак залишається важливий пробіл на перетині вимірювання поширення подій та оцінювання детермінованого стримування [2]. Зокрема, у літературі представлено порівняно мало експериментально обґрунтованих досліджень, що ізолюють вплив мінімальної точки вхідного контролю на подальшу ампліфікацію за контрольованих шаблонів розгалуження, отруйних повідомлень і циклів у безсерверних конвеєрах на базі AWS [8]. Саме цю нішу заповнює представлене дослідження, у якому ампліфікація подій розглядається як кількісно вимірюванций прояв ризику поширення, а шлюз із політикою оцінювання ризику та карантинною маршрутизацією оцінюється з погляду його здатності зменшувати відповідний радіус ураження без використання навчених моделей.

Мета статті. Основним внеском роботи є відтворювана експериментальна методологія та підтвердження концепції, які показують, що механізми вхідного контролю на основі політик здатні зменшувати подальше поширення для контрольованих сценаріїв розгалуження в межах розгортання з довіреними виробниками подій. Стаття не претендує на створення універсального або готового до промислової експлуатації механізму стримування. Натомість вона пропонує експериментально обґрунтовану основу вимірювання ампліфікації подій та оцінює, як мінімальна точка контролю з оцінюванням ризику впливає на подальший радіус ураження за контрольованих профілів навантаження.

МЕТОДИКА ДОСЛІДЖЕННЯ

У цьому дослідженні стримування радіуса ураження у безсерверному подійно-керованому обробленні вивчається шляхом порівняння двох експериментально контрольованих архітектур: базового конвеєра, у якому події надходять безпосередньо до основної черги оброблення, та захищеного конвеєра, у якому до шляху приймання введено функцію оцінювання ризику. Шлюз ухвалює рішення про допуск для кожної події та маршрутизує її або до основної черги (допуск), або до карантинної черги (стримування/маркування), що дає змогу заздалегідь обмежувати поширення. У дослідженні використано відтворюваний випробувальний стенд на основі AWS та чотири профілі навантаження, що відображають типові шаблони ампліфікації: нормальне оброблення, отруйні повідомлення, які детерміновано спричиняють збій і запускають повторні спроби, каскади з розгалуженням, що розширюють подальше навантаження, та обмежені ланцюги циклів із контрольованою глибиною стрибків. Основною метрикою є коефіцієнт ампліфікації (AF), визначений як відношення загальної кількості опрацьованих подій до кількості вхідних подій у межах одного запуску; його доповнюють приріст довжини карантинної черги та DLQ, кількість викликів, а також час стабілізації конвеєра.

Системна модель та модель загроз. У роботі розглянуто подійно-керований безсерверний конвеєр оброблення, у якому вхідні повідомлення можуть спричиняти подальше поширення через пряме оброблення, повторні спроби, розгалуження або обмежену циклічну поведінку. У таких системах головна безпекова проблема не зводиться лише до невдачі окремого повідомлення; вона включає можливість того, що одна допущена подія спричинить непропорційну кількість подальших виконань, тим самим розширюючи операційний радіус ураження, погіршуючи доступність та збільшуючи споживання хмарних ресурсів [4]. Цей погляд узгоджується з нещодавніми дослідженнями, у яких зловживання безсерверними системами розглядається не лише як класична задача відмови в обслуговуванні, а й як проблема фінансового виснаження та керування поширенням у середовищах з оплатою за фактичне використання [4]. Відповідно, прийнята в роботі модель загроз зосереджена на структурно ризикових подіях, які можуть здаватися синтаксично прийнятними на вході, проте після допуску до конвеєра спричиняти шкідливе подальше розгортання [1]. Метою експерименту, отже, не є моделювання повністю адаптивного супротивника; натомість оцінюється, чи здатна точка вхідного контролю зменшувати вплив поширення ризикових подій до того, як ланцюг подій розгорнеться.

Слід чітко зафіксувати важливе обмеження області застосовності моделі загроз. Функція оцінювання ризику, описана в розділі «Політика шлюзу ризику», аналізує структурні поля метаданих, наявні в корисному навантаженні кожної події (eventType, payloadSizeBytes, fanoutDegree, hopCount). Така схема припускає, що ці поля справді відображають структурні властивості події, – припущення, яке виконується, коли виробники подій автентифіковані, а метадані є довіреними. Конкретно це відповідає архітектурам, у яких виробники для SQS працюють у межах довіреного периметра, обмеженого IAM, як-от внутрішні мікросервіси, автоматизовані конвеєри або контрольовані інфраструктурою генератори. Модель загроз свідомо обмежено саме таким середовищем з довіреними виробниками. Безпекові наслідки зловмисно сформованих метаданих (наприклад, події з розгалуженням, навмисно позначеної як = 0 або eventType = "NORMAL" для уникнення оцінювання) виходять за межі поточного дослідження і визначаються як важливий напрям майбутньої роботи. У середовищах, де ідентичність виробника гарантувати не можна, функцію оцінювання ризику необхідно доповнювати структурними сигналами, отриманими зі спостережуваної поведінки виконання, а не з полів корисного навантаження, заданих самим виробником подій.

Архітектура експериментального стенда (базова і захищена). Експериментальне середовище реалізовано як безсерверний конвеєр на базі AWS, побудований навколо асинхронного оброблення повідомлень. Базова архітектура становить пряму схему приймання подій, у якій генератор надсилає події до основної черги, а обробник споживає їх без проміжного етапу контролю допуску. Захищена архітектура розширює цю схему, додаючи на вході спеціальну функцію оцінювання ризику; кожна вхідна подія спочатку оцінюється, а потім маршрутизується або до основної черги оброблення, або до карантинної черги. Захищена архітектура впроваджує два механізми стримування, відсутні в базовому: вхідний шлюз ризику, який оцінює і маршрутизує кожну подію ще до її потрапляння в основну чергу, а у конфігурації повного багаторівневого захисту – також обмеження розгалуження на стороні обробника подіями. Обидва механізми оцінюються незалежно, щоб виокремити їхній відповідний внесок. Модель виконання, керованого чергою, у випробувальному стенді відповідає стандартній інтеграції AWS Lambda з Amazon SQS через відображення джерел подій [18]. Чергу недоставлених повідомлень збережено як окремий механізм ізоляції для повідомлень, які не вдається успішно опрацювати після кількох повторних

спроб; це дозволяє при оцінюванні чітко розділяти поведінку у разі отруйних повідомлень та поведінку карантину [19]. На рис. 1 показано обидві архітектури поряд для порівняння.

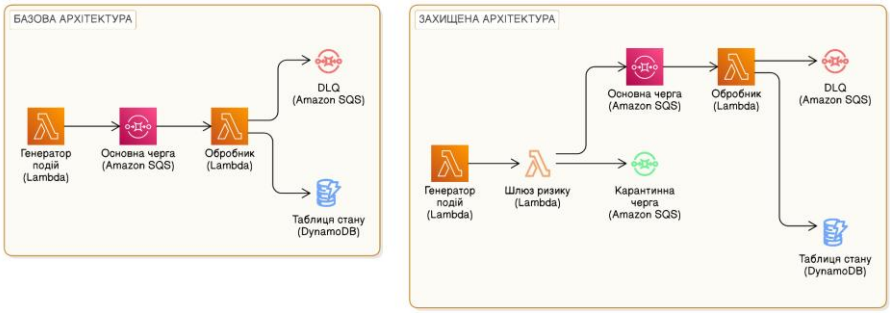

Рис. 1. Базова та захищена експериментальні архітектури.

Профілі навантаження. Щоб охопити різні режими поширення, в експерименті використано чотири контрольовані профілі навантаження. Профіль нормального навантаження представляє номінальне оброблення «один до одного», за якого кожна допущена вхідна подія має, як очікується, давати один успішний результат оброблення. Цей профіль слугує контрольним станом для перевірки того, чи зберігає захищена архітектура базову функціональну поведінку, попри додавання точки ухвалення рішення на вході.

Профіль отруйних повідомлень представляє детерміновано хибні вхідні дані, які запускають повторні спроби оброблення і зрештою можуть потрапити до черги недоставлених повідомлень. Цей профіль застосовано для спостереження за тим, як механізм оброблення відмов через повторні спроби поводиться в обох архітектурах і чи суттєво змінює вхідне оцінювання вплив сценаріїв збоїв на подальшу частину конвеєра.

Профіль розгалуження (fan-out) представляє основний сценарій ампліфікації в дослідженні. У цьому випадку одна вхідна подія породжує кілька подальших повідомлень, що дозволяє поширенню швидко зростати за відсутності стримування. Цей профіль є особливо важливим, оскільки безпосередньо відображає той тип ампліфікації подій, який у попередній літературі асоціюють зі сценаріями «відмови гаманця» та зловживань у безсерверних середовищах [4].

Профіль циклу (loop) представляє обмежене циклічне поширення. Замість необмеженої рекурсії ланцюг оброблення проходить фіксовану кількість стрибків і потім досягає термінальної умови. Цей профіль дає змогу досліджувати не лише ампліфікацію саму по собі, а й розрізнення між обмеженим виконанням, термінальним маркуванням та активним стримуванням. Чотири профілі навантаження разом з їхніми обсягами входу, очікуваними шаблонами поширення та основними сигналами ізоляції, що використовуються при оцінюванні, узагальнено в таблиці 1.

Таблиця 1

Профілі навантаження та експериментальні параметри

Профіль	Призначення	Вхід на запуск	Очікуване поширення	Основний сигнал ізоляції
Нормальний	Контрольний номінальний випадок		Оброблення 1:1 (AF ≈	Відсутній (без зростання карантину/DLQ)
Отруйний	Детермінований збій і повторні спроби		Збільшення через повторні спроби до DLQ (AF ≈ 4)	Зростання DLQ (dlqΔ ≈ 200)
Розгалуження	Каскад із високою ампліфікацією		Розгалуження (базове AF ≈ 16,6)	Стримування на основі карантину в захищеному режимі повного блокування (Зона A, τ = 0,0; верхня межа; лише для fan-out)
Обмежений цикл	Обмежене циклічне поширення		Ланцюг фіксованої глибини (AF ≈ 5)	Термінальне маркування / карантин у захищеному режимі (qΔ ≈ 50)

Політика шлюзу ризику та семантика карантину. Захищена архітектура вводить детермінований етап допуску на основі політики перед потраплянням подій до основної черги оброблення. Шлюз ризику присвоює кожній вхідній події скалярну оцінку ризику R(e) у діапазоні [0, 1] на основі чотирьох адитивних евристик, виведених з метаданих події. Початкове значення оцінки для всіх подій становить 0,05. Далі застосовуються чотири умовні прирости: (i) +0,70, якщо поле eventType дорівнює POISON або LOOP; (ii) +0,20, якщо payloadSizeBytes перевищує 32 000 байт; (iii) +0,20, якщо fanoutDegree дорівнює 20 або більше; (iv) +0,10, якщо hopCount дорівнює 3 або більше. Підсумкова оцінка обмежується значенням 0,99.

Структура ваг відображає відносну серйозність кожного сигналу ампліфікації. Домінуючий приріст +0,70 для подій POISON та LOOP охоплює їхній основний зв'язок зі збільшенням виконань через повторні спроби та обмежене циклічне поширення – двома класами навантаження, найбезпосередніше пов'язаними з некерованим зростанням обсягу виконань. Структурні сигнали fanoutDegree та payloadSizeBytes дають по +0,20 кожен, відображаючи вторинний ризик ампліфікації: високий ступінь розгалуження є прямим предиктором подальшого розширення, а великий розмір корисного навантаження може свідчити про складні або ланцюжкові структури подій. Приріст +0,10 за hopCount є найменшим, оскільки підвищена кількість стрибків є пізнім індикатором, який з'являється вже після початку поширення. Разом ці ваги утворюють чітко роздільні межі оцінок між експериментальними профілями навантаження, що дає змогу здійснювати маршрутизацію за порогом без потреби в навчених моделях.

Критерій payloadSizeBytes не спрацьовував у жодному експериментальному запуску, оскільки всі згенеровані корисні навантаження залишалися значно нижчими за поріг 32 000 байт (максимум близько 2100 байт). Формально:

$\in \{POISON, LOOP\}] + 0{,}20 \cdot 1[payloadSizeBytes > 32000] + 0{,}20 \cdot 1[fanoutDegree \geq 20] + 0{,}10 \cdot 1[hopCount \geq 3])$.

Рішення про маршрутизацію визначається одним пороговим параметром τ: подія допускається до основної черги, якщо $R(e) < \tau$, і перенаправляється до карантину в інакшому випадку. У таблиці 2 узагальнено отримані оцінки та рішення про допуск для кожного профілю навантаження за двох порогових конфігурацій, використаних у дослідженні.

Таблиця 2

Оцінки ризику та рішення про маршрутизацію за профілями навантаження

Профіль	eventType	fanoutDegree	hopCount	R(e)	Рішення ($\tau = 0{,}80$)	Рішення ($\tau = 0{,}25$)	Рішення ($\tau = 0{,}0$)
Нормальний	NORMAL	0	0	0,05	ДОПУСК	ДОПУСК	КАРАНТИН
Отруйний	POISON	0	0	0,75	ДОПУСК	КАРАНТИН	КАРАНТИН
Розгалуження	FANOUT	≥ 20	0	0,25	ДОПУСК	КАРАНТИН	КАРАНТИН
Цикл (стрибки 0-2)	LOOP	0	0–2	0,75	ДОПУСК	КАРАНТИН	КАРАНТИН
Цикл (стрибок ≥ 3)	LOOP	0	≥ 3	0,85	КАРАНТИН	КАРАНТИН	КАРАНТИН

Із реалізованих правил оцінювання та маршрутизації безпосередньо випливає кілька наслідків. По-перше, отруйні події отримують $R(e) = 0{,}75$, що нижче від типового порога $\tau = 0{,}80$, тож вони допускаються на вході та згодом ізолюються через вичерпання повторних спроб і поведінку DLQ. По-друге, поява карантину для циклу в захищених запусках не зумовлена повторним оцінюванням внутрішніх стрибків циклу на вході: шлюз ризику викликається лише для оригінальних вхідних подій (50 викликів для циклічного навантаження), а термінальне карантинне рішення приймається всередині обробника, коли hopCount сягає MAX_HOPS = 4. Отже, карантин для циклу відображає термінальне оброблення на стороні обробника, а не блокування проміжних стрибків з боку шлюзу. По-третє, конфігурація жорсткого блокування розгалуження ($\tau = 0{,}0$) гарантовано переводить усі вхідні події в карантин, оскільки мінімальна оцінка за політикою становить 0,05; тому ця конфігурація є умовою верхньої межі стримування, а не загальнопридатним порогом для розгортання.

Із структури оцінювання випливає центральне інтерпретаційне зауваження: карантин не несе однакового семантичного значення для всіх профілів навантаження. В експериментах з повним

блокуванням розгалуження (Зона А) карантин виконує роль механізму жорсткого стримування, оскільки перенаправлені події не потрапляють у подальший ланцюг обробника. В експериментах з обмеженим циклом карантин натомість виступає термінальним маркуванням: ланцюг циклу проходить усю свою глибину, а до карантину потрапляє лише фінальна подія, оскільки підвищене значення hopCount призводить до перевищення порога τ і формує аудиторський сигнал або сигнал доказової бази.

Метрики та процедура вимірювання. Основною метрикою оцінювання є коефіцієнт ампліфікації (AF), визначений як відношення загальної кількості опрацьованих подій до кількості вхідних подій у межах одного запуску. Ця метрика перетворює вплив поширення на вимірну величину, що безпосередньо узгоджується з більш широкою безпековою проблемою радіуса ураження у подійно-керованих системах та надмірних виконань, які призводять до зростання витрат [4]. Значення, близьке до 1, свідчить про оброблення «один до одного», тоді як значення, помітно більші за 1, вказують на подальше розширення. Для конфігурацій повного блокування (Зона А) значення AF, близьке до 0, свідчить про те, що поширення фактично припинено ще до досягнення етапу оброблення. Допоміжні метрики включають приріст черги карантину, приріст черги недоставлених повідомлень, кількість викликів обробника, кількість викликів шлюзу ризику, кількість обмежень одночасності та середню тривалість виконання, які реєструються засобами моніторингу AWS Lambda [20]. У дослідженні також використано час стабілізації конвеєра (time-to-quiescence, TTQ) як метрику, що показує, наскільки швидко конвеєр повертається до неактивного стану після початкового сплеску подій.

Експериментальний протокол та засоби забезпечення відтворюваності. Оцінювання побудовано за схемою A/B, у якій кожен профіль навантаження виконується як у базовому, так і в захищеному режимах за контрольованих умов. Кожен профіль виконується кілька разів, щоб зменшити вплив тимчасових коливань і забезпечити стабільнішу порівняльну інтерпретацію. Для профілю розгалуження запуски у захищеній архітектурі здійснювалися за порогом повного блокування $\tau = 0{,}0$ (Зона А), який динамічно застосовувався до Lambda-функції шлюзу ризику через оркестраційну змінну GUARDED_STRICT_THR перед кожним захищеним запуском з розгалуженням та відновлювався до типового значення $\tau = 0{,}80$ після нього. Зону В ($0{,}05 < \tau \leq 0{,}25$) додатково перевірено експериментально за $\tau = 0{,}25$ у двох конфігураціях (по $n = 10$): з увімкненим обмеженням розгалуження на обробнику (MAX_FANOUT = 5, конфігурація повного багаторівневого захисту) та з вимкненим обмеженням (MAX_FANOUT = 50, лише шлюз), щоб незалежно виокремити внесок вхідного шлюзу.

Для збереження ізоляції запусків кожному виконанню присвоюється унікальний ідентифікатор запуску, за яким із DynamoDB запитуються результати окремого запуску і інтерпретуються зміни довжини черг. Ізоляцію запусків забезпечено очищенням основної черги перед кожним запуском. Експериментальні налаштування відповідали моделі AWS Lambda з керуванням з боку Amazon SQS через відображення джерел подій із параметром повторної доставки SQS maxReceiveCount = 4, що пояснює спостережувану поведінку у разі отруйних повідомлень (200 входів → 800 спроб оброблення → приріст DLQ ≈ 200). Основними сигналами на рівні запуску є кількість результатів у DynamoDB, відфільтрованих за runId, та зміни довжини черг; метрики функцій з CloudWatch використовуються як додаткові операційні індикатори [20].

Описові статистики подано для десяти повторень на профіль/режим для нормального, розгалуженого та циклічного профілів і для трьох повторень отруйного профілю. Оскільки AF був детермінованим за фіксованої генерації навантаження, статистичне порівняння застосовано насамперед до метрик із реальною міжзапусковою варіацією, зокрема часу стабілізації конвеєра (TTQ) та обмежень одночасності обробника. Для профілю розгалуження використано однобічний U-критерій Манна – Уітні з попередньо заданими гіпотезами: TTQ у захищеному режимі < TTQ у базовому режимі та обмеження одночасності обробника в захищеному режимі < відповідного значення в базовому режимі. Розгалуження продемонструвало повне розділення для TTQ ($U = 100$; $p < 0{,}001$; $r = 1{,}0$) і для обмежень одночасності обробника ($U = 100$; $p < 0{,}001$; $r = 1{,}0$). Для отруйного профілю тести мали лише дослідницький характер і не виявили статистично значущого покращення (TTQ: $U = 4$; $p = 0{,}50$; обмеження одночасності: $U = 4{,}5$; $p = 1{,}00$), що узгоджується з роллю цього профілю як механістичної перевірки повторних спроб, а не основного сценарію стримування.

Обмеження експериментальної установки. Поточна конструкція захищає лише вхідний шлях конвеєра, а не кожен подальший перехід у графі подій. Шлюз ризику оцінює оригінальні вхідні повідомлення, тоді як події, згенеровані внаслідок внутрішнього оброблення, не проходять повторного оцінювання в тій самій точці допуску. Це обмеження особливо важливе для циклічного профілю, де карантин виконує радше функцію термінального маркування, ніж раннього блокування. Отже, захищену архітектуру слід трактувати як механізм стримування на вході, а не як універсальний фільтр для всього графа поширення.

Інші обмеження стосуються інтерпретації метрик і зовнішньої валідності. AF є зручною та відтворюваною мірою подальшого поширення, однак не охоплює всіх можливих побічних ефектів, зокрема записів у сховища, викликів сторонніх API або переходів бізнес-станів. Крім того, експеримент проведено в контрольованому AWS-стенді з чотирма синтетичними профілями навантаження, без змішаного трафіку, реальних трас застосунків і зловмисно змінених метаданих. Тому отримані результати слід інтерпретувати як контрольоване підтвердження концепції, а не як повну оцінку виробничих середовищ.

РЕЗУЛЬТАТИ ДОСЛІДЖЕННЯ

У цьому розділі представлено результати повторюваних A/B-експериментів, проведених для базової та захищеної архітектур за чотирма профілями навантаження, описаними у розділі «Методика дослідження». Аналіз зосереджено не лише на тому, чи були події успішно опрацьовані, а й на тому, наскільки сильно вони поширювалися конвеєром після допуску, оскільки головною метою дослідження є визначення того, чи можна спостерігати, вимірювати та зменшувати ампліфікацію подій за допомогою точки вхідного контролю. Така перспектива узгоджується з попередніми роботами, у яких непропорційне виконання у безсерверних системах розглядається не лише як питання надійності, а й як важлива для кібербезпеки проблема фінансового виснаження і зловживань [4]. Модель виконання, керованого чергою, відповідає інтеграції AWS Lambda з Amazon SQS [18], тоді як переходи до черги недоставлених повідомлень та метрики виконання інтерпретуються згідно з документованою семантикою Amazon SQS і моніторингу AWS Lambda [19, 20]. Загальна інтерпретація збільшення виконань через повторні спроби та ефектів стабілізації також узгоджується з попередніми емпіричними дослідженнями повторних спроб і поведінки автоматичних розмикачів у розподілених системах [16].

Базові вимірювання. Базову архітектуру оцінено першою для встановлення чіткої точки відліку щодо поведінки поширення за відсутності вхідного шлюзу. У цій конфігурації генератор надсилав події безпосередньо до основної черги оброблення, а обробник споживав їх без жодного попереднього контролю допуску.

За нормального навантаження базова архітектура демонструвала очікувану поведінку оброблення «один до одного»: 200 вхідних подій давали 200 успішно опрацьованих подій, AF залишався рівним 1,0, а приріст карантину та DLQ не спостерігався. Отже, цей профіль підтверджує відсутність непередбаченої ампліфікації за номінального трафіку та формує контрольний стан для подальшого порівняння із захищеною архітектурою.

Отруйне навантаження, навпаки, демонструвало збільшення кількості виконань через повторні спроби: 200 вхідних подій спричиняли 800 спроб оброблення, після чого повідомлення ізолювалися в DLQ. Таким чином, AF = 4,0 у цьому профілі відображає не розгалуження, а механіку повторної доставки та ізоляції невдалих повідомлень.

Навантаження з розгалуженням дало в базовій архітектурі найсильніший ефект поширення. У кожному повторному запуску 50 вхідних подій давали 830 опрацьованих подій, що відповідає AF = 16,6. Усі зареєстровані результати були успішними, повідомлення не потрапляли до карантину, а зростання черги недоставлених повідомлень не спостерігалося. Кількість викликів обробника близько відповідала масштабу поширення, сягаючи приблизно 830 на запуск, а час стабілізації конвеєра коливався в межах -49 с (середнє 40,0 ± 6,9 с). Ці вимірювання визначають базовий профіль розгалуження як найвиразніший сценарій неконтрольованої ампліфікації в експериментальному наборі.

Циклічне навантаження дало інший, але також нетривіальний шаблон поширення. У всіх десяти базових запусках 50 вхідних подій породжували 250 опрацьованих подій, що відповідає AF = 5,0. На відміну від сценарію з розгалуженням, це поширення залишалося обмеженим і повністю успішним: усі 250 результатів зареєстровано як успішні, тоді як прирости довжини карантину та черги недоставлених повідомлень дорівнювали нулю. Кількість викликів обробника стабільно сягала 250, а час стабілізації конвеєра коливався в межах 12-38 с.

У сукупності базові результати утворюють несуперечливу точку відліку для подальшого дослідження. Нормальний профіль підтверджує функціональну стабільність без ампліфікації, отруйний – демонструє збільшення виконань через повторні спроби перед ізоляцією у черзі недоставлених повідомлень, розгалуження – виявляє виражене неконтрольоване подальше розширення, а цикл – обмежене, але нетривіальне поширення.

Вимірювання у захищеному режимі. Захищену архітектуру оцінено за тих самих чотирьох профілів навантаження, щоб з’ясувати, як уведення етапу оцінювання ризику впливає на подальше поширення.

За нормального навантаження захищена архітектура зберегла функціональну еквівалентність із базовою: 200 вхідних подій давали 200 успішних результатів, AF залишався рівним 1,0, а карантин і DLQ не зростали. Додаткові виклики шлюзу ризику підтверджують, що всі вхідні події проходили через етап контрольованого допуску, хоча цей етап збільшив час стабілізації конвеєра порівняно з базовим режимом.

Для отруйного навантаження захищений режим не змінив основної динаміки повторних спроб: події були допущені на основний шлях, спричиняли 800 невдалих спроб оброблення і зрештою ізолювалися в DLQ. Отже, поточна конфігурація шлюзу не призначена для раннього блокування цього профілю за типовим порогом $\tau = 0{,}80$. Стримування отруйних повідомлень залишається функцією DLQ та семантики повторних спроб, а не вхідного шлюзу.

Навантаження з розгалуженням забезпечило найвиразніший контраст у поширенні між двома архітектурами. У захищеному режимі за порогом повного блокування ($\tau = 0{,}0$, Зона A) усі 50 вхідних подій повністю перенаправлено до карантину, а кількість викликів обробника впала до нуля (AF = 0,0); час стабілізації конвеєра становив приблизно 15 с (середнє 15,1 ± 0,3 с). Цей результат забезпечує теоретичну верхню межу стримування: конфігурація Зони A блокує також нормальний трафік, тому не може використовуватися як загальний операційний поріг.

Циклічне навантаження дало в захищеному режимі більш нюансовану картину. У всіх десяти повторних запусках 50 вхідних подій давали 250 опрацьованих подій, тож AF залишався рівним 5,0 і відповідав обмеженій глибині поширення, спостережуваній у базовій конфігурації. Водночас захищений циклічний профіль додав приріст карантину 50 і зареєстрував 50 невдалих результатів поряд із 200 успішними. Шлюз ризику реєстрував 50 викликів на запуск, що відповідало початковій множині допущених вхідних подій, тоді як обробник реєстрував 250 викликів, свідчачи про збереження обмеженого подальшого поширення. Час стабілізації конвеєра залишався низьким – у межах 14-33 с (середнє 20,9 ± 6,2 с).

Порівняльний аналіз: базова та захищена архітектури. Безпосереднє порівняння базової та захищеної архітектур показує, що ефект вхідного контролю суттєво залежить від типу навантаження. У таблиці 3 наведено повні результати «базова – захищена» за всіма профілями навантаження (n = 10).

Таблиця 3

Результати порівняння базової та захищеної архітектур (n = 10; отруйний n = 3; Зона B у двох конфігураціях)

Профіль	Режим	Вхід	Опрацьовано всього	AF	TTQ, середнє ± SD (с)	Δ карантину	Δ DLQ	Виклики обробника	Виклики шлюзу ризику
Нормальний	Базовий	200	200	1,0	13,9 ± 3,1	0	0	200	0
Нормальний	Захищений	200	200	1,0	31,4 ± 2,2	0	0	200	200
Отруйний	Базовий	200	800	4,0	610,7 ± 108,5	0	200	800	0
Отруйний	Захищений	200	800	4,0	600,3 ± 90,7	0	200	800	200
Розгалуження	Базовий	50	830	16,6	40,0 ± 6,9	0	0	830	0
Розгалуження	Захищений	50	0	0,0	15,1 ± 0,3	50	0	0	50
Розгалуження	Зона B, лише шлюз	50	214	4,28	16,2 ± 2,9	25	0	214	50
Розгалуження	Зона B, повний захист	50	150	3,0	17,3 ± 4,1	25	0	150	50
Цикл	Базовий	50	250	5,0	22,9 ± 7,5	0	0	250	0
Цикл	Захищений	50	250	5,0	20,9 ± 6,2	50	0	250	50

На рис. 2 узагальнено контраст ампліфікації між базовою та захищеною архітектурами за всіма профілями навантаження.

КІБЕРБЕЗПЕКА: освіта, наука, техніка
CYBERSECURITY: EDUCATION, SCIENCE, TECHNIQUE
№ 1 (33), 2026
ISSN 2663 – 4023

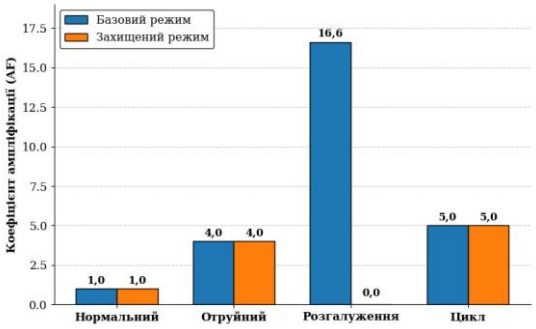

Рис. 2. Коефіцієнт ампліфікації AF за профілями навантаження у базовому та захищеному режимах.

Порівняння показує, що ефективність вхідного контролю істотно залежить від структури навантаження. Для нормального профілю захищена архітектура зберігає поведінку «один до одного» і не створює хибного карантину, хоча додавання шлюзу ризику збільшує час стабілізації конвеєра порівняно з базовим режимом. Для отруйного профілю AF залишається рівним 4,0 в обох архітектурах, оскільки за типовим порогом $\tau = 0{,}80$ такі події допускаються до основного шляху і надалі ізолюються через DLQ після повторних спроб. Отже, у цьому сценарії стримування забезпечується не вхідним шлюзом, а семантикою повторної доставки та черги недоставлених повідомлень.

Найсильніший ефект вхідного контролю спостерігається для навантаження з розгалуженням. У базовій архітектурі 50 вхідних подій породжують 830 опрацьованих подій, що відповідає AF = 16,6. Як показано на рис. 3, у конфігурації повного блокування ($\tau = 0{,}0$, Зона A) захищена архітектура зменшує AF до 0,0, оскільки всі вхідні події спрямовуються до карантину до потрапляння в основну чергу. Водночас цей режим є лише верхньою межею стримування, оскільки блокує також нормальний трафік. Практично важливішою є селективна конфігурація Зони B за $\tau = 0{,}25$: лише вхідний шлюз зменшує AF до 4,28, а поєднання шлюзу з обмеженням розгалуження на рівні обробника додатково знижує AF до 3,0.

Для циклічного профілю AF залишається рівним 5,0 в обох архітектурах, що підтверджує обмежений характер поширення, але також демонструє межі вхідного контролю. Шлюз оцінює лише початкові події, тоді як внутрішньо згенеровані події циклу продовжують оброблятися в межах обробника. Тому карантин у цьому профілі слід інтерпретувати не як раннє блокування, а як термінальне маркування завершеного циклічного ланцюга.

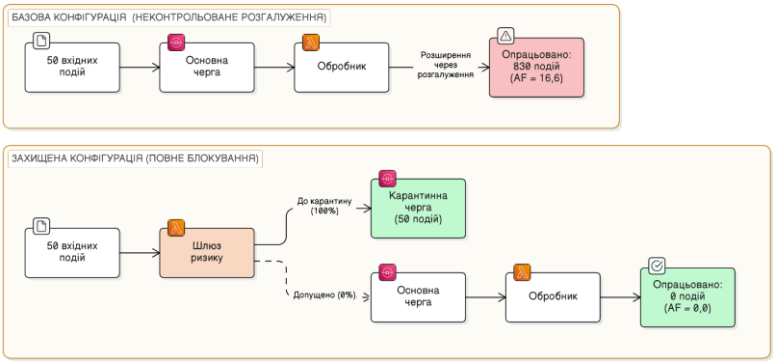

Рис. 3. Стримування розгалуження за порогом повного блокування ($\tau = 0{,}0$, Зона A).

Результати стримування: облік карантину та збоїв. Попередні підрозділи показали, що приріст карантину та облік збоїв не мають однакової інтерпретації для всіх профілів навантаження. Це розрізнення є важливим, оскільки той самий числовий сигнал може мати різний операційний зміст залежно від структури поширення в межах навантаження. У подійно-керованих системах така неоднозначність є звичайним явищем, оскільки видимий стан черг і лічильників часто відображає як бізнес-логіку, так і платформозалежну семантику доставки повідомлень [19].

Для навантаження з розгалуженням карантин має найчіткішу інтерпретацію. За конфігурації повного блокування (Зона A, $\tau = 0{,}0$) усі 50 вхідних подій перенаправлено до карантину, кількість викликів обробника залишилася рівною нулю, а AF упав до 0,0. Події не потрапили на основний шлях оброблення, тож і подальшого розширення не відбулося.

Для отруйного навантаження семантика інша. Зростання карантину не зафіксовано в жодній з архітектур, тоді як приріст черги недоставлених повідомлень стабільно сягав 200 після вичерпання повторних спроб. Як базові, так і захищені запуски реєстрували 800 невдалих спроб оброблення, що свідчить: збільшення кількості збоїв виникало через повторне оброблення далі в конвеєрі, а не через рішення про маршрутизацію на вході. Цей результат підтверджує інтерпретацію, що черги недоставлених повідомлень залишаються основним механізмом ізоляції для детермінованих збоїв оброблення [19].

Циклічне навантаження потребує найточнішого механістичного пояснення. У захищеній архітектурі шлюз ризику оцінює кожну вхідну подію та ухвалює рішення про маршрутизацію шляхом порівняння R(e) з порогом τ. Для вхідних подій циклу за hopCount = 0 та eventType = LOOP шлюз обчислює R(e) = 0,75, що нижче від типового порога τ = 0,80, тож усі 50 початкових подій допускаються до основної черги оброблення. Усе подальше поширення циклу повністю обробляється в межах обробника. Коли h = MAX_HOPS = 4, обробник переходить у термінальну гілку: він надсилає подію безпосередньо до карантинної черги та реєструє результат як bounded_loop_quarantined. Кожен з 50 вхідних ланцюгів виконує чотири успішні стрибки, після яких слідує один термінальний, що дає 200 успішних результатів і 50 карантинних, разом 250 викликів обробника та AF = 5,0. Приріст карантину 50, отже, відображає маршрутизацію виходу, ініційовану обробником, а не блокування на рівні вхідного шлюзу.

Із ширшої методологічної перспективи ці результати показують, що жодна окрема метрика не є достатньою для оцінювання стримування в подійно-керованих безсерверних конвеєрах. Зростання карантину саме по собі не відповідає на запитання, чи була подія заблокована до оброблення, ізольована після обмеженого виконання, чи лише перенаправлена для аудиту. Тому в цій роботі результати стримування тлумачаться через сумісне читання AF, змін довжини черг, кількості викликів обробника і шлюзу та обліку успіхів/збоїв.

Накладні витрати та побічні ефекти (затримка, обмеження одночасності, стабільність). Окрім ефективності стримування, захищену архітектуру потрібно оцінювати і з погляду операційних накладних витрат та побічних ефектів. Така ширша перспектива узгоджується з попередніми роботами, які показують, що повторні спроби, автоматичні розмикачі та засоби керування трафіком здатні підвищувати стійкість в одному вимірі, водночас спричиняючи побічні ефекти в іншому [16].

Нормальне навантаження дає найчіткіше уявлення про накладні витрати шляху контролю за номінальних умов. Захищений режим стабільно демонстрував більший час стабілізації конвеєра, ніж базовий: зі значень 10-20 с у базових запусках він зростав до 27-36 с у захищених. Ця різниця свідчить, що навіть за відсутності ампліфікації додавання етапу шлюзу ризику створює додаткову затримку виконання і маршрутизації. Водночас ці накладні витрати залишалися помірними і не змінювали функціональної поведінки навантаження.

Навантаження з розгалуженням демонструє найкорисніший профіль побічних ефектів для захищеної архітектури. У базовому режимі це навантаження давало 830 викликів обробника на запуск і спричиняло помітне обмеження одночасності (у середньому 36,3 події обмеження на запуск), що відображає високе миттєве розширення подальшого виконання. У режимі повного блокування Зони A, навпаки, кількість викликів обробника впала до нуля, обмежень одночасності не виникало, а час стабілізації конвеєра залишався коротким – близько 15 с (середнє 15,1 ± 0,3 с). Це означає, що захищена архітектура не лише зменшила ампліфікацію, а й усунула пов’язаний з розгалуженням тиск на обробник.

Отруйне навантаження виявляє інший тип операційного побічного ефекту. Як базові, так і захищені запуски мали тривалі періоди стабілізації – час стабілізації конвеєра сягав кількох сотень секунд. Цей ефект спричинений не самим шлюзом, а повторними циклами спроб перед ізоляцією у черзі недоставлених повідомлень. Захищена архітектура не зменшила цих витрат за поточної конфігурації істотно, що свідчить: збільшення виконань через отруйні повідомлення залишається переважно проблемою надійності далі в конвеєрі і потребує додаткових механізмів раннього обмеження до накопичення повторних спроб.

Циклічне навантаження не показало статистично значущої різниці у часі стабілізації між базовим і захищеним режимами (22,9 ± 7,5 с проти 20,9 ± 6,2 с; U-критерій Манна – Уітні = 54,5; p = 0,76; n = 10). Початкове спостереження за n = 3 вказувало на менший TTQ у захищеному режимі, але це не підтвердилося за більшого обсягу вибірки, що свідчить про варіацію малої вибірки, а не реальний ефект. Загалом аналіз побічних ефектів показує, що захищена архітектура створює вимірні, але прийнятні накладні витрати за номінальних навантажень, дає незначне пряме покращення для сценаріїв з домінуванням повторних спроб (отруйних) та забезпечує суттєві операційні переваги в умовах ампліфікації з розгалуженням, оскільки усуває навантаження на обробник і пов’язані з ним обмеження одночасності.

Аналіз чутливості до порога. Оскільки функція оцінювання ризику є детермінованою та явно специфікованою, поведінку шлюзу під час маршрутизації можна аналітично описати для всього діапазону значень порога. Цей аналіз виокремлює три відмінні зони стримування, межі яких визначаються безпосередньо значеннями оцінок, що отримуються для кожного профілю навантаження за фіксованого розподілу входу (SEED = 42, n = 50). У таблиці 4 узагальнено ці три зони.

Таблиця 4

Порогові зони та результати стримування для профілю розгалуження (n = 50, SEED = 42; Зона B перевірена експериментально за τ = 0,25)

Зона	Діапазон τ	Подій у карантин	Подій до обробника	AF	Нормальний трафік
A	$\tau \leq 0{,}05$	50/50	0	0,0	ЗАБЛОКОВАНО (верхня межа)
B (лише шлюз)	$0{,}05 < \tau \leq 0{,}25$	25/50	25 → 214	4,28	ДОПУЩЕНО
B (повний захист)	$0{,}05 < \tau \leq 0{,}25$	25/50	25 → 150	3,0	ДОПУЩЕНО
C	$\tau > 0{,}25$	0/50	50 → 830	16,6	ДОПУЩЕНО
—	$\tau = 0{,}80$ (типове)	0/50	50 → 830	16,6	ДОПУЩЕНО

Зона A ($\tau \leq 0{,}05$) відповідає конфігурації повного блокування, використаній в експериментах із захищеним розгалуженням. За такої конфігурації мінімальна оцінка події 0,05 задовольняє умові $R(e) \geq \tau$ для всіх подій незалежно від типу, що зумовлює повне блокування на вході та AF = 0,0. Однак Зона A блокує і події NORMAL. Тому налаштування повного блокування не може застосовуватися як загальний операційний поріг без блокування легітимного трафіку, а результат AF = 0,0 слід тлумачити як теоретичну верхню межу.

Зона B ($0{,}05 < \tau \leq 0{,}25$) – це вікно вибіркового стримування на вході: події NORMAL (R = 0,05) допускаються, тоді як події FANOUT із fanoutDegree ≥ 20 (R = 0,25) перенаправляються до карантину. За SEED = 42 для n = 50 вхідних подій розгалуження генератор обирає значення fanoutDegree з множини {5, 10, 20, 30} з рівною ймовірністю, що дає рівно 25 подій з fanoutDegree ≥ 20 і 25 з fanoutDegree < 20.

Зону B перевірено експериментально за τ = 0,25 у двох конфігураціях (по n = 10), щоб виокремити внесок кожного механізму стримування. У конфігурації «лише шлюз» (MAX_FANOUT = 50, обмеження на обробнику фактично вимкнено) AF зменшився з 16,6 до 4,28 ± 0,07 – це 74,2 % зниження, які повністю припадають на вхідну фільтрацію. Цей результат підтверджує початкову аналітичну оцінку 4,3 з точністю до 0,5 % і валідує розрахунок для конфігурації «лише шлюз». У конфігурації повного багаторівневого захисту (MAX_FANOUT = 5) AF додатково зменшився до 3,0 ± 0,04 – це 82,0 % загального зниження. Розклад показує, що за поточного навантаження та метрики AF саме «лише шлюз» (16,6 → 4,28) забезпечує 90,5 % загального спостереженого зниження AF, тоді як обмеження на обробнику дає решту 9,5 % (4,28 → 3,0).

Час стабілізації конвеєра в обох конфігураціях Зони B був порівнянним: лише шлюз – 16,2 ± 2,9 с; повний багаторівневий захист – 17,3 ± 4,1 с (U-критерій Манна — Уітні = 40,5; p = 0,48; н.з.); обидва значення суттєво нижчі за базовий профіль розгалуження (40,0 ± 6,9 с). Статистично значущої різниці у TTQ між двома конфігураціями Зони B за тестованого навантаження не виявлено. Зона C ($\tau > 0{,}25$) не забезпечує жодного блокування розгалуження на вході, оскільки всі такі вхідні події допускаються шлюзом, у тому числі за типового значення $\tau = 0{,}80$. Залежність «поріг – стримування» є ступінчастою, а не неперервною: переходи відбуваються точно на межах оцінок, заданих реалізованою політикою. Будь-яке значення в Зоні B одночасно зберігає нормальний трафік і блокує події з високим ступенем розгалуження; тож на практиці слід обирати $\tau \in (0{,}05; 0{,}25]$.

Інтерпретація поведінки розгалуження та циклу. Експериментальні результати показують, що ампліфікацію подій у безсерверних конвеєрах не варто розглядати як єдине однорідне явище, оскільки різні шаблони поширення мають різні безпекові й операційні значення. Навантаження з розгалуженням представляє неконтрольоване подальше розширення, коли одна допущена подія запускає непропорційно великий обсяг подальшого оброблення і тим самим безпосередньо збільшує радіус ураження [4]. Натомість навантаження з обмеженим циклом представляє структуроване поширення фіксованої глибини; воно залишається релевантним для ампліфікації, проте не обов'язково тягне за собою той самий

У профілі розгалуження вхідний шлюз працював як базовий механізм стримування, оскільки всі ризикові події перенаправлялися ще до того, як могли потрапити на шлях обробника. Падіння AF з 16,6 до 0 свідчить про те, що вхідна маршрутизація здатна функціонувати як ефективний механізм зменшення радіуса ураження, коли джерело ампліфікації локалізоване в точці допуску. Цей результат узгоджується із загальною ідеєю, що зловживання у безсерверних системах стають найбільш шкідливими саме тоді, коли спочатку невеликому вхідному обсягу дозволяють безперешкодно розгорнутися у великий обсяг подальших виконань [4].

Профіль циклу, однак, показує, що не всяке поширення слід інтерпретувати через ту саму логіку стримування. У захищених циклічних запусках ампліфікація залишалася такою ж, як і в базовому випадку, що свідчить про те, що обмежений ланцюг усе ж виконувався до кінця. Водночас з'явилися кориріст карантину та облік термінальних збоїв, тож захищена архітектура інакше обробляла фінальну подію циклу, ніж з рештою ланцюга. Це означає, що поведінку захищеного циклу краще трактувати як термінальне маркування або ізоляцію обмеженої події, а не як пряме стримування поширення.

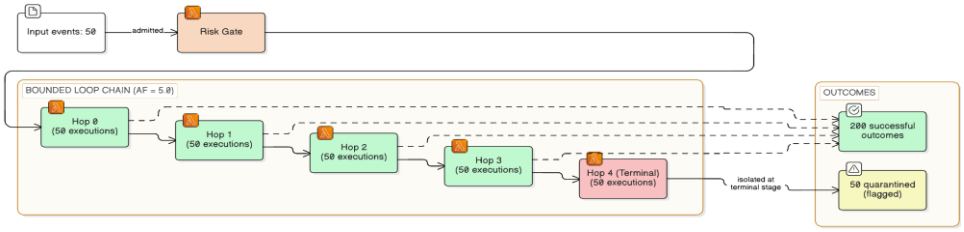

Рис. 4. Інтерпретація поведінки обмеженого циклу у захищеному режимі.

З аналітичного погляду ця відмінність між розгалуженням і циклом є одним із найважливіших висновків дослідження. Вона показує, що карантин не має універсального семантичного значення в механізмах захисту подійно-керованих систем. В одному навантаженні він представляє жорстке стримування, а в іншому – вибіркове термінальне оброблення. Тож наявні результати свідчать, що дослідження стримування в безсерверних конвеєрах завжди мають інтерпретувати зростання черг разом з кількістю виконань обробника та метриками ампліфікації, а не покладатися на жоден окремий сигнал ізольовано.

Загрози для валідності. Під час інтерпретації результатів цього дослідження необхідно враховувати кілька міркувань щодо валідності. По-перше, внутрішня валідність експерименту залежить від коректності процедури вимірювання, яка пов'язує стан черг, результати обробника та метрики виконання з окремими запусками. Цей ризик пом'якшено присвоєнням унікальних ідентифікаторів запусків, очищенням черг перед експериментальними запусками та відокремленням базового і захищеного середовищ через явні конфігураційні контролі. Ці запобіжні заходи знижують взаємний вплив між запусками і посилюють причинно-наслідкову порівнюваність між двома архітектурами, але не усувають усіх можливих часових викривлень, які можуть виникати в асинхронних хмарних системах [18].

Друге зауваження щодо внутрішньої валідності пов'язане з припущенням про довіреного виробника, закладеним у функції оцінювання ризику. Генератор навантаження заповнює поля метаданих події (eventType, fanoutDegree, hopCount) відповідно до заданого експериментального протоколу, без навмисного спотворення цих значень. Унаслідок цього результати експерименту відображають продуктивність шлюзу за моделі з довіреними виробниками, а не за зловмисницької моделі, у якій поля корисного навантаження свідомо маніпулюються, щоб обійти оцінювання. Безпекова валідність результатів, отже, є умовною – вона залежить від того, чи відповідає контекст розгортання припущенню про довіреного виробника, описаному в розділі «Системна модель».

Третє внутрішнє обмеження валідності полягає в залежності від спостережуваних сигналів, що генеруються провайдером. Такі метрики, як кількість викликів, обмеження одночасності та тривалість, отримуються з AWS CloudWatch і тому відображають семантику агрегації та звітування провайдера, а не низькорівневі трасування часу виконання [20]. Для вимірювання довжини черг використовується періодичний опитувальний підхід через SQS GetQueueAttributes до стабілізації, що пом'якшує затримку eventual consistency у ApproximateNumberOfMessages; проте залишкові затримки звітування за умов високої пропускної здатності повністю виключити не можна [19].

З погляду конструктної валідності головне питання полягає в тому, чи адекватно AF відображає радіус ураження у подійно-керованих безсерверних системах. AF забезпечує практичну та відтворювану міру обсягу подальшого поширення, що робить його особливо придатним для контрольованих A/B-порівнянь. Однак він не охоплює усіх можливих форм впливу – наприклад, запис у постійне сховище,

виклики сторонніх API чи переходи стану на рівні бізнес-логіки. Зовнішня валідність обмежена випробувальним стендом в одному акаунті та одному регіоні, а також чотирма контрольованими профілями навантаження. Попри це, центральна ідея – оцінювати допуск події як точку контролю радіуса ураження – залишається релевантною і поза межами конкретної реалізації, особливо тому, що базові проблеми повторних спроб, розгалуження та обмежених циклів широко проявляються в подійно-керованих хмарних системах [1].

Практичні наслідки для проєктування безпеки подійно-керованих систем. З практичного погляду результати показують, що подійно-керовані безсерверні системи доцільно оцінювати не лише за коректністю виконання або наявністю механізмів відновлення, а й за здатністю обмежувати подальше поширення подій після їх допуску. Повторні спроби, DLQ, обмеження одночасності та провайдерський моніторинг залишаються необхідними, однак вони переважно діють після початку поширення або лише регулюють темп виконання. Натомість вхідний шлюз ризику дає змогу впливати на радіус ураження до потрапляння подій у головний шлях оброблення. Практичне застосування такого підходу є найбільш обгрунтованим у середовищах з довіреними виробниками подій, де структурні метадані можуть використовуватися для детермінованого ухвалення рішень про допуск або карантин.

Зв’язок із вбудованими механізмами контролю AWS. Вхідний шлюз, досліджений у цій статті, доповнює засоби контролю платформи AWS, а не замінює їх. У таблиці 5 узагальнено їхні можливості щодо стримування за чотирма профілями навантаження з використанням як експериментальних результатів, так і документованої семантики кожного механізму. До таблиці 5 не включено два додаткові вбудовані механізми AWS – фільтрацію подій за вмістом в Amazon EventBridge та політики фільтрації Amazon SNS, – оскільки вони стосуються інших топологій тригерів, відмінних від виконання Lambda, керованого чергою, а їхня деталізація фільтрування обмежена зіставленням за зразками на статичних полях і не охоплює складеного скалярного оцінювання ризику.

Таблиця 5

Порівняння механізмів контролю поширення за профілями навантаження

Механізм	**Розміщення**	**Стримування розгалуження**	**Ізоляція отруйних повідомлень**	**Термінація циклу**	**Накладні витрати на нормальний трафік**
SQS DLQ + maxReceiveCount	Після оброблення	Відсутнє	Так – після вичерпання повторних спроб	Відсутнє	Відсутні
Зарезервована одночасність Lambda	На вході обробника	Часткове – обмежує одночасність, але не блокує	Відсутнє	Відсутнє	Відсутні, якщо межу не перевищено
Вхідний шлюз τ = 0,80 (типовий)	Перед обробленням	Відсутнє ($R = 0{,}25 < \tau$)	Відсутнє ($R = 0{,}75 < \tau$)	Термінальне маркування	+~15-20 мс на подію (середня тривалість шлюзу)
Вхідний шлюз $\tau \in (0{,}05; 0{,}25]$ (Зона B, лише шлюз)	Перед обробленням	Часткове – AF = 4,28, зниження на 74,2 %	Відсутнє	Термінальне маркування	+~15-20 мс на подію (середня тривалість шлюзу)
Вхідний шлюз $\tau \in (0{,}05; 0{,}25]$ + обмеження на обробнику (Зона B, повний захист)	Перед обробленням	Часткове – AF = 3,0, зниження на 82,0 %	Відсутнє	Термінальне маркування	+~15-20 мс на подію (середня тривалість шлюзу)
Вхідний шлюз $\tau \leq 0{,}05$ (Зона A / повне блокування)	Перед обробленням	Повне – AF = 0,0	Повне – все заблоковано (аналітично)	Повне – все заблоковано	Нормальний трафік заблоковано

З цього порівняння випливають три спостереження. По-перше, механізм черги недоставлених повідомлень є єдиним засобом, який ізолює отруйні повідомлення, але робить це лише після того, як повне збільшення виконань через повторні спроби вже відбулося [19]. Експериментальні дані це підтверджують: і базові, і захищені запуски з отруйними повідомленнями давали 800 викликів обробника та приріст DLQ 200, тобто радіус ураження повністю реалізовувався ще до ізоляції через DLQ.

По-друге, зарезервована одночасність Lambda виконує роль додаткового запобіжного механізму проти розширення розгалуження, обмежуючи миттєву пропускну здатність обробника, проте не запобігає потраплянню в конвеєр подій з високою ампліфікацією. Базові запуски з розгалуженням фіксували в середньому 36,3 події обмеження на запуск поряд з 830 викликами обробника, що підтверджує: вбудоване керування одночасністю платформи спрацьовувало, проте не зменшувало обсягу подальшого оброблення.

По-третє, вхідний шлюз є єдиним механізмом у цьому наборі, який діє до того, як починається подальше поширення. За порогом Зони B він вибірково блокує події з найбільшим ступенем розширення, водночас допускаючи нормальний трафік, зменшуючи AF з 16,6 до 4,28 лише через вхідну фільтрацію (зниження на 74,2 %) та до 3,0 у поєднанні з обмеженням розгалуження на обробнику (загальне зниження на 82,0 %).

ВИСНОВКИ ТА ПЕРСПЕКТИВИ ПОДАЛЬШИХ ДОСЛІДЖЕНЬ

У статті надмірне поширення подій у безсерверних конвеєрах AWS розглянуто як кількісно вимірюваний кіберризик, що збільшує операційний радіус ураження, ризики доступності та фінансову експозицію. Для дослідження цього ризику запропоновано відтворюваний A/B-експериментальний дизайн, який порівнює базову архітектуру прямого надходження подій до основної черги із захищеною архітектурою, де перед основним шляхом оброблення розміщено детермінований вхідний шлюз оцінювання ризику з карантинною маршрутизацією.

Результати показують, що ефект вхідного шлюзу суттєво залежить від структури навантаження. За номінального трафіку захищена архітектура зберегла поведінку оброблення «один до одного» і додала лише помірні накладні витрати до шляху контролю. За умов отруйних повідомлень захищений режим у поточній конфігурації істотно не зменшив збільшення виконань через повторні спроби – це свідомий наслідок логіки оцінювання, у якій події POISON опиняються нижче від типового порога; тому для цього класу навантажень провідними механізмами залишаються семантика повторних спроб і ізоляція через DLQ [19]. Для навантаження з розгалуженням експериментально перевірена захищена конфігурація з порогом повного блокування ($\tau = 0{,}0$, Зона A) повністю усунула подальше оброблення (AF = 0,0), перенаправляючи всі вхідні події до карантину до будь-якої участі обробника; цей результат Зони A є верхньою межею стримування, але водночас блокує і нормальний трафік, а отже, непридатний для загального використання. Аналіз чутливості до порога визначає Зону B ($0{,}05 < \tau \leq 0{,}25$) як операційно придатне вікно вибіркового стримування на вході. Зону B перевірено експериментально за $\tau = 0{,}25$ у двох конфігураціях (по n = 10). Конфігурація «лише шлюз» (обмеження на обробнику вимкнено) дала AF = 4,28 – це 74,2 % зниження, які повністю припадають на вхідну фільтрацію, і підтверджує початкову аналітичну оцінку з точністю до 0,5 %. Конфігурація повного багаторівневого захисту (з увімкненим обмеженням на обробнику) додатково знизила AF до 3,0 – це 82,0 % загального зниження. Наукова новизна дослідження полягає в трьох головних аспектах. По-перше, ампліфікацію подій формалізовано через практичну схему вимірювання, у якій центральною метрикою є AF, а допоміжними показниками – зміни довжини черг, кількість викликів та метрики стабілізації. По-друге, запропоновано відтворюваний експериментальний стенд на базі AWS з контрольованими профілями навантаження, що робить поведінку поширення вимірною і порівнюваною [25]. По-третє, показано, що простий детермінований вхідний шлюз може виконувати роль базового механізму зменшення радіуса ураження у подійно-керованих безсерверних конвеєрах без використання навчених моделей чи складних механізмів адаптивного ухвалення рішень.

З практичного погляду результати свідчать, що в подійно-керованих системах, наближених до промислової експлуатації, ампліфікацію слід трактувати як ключову безпекову та операційну характеристику. Стандартні механізми, як-от повторні спроби, черги недоставлених повідомлень і провайдерський моніторинг, залишаються необхідними, проте самі по собі не забезпечують раннього контролю поширення [18, 19, 20]. Тож висновки роботи підтримують застосування явних точок ухвалення рішення на вході в тих випадках, коли невелика кількість помилково допущених подій могла б породити непропорційно великі подальші ефекти. У ширшому плані дослідження показує, що вимірювання поширення є необхідним кроком до проєктування подійно-керованих систем, які є не лише

функціонально коректними та стійкими після збою, а й здатні обмежувати радіус ураження ще до того, як каскад повністю розгорнеться.

З поточної роботи природно випливає кілька напрямів подальших досліджень. Перший стосується області стримування. У наявній конструкції шлюз ризику працює лише на вході, тобто оцінює оригінальні вхідні події, але не переоцінює подій, що генеруються пізніше всередині ланцюга оброблення. Розширення логіки стримування за межі точки початкового допуску дасть змогу досліджувати, чи можна зменшувати поширення не лише на вході, а й у глибших частинах графа подій. Така розширена логіка була б особливо актуальною для сценаріїв з обмеженим циклом, де наявні результати показують, що події, згенеровані всередині конвеєра, залишаються поза прямим впливом вхідного шлюзу.

Другий напрям стосується семантики карантину та термінального оброблення. Експерименти показали, що карантин не завжди має той самий операційний зміст. У сценаріях з розгалуженням він є чітким механізмом стримування, тоді як у сценаріях з обмеженим циклом він більше виступає каналом термінальної ізоляції або маркування. Тож майбутні роботи могли б уточнити логіку контролю так, щоб стримування, аудит і термінальна класифікація трактувалися як явно роздільні наслідки, а не агрегувалися в єдину інтерпретацію карантину.

Третій напрям стосується розширених стратегій оцінювання ризику. Наявна робота свідомо ґрунтується на детермінованому шлюзі, що реалізує політику оцінювання ризику задля прозорості, відтворюваності та причинно-наслідкової інтерпретованості. Однак подальші дослідження могли б вивчати адаптивні або контекстно-чутливі політики оцінювання, які використовують додаткові метадані подій, стан черг або історичні характеристики виконання, залишаючись водночас достатньо придатними для інтерпретації під час розгортання, наближеного до промислової експлуатації.

Нарешті, експериментальну базу можна було б розширити як за охопленням платформ, так і за різноманіттям навантажень. Наявне дослідження зосереджено на конвеєрі AWS Lambda та Amazon SQS з чотирма контрольованими профілями. Майбутні роботи могли б розглянути складніші шини подій, багатоетапні робочі процеси, міжсервісне розгалуження або гетерогенні асинхронні архітектури. Також було б корисно оцінити, чи зберігає запропонована методологія вимірювання ампліфікації свою ефективність за вищих навантажень, змішаних робочих умов або менш синтетичної поведінки застосунків. Ці напрями посилили б зовнішню валідність і дали б змогу визначити, наскільки широко запропонований підхід до вимірювання та стримування узагальнюється на наближені до промислових подійно-керовані системи.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Marin, E., Perino, D., & Di Pietro, R. (2022). Serverless computing: A security perspective. *Journal of Cloud Computing, 11*, 69. https://doi.org/10.1186/s13677-022-00347-w
2. Escaleira, P., Cunha, V. A., Barraca, J. P., Gomes, D., & Aguiar, R. (2025). A systematic review on security mechanisms for serverless computing. *Cluster Computing*. https://doi.org/10.1007/s10586-025-05371-4
3. National Institute of Standards and Technology. (2024). *The NIST Cybersecurity Framework (CSF) 2.0*. https://doi.org/10.6028/NIST.CSWP.29
4. Kelly, D., Glavin, F. G., & Barrett, E. (2021). Denial of wallet – Defining a looming threat to serverless computing. *Journal of Information Security and Applications, 60*, 102843. https://doi.org/10.1016/j.jisa.2021.102843
5. Xiong, J., Wei, M., Lu, Z., & Liu, Y. (2021). Warmonger: Inflicting denial-of-service via serverless functions in the cloud. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security* (pp. 1046-1060). https://doi.org/10.1145/3460120.3485372
6. Shen, J., Zhang, H., Geng, Y., Li, J., Wang, J., & Xu, M. (2022). Gringotts: Fast and accurate internal denial-of-wallet detection for serverless computing. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*. https://doi.org/10.1145/3548606.3560629
7. Kelly, D., Glavin, F. G., & Barrett, E. (2024). DoWNet – Classification of denial-of-wallet attacks on serverless application traffic. *Journal of Cybersecurity, 10*(1), tyae004. https://doi.org/10.1093/cybsec/tyae004
8. Dorsett, M., Mann, S., Chowdhury, J., & Mahmood, A. (2025). *A comprehensive review of denial of wallet attacks in serverless architectures*. arXiv. https://doi.org/10.48550/arXiv.2508.19284
9. Alkhalifa, A. K., Aljebreen, M., Alanazi, R., Ahmad, N., Alahmari, S., Alrusaini, O., Alqazzaz, A., & Alkhiri, H. (2025). Mitigating malicious denial of wallet attack using attribute reduction with deep

learning approach for serverless computing on next-generation applications. *Scientific Reports, 15*, 18720. https://doi.org/10.1038/s41598-025-01178-w

10. Lavi, D., Brodt, O., Mimran, D., Elovici, Y., & Shabtai, A. (2025). Detection of compromised functions in a serverless cloud environment. *Computers & Security, 150*, 104261. https://doi.org/10.1016/j.cose.2024.104261
11. Ben-Shimol, L., Grolman, E., Elyashar, A., Maimon, I., Mimran, D., Brodt, O., Strassmann, M., Lehmann, H., Elovici, Y., & Shabtai, A. (2025). Observability and incident response in managed serverless environments using ontology-based log monitoring. *IEEE Transactions on Cloud Computing*. https://doi.org/10.1109/TCC.2025.3528320
12. Li, C., Huang, L., He, D., Wen, Y., Liu, G., & Duan, L. (2025). *FaaSMT: Lightweight serverless framework for intrusion detection using Merkle tree and task inlining*. arXiv. https://doi.org/10.48550/arXiv.2503.06532
13. Nguyen, C., Elmroth, E., & Bhuyan, M. (2025). Silent failures in stateless systems: Rethinking anomaly detection for serverless computing. In *2025 IEEE International Conference on Service-Oriented System Engineering (SOSE)*. https://doi.org/10.1109/SOSE67019.2025.00006
14. Polinsky, I., Datta, P., Bates, A., & Enck, W. (2024). GRASP: Hardening serverless applications through graph reachability analysis of security policies. In *Proceedings of the ACM Web Conference 2024*. https://doi.org/10.1145/3589334.3645436
15. Lazzari, L., & Farias, K. (2023). *Uncovering the hidden potential of event-driven architecture: A research agenda*. arXiv. https://doi.org/10.48550/arXiv.2308.05270
16. Saleh Sedghpour, M. R., Klein, C., & Tordsson, J. (2022). An empirical study of service mesh traffic management policies: Circuit breakers and retries. In *Proceedings of the 2022 ACM/SPEC International Conference on Performance Engineering*. https://doi.org/10.1145/3489525.3511686
17. Mohammad, M. (2025). *Resilient microservices: A systematic review of recovery patterns, strategies, and evaluation frameworks*. arXiv. https://doi.org/10.48550/arXiv.2512.16959
18. Amazon Web Services. (2026a). *Using Lambda with Amazon SQS*. AWS Lambda Developer Guide. https://docs.aws.amazon.com/lambda/latest/dg/with-sqs.html
19. Amazon Web Services. (2026b). *Using dead-letter queues in Amazon SQS*. Amazon SQS Developer Guide. https://docs.aws.amazon.com/AWSSimpleQueueService/latest/SQSDeveloperGuide/sqs-dead-letter-queues.html
20. Amazon Web Services. (2026c). *Using CloudWatch metrics with Lambda*. AWS Lambda Developer Guide. https://docs.aws.amazon.com/lambda/latest/dg/monitoring-metrics.html
21. Hamza, M., Akbar, M. A., & Capilla, R. (2023). *Understanding cost dynamics of serverless computing*. arXiv. https://doi.org/10.48550/arXiv.2311.13242
22. Lin, C., Ma, Y., & Shahrad, M. (2025). *Getting to the bottom of serverless billing*. arXiv. https://doi.org/10.48550/arXiv.2506.01283
23. Bodner, T., Radig, T., Justen, D., Ritter, D., & Rabl, T. (2025). *An empirical evaluation of serverless cloud infrastructure for large-scale data processing*. arXiv. https://doi.org/10.48550/arXiv.2501.07771
24. Soldani, J., Forti, S., Roveroni, L., & Brogi, A. (2025). Explaining microservices' cascading failures from their logs. *Software: Practice and Experience, 55*(5), 809-828. https://doi.org/10.1002/spe.3400
25. Molnar, V. (2025). *Event amplification risk in AWS serverless pipelines* [GitHub repository]. GitHub. https://github.com/j9mbo/event-amplification-risk-aws

КІБЕРБЕЗПЕКА: освіта, наука, техніка
CYBERSECURITY: EDUCATION, SCIENCE, TECHNIQUE
№ 1 (33), 2026
ISSN 2663 – 4023

Vitalii Molnar
Postgraduate student, Department of Information Security
Lviv Polytechnic National University, Lviv, Ukraine
ORCID: 0009-0001-3183-0117
vitalii.v.molnar@lpnu.ua

Dmytro Sabodashko
PhD, Department of Information Security
Lviv Polytechnic National University, Lviv, Ukraine
ORCID: 0000-0003-1675-0976
dmytro.v.sabodashko@lpnu.ua

DETERMINISTIC PROPAGATION CONTROL IN EVENT-DRIVEN SERVERLESS SYSTEMS

Abstract. Event-driven serverless pipelines can exhibit event amplification, where a small number of admitted inputs triggers a disproportionately large volume of downstream executions through fan-out, retries, or cyclic propagation. This phenomenon is security-relevant because it expands operational blast radius, increases availability risk, and enables cost-exhaustion effects in pay-per-use environments. This paper presents a reproducible AWS-based experimental study of propagation control in a queue-triggered serverless pipeline built on AWS Lambda and Amazon SQS. Two architectures are compared using an A/B protocol: a baseline design in which events are injected directly into the main queue, and a guarded design that inserts a deterministic ingress risk gate with quarantine routing prior to the main processing path. The evaluation operationalises propagation using an Amplification Factor complemented by queue deltas and invocation metrics. Results show functional equivalence under nominal traffic (Amplification Factor = 1.0 in both modes) and retry-driven inflation for poison messages consistent with dead-letter queue isolation. For the fan-out workload, a total-blocking threshold ($\tau = 0.0$, Zone A) eliminated all downstream processing (Amplification Factor = 0.0), representing a theoretical upper bound that also blocks normal traffic. Zone B ($0.05 < \tau \leq 0.25$) was experimentally validated at $\tau = 0.25$: the ingress gate alone reduced AF from 16.6 to 4.28 (74.2% reduction); adding the processor-side fan-out cap further reduced AF to 3.0 (82.0% total reduction). A bounded loop workload remains propagation-bounded (Amplification Factor = 5.0) while exhibiting workload-dependent terminal flagging semantics. Overall, the findings demonstrate that a minimal, deterministic ingress control point can substantially reduce blast radius for high-amplification event patterns without relying on learned models, provided that event producers operate within a trusted-producer perimeter.

Keywords: serverless security; event-driven architectures; AWS Lambda; Amazon SQS; event amplification; blast radius; quarantine routing

REFERENCES (TRANSLATED AND TRANSLITERATED)

1. Marin, E., Perino, D., & Di Pietro, R. (2022). Serverless computing: A security perspective. *Journal of Cloud Computing, 11*, 69. https://doi.org/10.1186/s13677-022-00347-w
2. Escaleira, P., Cunha, V. A., Barraca, J. P., Gomes, D., & Aguiar, R. (2025). A systematic review on security mechanisms for serverless computing. *Cluster Computing*. https://doi.org/10.1007/s10586-025-05371-4
3. National Institute of Standards and Technology. (2024). *The NIST Cybersecurity Framework (CSF) 2.0*. https://doi.org/10.6028/NIST.CSWP.29
4. Kelly, D., Glavin, F. G., & Barrett, E. (2021). Denial of wallet – Defining a looming threat to serverless computing. *Journal of Information Security and Applications, 60*, 102843. https://doi.org/10.1016/j.jisa.2021.102843
5. Xiong, J., Wei, M., Lu, Z., & Liu, Y. (2021). Warmonger: Inflicting denial-of-service via serverless functions in the cloud. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security* (pp. 1046-1060). https://doi.org/10.1145/3460120.3485372
6. Shen, J., Zhang, H., Geng, Y., Li, J., Wang, J., & Xu, M. (2022). Gringotts: Fast and accurate internal denial-of-wallet detection for serverless computing. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*. https://doi.org/10.1145/3548606.3560629

7. Kelly, D., Glavin, F. G., & Barrett, E. (2024). DoWNet – Classification of denial-of-wallet attacks on serverless application traffic. *Journal of Cybersecurity, 10*(1), tyae004. https://doi.org/10.1093/cybsec/tyae004
8. Dorsett, M., Mann, S., Chowdhury, J., & Mahmood, A. (2025). *A comprehensive review of denial of wallet attacks in serverless architectures*. arXiv. https://doi.org/10.48550/arXiv.2508.19284
9. Alkhalifa, A. K., Aljebreen, M., Alanazi, R., Ahmad, N., Alahmari, S., Alrusaini, O., Alqazzaz, A., & Alkhiri, H. (2025). Mitigating malicious denial of wallet attack using attribute reduction with deep learning approach for serverless computing on next-generation applications. *Scientific Reports, 15*, 18720. https://doi.org/10.1038/s41598-025-01178-w
10. Lavi, D., Brodt, O., Mimran, D., Elovici, Y., & Shabtai, A. (2025). Detection of compromised functions in a serverless cloud environment. *Computers & Security, 150*, 104261. https://doi.org/10.1016/j.cose.2024.104261
11. Ben-Shimol, L., Grolman, E., Elyashar, A., Maimon, I., Mimran, D., Brodt, O., Strassmann, M., Lehmann, H., Elovici, Y., & Shabtai, A. (2025). Observability and incident response in managed serverless environments using ontology-based log monitoring. *IEEE Transactions on Cloud Computing*. https://doi.org/10.1109/TCC.2025.3528320
12. Li, C., Huang, L., He, D., Wen, Y., Liu, G., & Duan, L. (2025). *FaaSMT: Lightweight serverless framework for intrusion detection using Merkle tree and task inlining*. arXiv. https://doi.org/10.48550/arXiv.2503.06532
13. Nguyen, C., Elmroth, E., & Bhuyan, M. (2025). Silent failures in stateless systems: Rethinking anomaly detection for serverless computing. In *2025 IEEE International Conference on Service-Oriented System Engineering (SOSE)*. https://doi.org/10.1109/SOSE67019.2025.00006
14. Polinsky, I., Datta, P., Bates, A., & Enck, W. (2024). GRASP: Hardening serverless applications through graph reachability analysis of security policies. In *Proceedings of the ACM Web Conference 2024*. https://doi.org/10.1145/3589334.3645436
15. Lazzari, L., & Farias, K. (2023). *Uncovering the hidden potential of event-driven architecture: A research agenda*. arXiv. https://doi.org/10.48550/arXiv.2308.05270
16. Saleh Sedghpour, M. R., Klein, C., & Tordsson, J. (2022). An empirical study of service mesh traffic management policies: Circuit breakers and retries. In *Proceedings of the 2022 ACM/SPEC International Conference on Performance Engineering*. https://doi.org/10.1145/3489525.3511686
17. Mohammad, M. (2025). *Resilient microservices: A systematic review of recovery patterns, strategies, and evaluation frameworks*. arXiv. https://doi.org/10.48550/arXiv.2512.16959
18. Amazon Web Services. (2026a). *Using Lambda with Amazon SQS*. AWS Lambda Developer Guide. https://docs.aws.amazon.com/lambda/latest/dg/with-sqs.html
19. Amazon Web Services. (2026b). *Using dead-letter queues in Amazon SQS*. Amazon SQS Developer Guide. https://docs.aws.amazon.com/AWSSimpleQueueService/latest/SQSDeveloperGuide/sqs-dead-letter-queues.html
20. Amazon Web Services. (2026c). *Using CloudWatch metrics with Lambda*. AWS Lambda Developer Guide. https://docs.aws.amazon.com/lambda/latest/dg/monitoring-metrics.html
21. Hamza, M., Akbar, M. A., & Capilla, R. (2023). *Understanding cost dynamics of serverless computing*. arXiv. https://doi.org/10.48550/arXiv.2311.13242
22. Lin, C., Ma, Y., & Shahrad, M. (2025). *Getting to the bottom of serverless billing*. arXiv. https://doi.org/10.48550/arXiv.2506.01283
23. Bodner, T., Radig, T., Justen, D., Ritter, D., & Rabl, T. (2025). *An empirical evaluation of serverless cloud infrastructure for large-scale data processing*. arXiv. https://doi.org/10.48550/arXiv.2501.07771
24. Soldani, J., Forti, S., Roveroni, L., & Brogi, A. (2025). Explaining microservices' cascading failures from their logs. *Software: Practice and Experience, 55*(5), 809-828. https://doi.org/10.1002/spe.3400
25. Molnar, V. (2025). *Event amplification risk in AWS serverless pipelines* [GitHub repository]. GitHub. https://github.com/j9mbo/event-amplification-risk-aws

Отримано редакцією журналу / Received: 27.02.26
Прорецензовано / Revised: 10.03.26
Схвалено до друку / Accepted: 25.06.26