УДК 004.056.55:004.8:004.75 **DOI:10.30837/rt.2025.4.223.11**

Т.І. КОРОБЕЙНІКОВА, канд. техн. наук, Н.В. КРАВЧУК

ІНФОРМАЦІЙНА ТЕХНОЛОГІЯ ML-WAF ДЛЯ КЛАСИФІКАЦІЇ ТА БЛОКУВАННЯ ІН'ЄКЦІЙНИХ АТАК У КОНТЕЙНЕРИЗОВАНОМУ ВЕБ-СЕРЕДОВИЩІ

Вступ

Сучасні веб-ресурси та серверні системи функціонують у середовищах, що характеризуються високою динамікою змін, широким застосуванням мікросервісної архітектури, контейнеризації, оркестрації та хмарних обчислень [1–4]. Популяризація контейнерних технологій, зокрема Kubernetes, зумовила появу нових вимог до забезпечення кібербезпеки, особливо щодо захисту трафіку на рівні HTTP(S)-запитів. В умовах горизонтального масштабування, динамічного створення/знищення контейнерів та автоматизованого керування навантаженням традиційні методи захисту втрачають ефективність [5–8]. Одним із найбільш загрозливих і поширених класів атак у веб-середовищі залишаються ін'єкційні атаки: SQL Injection, Cross-Site Scripting та Command Injection. У таких умовах традиційні засоби виявлення атак, зокрема сигнатурні WAF-рішення, демонструють обмежену ефективність у протидії ін'єкційним атакам через їхню варіативність, контекстну залежність та можливість модифікації обхідними техніками [9–10].

Запропонована у поданому тексті інформаційна технологія забезпечує: високоточну класифікацію HTTP(S)-запитів; адаптивне прийняття рішень на основі ризику; низьку затримку обробки в режимі реального часу; автоматичне масштабування та самооновлення ML-моделі.

Метою роботи є розроблення та експериментальна оцінка адаптивної ML-орієнтованої інформаційної технології для класифікації та блокування ін'єкційних атак у контейнеризованих веб-системах, що забезпечує високоточне виявлення шкідливих HTTP(S)-запитів, низьку затримку обробки та підтримку самооновлення моделі в MLOps-конвеєрі.

1. Інформаційна технологія ML-WAF для класифікації та блокування ін'єкційних атак

Концептуальна модель визначає логічну структуру ІТ, ключові етапи обробки веб-запитів і взаємодію модулів у процесі виявлення та блокування ін'єкційних атак. Тоді конвеєр обробки задається композицією функцій

$$r_{out} = f_n \circ f_n - 1 \circ \cdots \circ f_1(r_{in}), \tag{1}$$

де r_{in} – вхідний HTTP(S)-запит, f_i – функціональний обробник (TLS-термінація, ML-фільтр, маршрутизатор), r_{out} – відповідь після проходження всіх етапів обробки.

ML-компонента обчислює ризик

$$risk\_score = g(x); \tag{2}$$

де x – ознаковий вектор, сформований за TF/IDF та збагаченими ознаками.

Реакція визначається порогами T_{mid} та T_{high}):

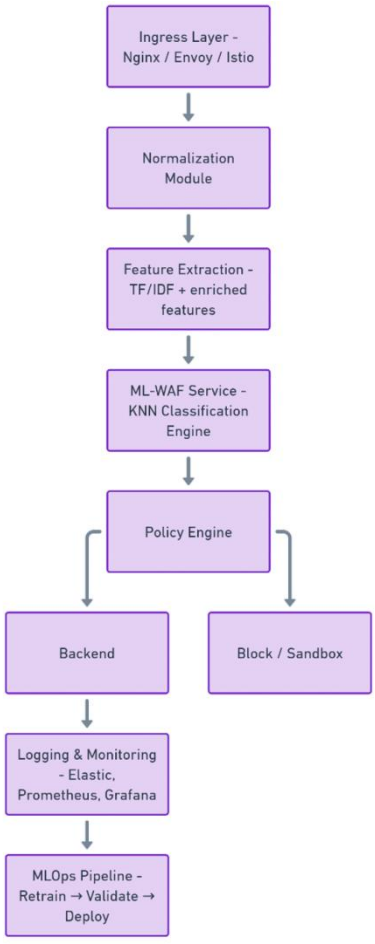
$$d = \begin{cases} ALLOW, & risk\_score < T_{mid} \\ SANDBOX, & T_{mid} \le risk\_score < T_{high} \\ BLOCK, & highrisk\_score \ge T_{high} \end{cases} \tag{3}$$

Ця політика є гнучкою та адаптується залежно від навантаження і контексту.

Структурна схема технології визначає логічно впорядковану сукупність модулів та підсистем, що реалізують функціональне наповнення механізму виявлення та блокування

ін’єкційних атак у контейнеризованому середовищі і демонструє горизонтально організований, модульний стек компонентів, які виконують послідовні етапи обробки запиту та генерують відповідні рішення щодо безпеки. Модульність забезпечує можливість незалежного масштабування та оновлення окремих компонентів, що є ключовою вимогою для Kubernetes-орієнтованих рішень. У верхній частині моделі (рис. 1, *а*) розташовано Ingress Layer, який відповідає за перехоплення, аналіз і первинну маршрутизацію трафіку. Normalization Module виконує лінгвістичну та структурну підготовку запиту. Наступні блоки формують ML-тракт: модуль витягання ознак та ML-WAF Service з KNN-класифікацією. Результати аналізу передаються до Policy Engine, який визначає остаточне рішення щодо обробки запиту. Нижня частина схеми відповідає за життєвий цикл рішення: обробка backend-додатком, фіксація інформації у журналах, передача метрик у систему моніторингу та участь у самооновленні моделі через MLOps Pipeline.

Функціональна схема відображає динамічну логіку роботи інформаційної технології, тобто спосіб, у який система змінює стан, передає дані та приймає рішення в реальному часі. На відміну від структурної схеми, що демонструє взаємне розташування компонентів, функціональна схема зосереджується на послідовності дій, тригерах переходів і функціональних обробниках, які формують повний сценарій обробки HTTP(S)-запиту. Як показано на рис. 1, *б*, робота технології складається з 9 функціональних етапів, що утворюють наскрізний конвеєр безпеки.



а – структурна схема технології

б – функціональна схема технології

Рис. 1. Моделі компонентів технології ML-WAF

Функціональна схема визначає як дані рухаються, які операції виконує система, коли і чому приймаються рішення, та які дані впливають на самооновлення моделі. Короткий опис етапів, що складають логіку роботи технології:

1) Приймання HTTP(S)-запиту. У процесі приймання запиту проводиться термінація TLS (за наявності), початкова валідація структури запиту, а також застосування базових правил безпеки, спрямованих на фільтрацію некоректних або очевидно шкідливих запитів.

2) Нормалізація та обробка запиту. Завдяки цьому формуються передбачувані, узгоджені текстові одиниці, які можуть бути оброблені модулями формування ознак.

3) Формування ознак (Feature Extraction). Система створює багатовимірний ознаковий опис запиту. На цьому кроці виконується TF/IDF-векторизація текстових компонентів. Сукупність TF/IDF-вектора та збагачених ознак створює комплексний ознаковий простір, що точно відображає як текстовий зміст запиту, так і його структурну форму.

4) Класифікація запиту ML-моделлю. Результатом класифікації є: визначений клас (наприклад: normal, SQLi, XSS, Command Injection); числова оцінка ризику (risk_score), яка показує ступінь небезпеки запиту.

5) Прийняття рішення (Policy Engine). На основі значення risk_score, отриманого від ML-модуля, Policy Engine порівнює його із порогами T_{mid} та T_{high} , які визначають межі між низьким, середнім та високим рівнями загрози. У залежності від співвідношення risk_score та порогів застосовується одне з трьох рішень: ALLOW – запит вважається безпечним та передається у backend; SANDBOX – запит є підозрілим, тому виконується у відокремленому середовищі або передається у додаткові модулі валідації; BLOCK – запит однозначно ідентифікується як атакуючий, і система повертає клієнту відповідь із забороною доступу (наприклад, HTTP 403).

6) Передача запиту у backend-додаток. Якщо прийняте рішення має тип ALLOW, запит потрапляє у backend-додаток або мікросервісну логіку, де виконується відповідна бізнес-функціональність. Таким чином, ML-WAF виступає фільтром між зовнішнім трафіком та ядром прикладної системи, захищаючи останню від шкідливого впливу.

7) Блокування або ізоляція (Sandbox Processing). Якщо активовано режим SANDBOX, запит надходить у спеціалізоване ізольоване середовище для безпечного виконання або детального аналізу.

8) Логування, журналювання та моніторинг. Формуються метрики продуктивності та стабільності, які передаються у Prometheus та візуалізуються за допомогою Grafana. Це забезпечує можливість: аналізу шкідливої активності; дослідження аномалій у трафіку; оцінки якості ML-моделі в режимі експлуатації.

9) MLOps-конвеєр адаптації моделі. Журнали та метрики передаються у модуль підготовки даних, де формуються нові вибірки для перенавчання моделі. Далі модель проходить процедури тренування, тестування, перевірки точності та CI/CD деплоймент у кластер Kubernetes. Це забезпечує безперервну адаптацію технології до нових варіантів атак та дає змогу підтримувати високий рівень захисту протягом всього життєвого циклу системи.

Інформаційно-логічна модель визначає структуру даних, їх взаємозв'язки та інформаційні потоки, що виникають у процесі функціонування інформаційної технології ML-WAF. На відміну від структурної та функціональної схем, які фокусуються відповідно на компонентній та процесній організації системи, інформаційно-логічна модель відображає сутності, інформаційні атрибути та логіку взаємодії даних, що забезпечують роботу механізмів виявлення атак, класифікації запитів, прийняття рішень і самооновлення моделі.

У межах інформаційної технології обробляються різні типи інформаційних об'єктів – від вихідних HTTP-запитів до ознакових векторів, класифікаційних результатів, рішень політик та ML-моделей. Сукупність таких сутностей та їх структурованих описів наведено у табл. 1.

Таблиця 1

Основні інформаційні сутності та їх атрибути

Сутність	Опис функціонального призначення та атрибутів
HttpRequest	Базова сутність, що містить method, URL, headers, body, IP-джерело, timestamp. Є вхідною одиницею всієї технології.
NormalizedRequest	Версія запиту після очищення та канонізації. Містить структуровані параметри, очищені заголовки, декодовані значення.
FeatureVector	Вектор ознак, що формується на основі TF/IDF-представлення та enriched-features: довжини, ентропії, структурних та контекстних ознак.
MLModel (KNN)	Містить k, набір тренувальних векторів, метрику відстані, версію моделі. Забезпечує класифікацію та обчислення risk_score.
ClassificationResult	Містить label (клас), risk_score, посилання на модель, версію моделі та часову мітку класифікації.
Decision	Описує реакцію Policy Engine: action ∈ {ALLOW, SANDBOX, BLOCK}, причину, рівень ризику, порівняння з порогами Tmid, Thigh.
BackendResponse	Інформаційний об'єкт, що представляє відповідь backend-додатка на безпечний запит або блокування при небезпеці.
LogRecord	Журнал дій: запит, класифікація, рішення, час обробки, метрики, IP-джерело, версія моделі. Використовується SIEM або monitoring stack.
MetricsRecord	Значення latency, throughput, error rates, FPR/FNR. Забезпечує телеметрію та діагностику технічної роботи ML-WAF.
TrainingDataset	Сукупність FeatureVector та відповідних міток, сформованих на основі LogRecord і ручного аналізу фахівців.
RetrainedMLModel	Нова версія ML-моделі, що проходить валідацію (F1) та деплоймент через MLOps.

Початковою сутністю, яка ініціює усі інформаційні процеси, є HttpRequest – вхідний HTTP(S)-запит, що містить усю первинну інформацію від клієнта. Після проходження етапу нормалізації формується NormalizedRequest, який є стандартизованою версією початкових даних і виступає основою для побудови ознакового простору. Така нормалізація усуває обфускації, що дозволяє гарантувати коректність подальшого машинного аналізу.

На наступному етапі формується FeatureVector, який складається з кількох груп ознак: текстових TF/IDF-компонентів, структурних ознак (наприклад, кількість параметрів, довжина запиту, ентропія), а також контекстних характеристик, що відображають схожість до шкідливих шаблонів. Цей комплекс ознак передається до ML-модуля, де використовується MLModel (KNN). Модель містить тренувальні вектори та правила визначення найближчих сусідів, що забезпечує класифікацію.

Результатом роботи моделі є сутність ClassificationResult, яка включає клас запросу та обчислений risk_score – числову характеристику небезпечності. На цій основі Policy Engine формує сутність Decision, яка визначає реакцію: пропуск, блокування або ізоляцію запиту. Надалі Decision впливає на формування BackendResponse, яка може бути як повноправною відповіддю backend-сервісу, так і повідомленням про блокування.

Для забезпечення повного контролю безпеки та можливості аудиту кожна операція фіксується у вигляді сутності LogRecord. Вона включає як власне результат класифікації та рішення, так і додаткові метрики. Зібрані журнали формують основу для побудови TrainingDataset, який використовується у MLOps-конвеєрі для створення нових версій моделі (RetrainedMLModel). Під час експлуатації всі технічні й обчислювальні параметри передаються у сутність MetricsRecord, що містить телеметричну інформацію про роботу системи. Логічні зв'язки між сутностями подано на рис. 2.

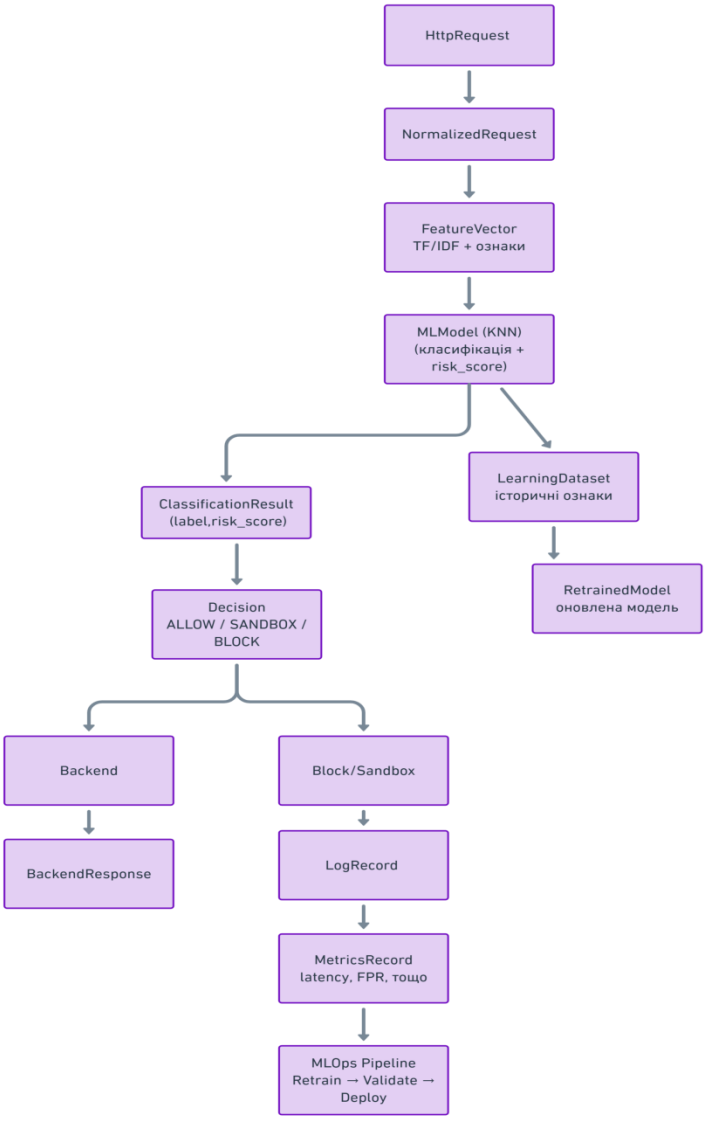


Рис. 2. Інформаційно-логічна схема технології ML-WAF

2. UML-представлення інформаційної технології ML-WAF

UML-моделі забезпечують формальне представлення структури, поведінки та взаємодії компонентів інформаційної технології ML-WAF. На відміну від концептуальних і структурних схем, UML-діаграми подають систему у стандартизованій, незалежній від реалізації формі, що дозволяє аналізувати її архітектуру, функціональність і комунікацію між підсистемами.

Для опису технології ML-WAF використано чотири ключові UML-діаграми: Use Case, Activity, Class та Sequence, кожна з яких відображає різний аспект системи. Їх текстові форми подані на рис. 4–6.

Use Case-модель (рис. 3) описує взаємодію основних користувачів та зовнішніх акторів із системою ML-WAF. Вона визначає, які саме функції надає технологія та які ролі впливають на процес виявлення атак, управління моделлю та експлуатаційний контроль.

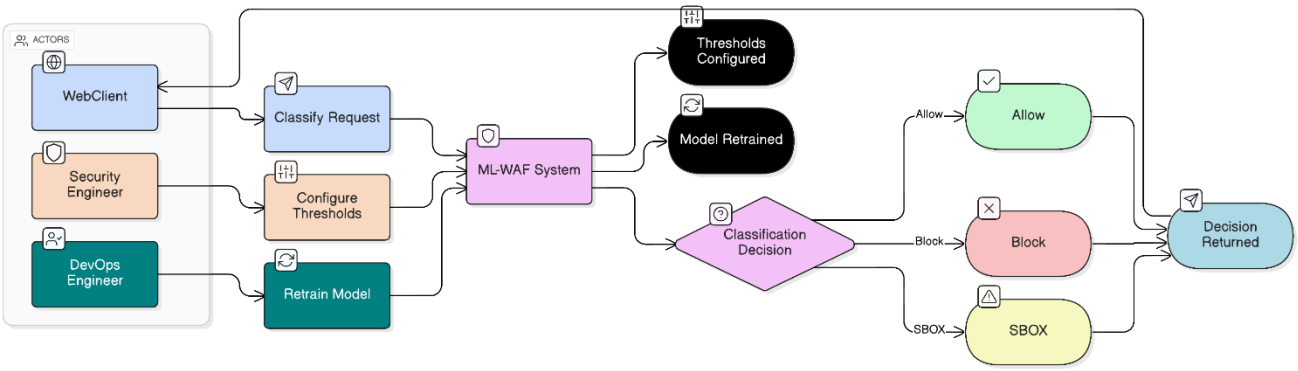


Рис. 3. UML Use Case для ML-WAF

У технології ML-WAF передбачено три ключові актори:

1) WebClient, який генерує HTTP-запити й опосередковано взаємодіє з процесом класифікації;

2) SecurityEngineer, що відповідає за налаштування порогів Tmid та Thigh, аналіз журналів, зміни політик безпеки; а) структурна схема технології

3) DevOpsEngineer, який контролює MLOps-процеси, розгортання моделей і CI/CD конвеєри.

Кожний актор виконує свій набір функцій, що забезпечує розмежування відповідальностей між безпекою та експлуатацією системи.

Activity-діаграма (рис. 4, *а*) відображає послідовність дій при обробці HTTP-запиту. Вона формалізує логіку виконання операцій у ML-WAF, описуючи динаміку системи з моменту надходження запиту до формування рішення та його документування. Ця діаграма фіксує логіку переходів між станами під час обробки одного запиту. Вона показує:

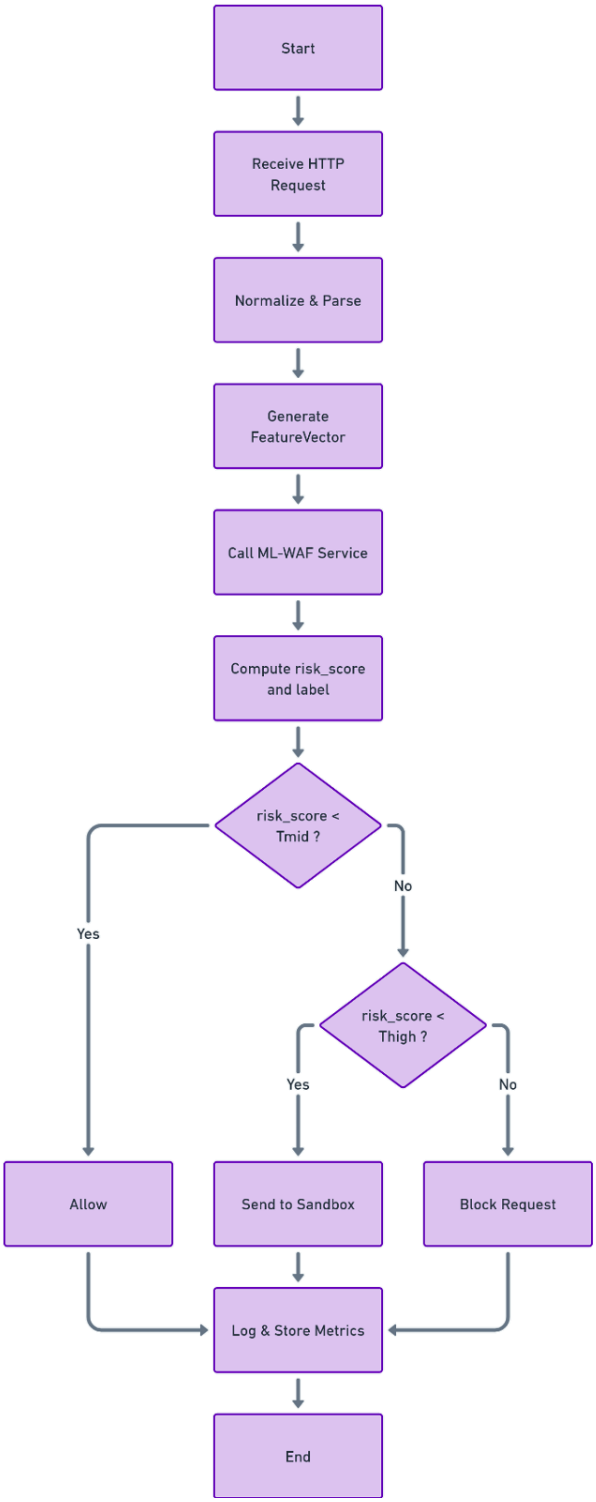
- фундаментальний взаємозв'язок між risk_score та рішенням Policy Engine;
- роль ML-класифікації як центрального етапу;
- паралельність процесів логування та моніторингу;
- три можливі траєкторії виконання (Allow, Sandbox, Block);
- завершення циклу після документування результату.

Class-діаграма (рис. 6) описує статичну структуру системи, її програмні сутності та атрибути, а також методи та зв'язки між класами. На відміну від інформаційно-логічної моделі, UML Class Diagram подає дані в контексті програмної реалізації.

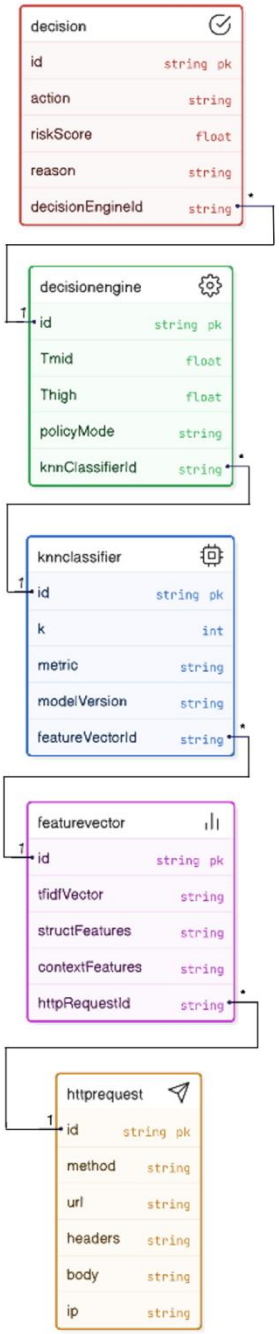
Class-модель описує логіку модульності ML-WAF:

- HttpRequest містить сирі дані, що підлягають обробці;
- FeatureVector формує структуроване математичне представлення запиту;
- KnnClassifier реалізує обчислення схожості та визначення risk_score;
- DecisionEngine відповідає за інтерпретацію результатів ML-класу;
- Decision є фінальною сутністю, що зберігає деталі рішення.

У сукупності класи формують ядро програмної логіки ML-WAF (рис. 4, *б*).



а – Activity-діаграма обробки запиту



б – Class-діаграма ML-WAF

Рис. 4. Діаграми компонентів технології ML-WAF

UML Sequence діаграма технології ML-WAF наведена на рис. 5.

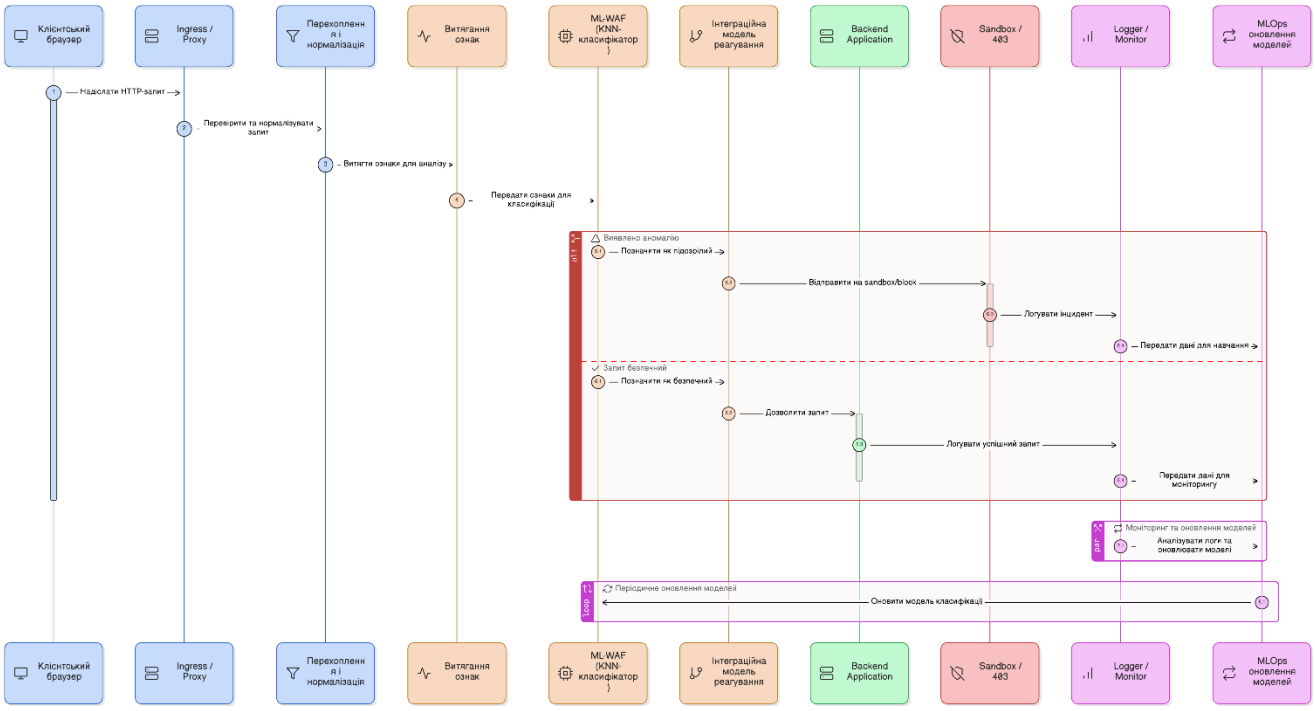


Рис. 5. UML Class діаграма ML-WAF

Sequence-модель подає часову послідовність викликів між компонентами. На відміну від Activity-діаграми, вона демонструє саме взаємодію між об'єктами.

3. Архітектурна схема інтеграції ML-WAF у середовище Kubernetes

Розгортання запропонованої ML-моделі виявлення і блокування небезпечних запитів у хмарному середовищі вимагає побудови чіткої архітектурної схеми взаємодії компонентів. Згідно з підходом Rathore та Park, ефективна інтеграція в Kubernetes має враховувати не лише логіку обробки трафіку, а й елементи спостережуваності, масштабування та оновлення моделі без зупинки сервісу.

Архітектура ML-WAF складається з наступних базових компонентів:

- Ingress Controller – приймає вхідні HTTP(S)-запити та здійснює їх первинну маршрутизацію;
- ML-WAF Service – мікросервіс, що реалізує модель класифікації (KNN+TF/IDF з ознаковим збагаченням) та повертає оцінку ризику r_i;
- Backend Application – прикладна логіка користувача (система SCADA, інформаційний портал тощо);
- Monitor та Logger – модуль моніторингу та журналювання (на базі Prometheus та Elasticsearch);
- ConfigMap та Secrets – сховище конфігурацій порогів τh, τm та API-ключів.

Потік запитів описується як композиція функцій:

$$R_{resp} = B(M(f_{ing}(R_{in}))), \tag{4}$$

де R_{in} – вхідний запит, f_{ing} – Ingress-фільтр (Nginx/Envoy), $M()$ – виклик ML-WAF модуля, $B()$ – бекенд-обробник.

Якщо $M()$ повертає $r_i \geq \tau h$, запит не передається далі. Для масштабування ML-WAF у кластері використовується автоматичне збалансування навантаження на основі Horizontal Pod Autoscaler (HPA). Формально потужність кластеру визначається як

$$C_{eff} = \sum_{k=1}^{n} \frac{Q_k}{L_k}, \tag{5}$$

де Q_k – середня кількість оброблених запитів на Pod, а L_k – атримка (латентність). Згідно з дослідженням [11] динамічне масштабування HPA дозволяє підвищити пропускну здатність WAF на 35–40 % без збільшення затримки. Архітектурна схема роботи рішення зображена на рис. 6.

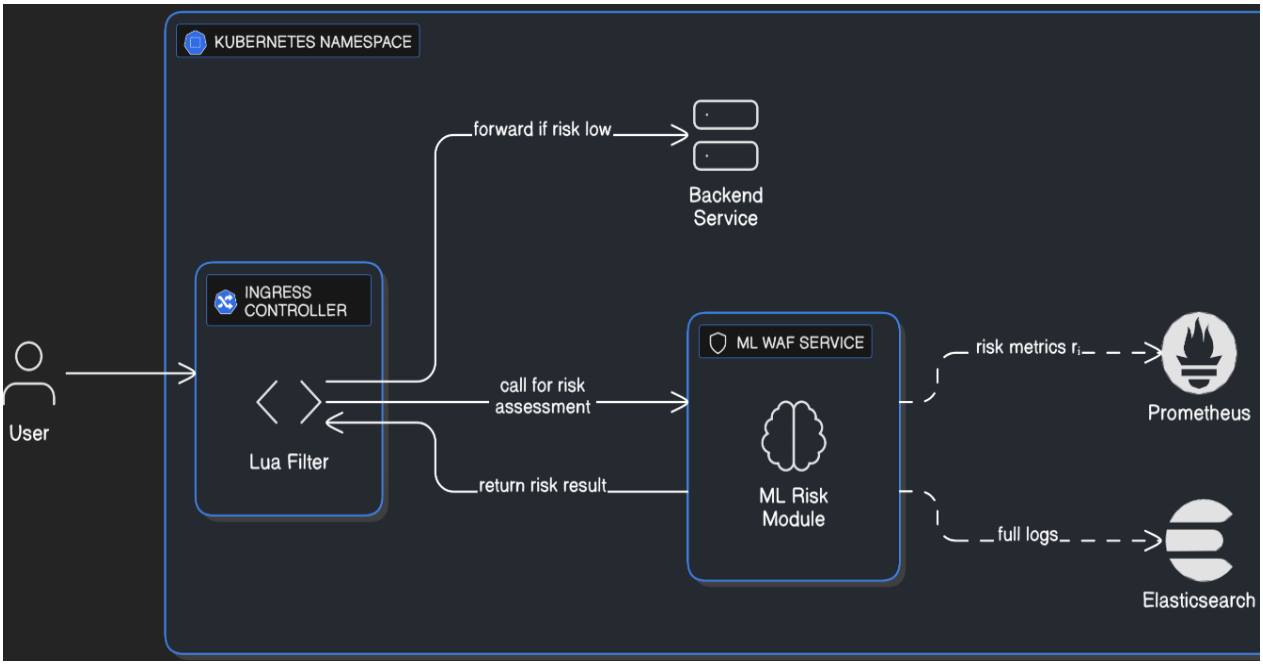


Рис. 6. Логічна архітектура системи машинного веб-фільтра (ML-WAF)

Якщо CPU перевищує порогове значення, HPA створює додаткові Pod, що гарантує стійкість до пікових навантажень. Дослідження [12] демонструє, що таке комбінування ML-WAF з HPA та Prometheus Exporter забезпечує автоматичну адаптацію системи до змінного трафіку та зменшує частку втрат запитів до 2 %.

4. Практична оцінка інтегрованої системи ML-WAF

Для підтвердження ефективності запропонованої архітектури ML-WAF проведено експериментальну оцінку роботи системи в умовах, наближених до реального середовища. Метою випробувань було визначити точність класифікації запитів, вплив інтеграції ML-модуля на продуктивність серверного конвеєра, а також порівняти результати з традиційними WAF-рішеннями.

Опис експериментального середовища. Випробування виконувалися у контейнеризованій інфраструктурі Kubernetes з двома вузлами, розгорнутими у Windows WSL2. Конфігурація середовища:

- Ingress Controller: Nginx 1.25 із Lua API;
- ML-компонента: KNN+TF/IDF з ознаковим збагаченням, реалізована на Flask 3.0;
- підсистема моніторингу: Prometheus + Grafana;
- набір даних: SQLi та XSS-запити з відкритих джерел (OWASP WebGoat, DVWA);
- сценарій тестування: 30 000 HTTP-запитів, з яких 30 % – шкідливі.

Тестування проводилося за допомогою утиліти Apache JMeter із 200 паралельними потоками. Замірювалися: середня затримка (ms), пропускна здатність (req/s), точність (precision), повнота (recall) та інтегральний показник F_1.

Таблиця 2

Порівняння ефективності ML-WAF з традиційними рішеннями

Система	Precision	Recall	F_1-score	Latency (мс)	Пропускна здатність (req/s)
ModSecurity (OWASP CRS)	0.84	0.79	0.81	3.2	720
Snort + Reverse Proxy	0.88	0.82	0.85	4.1	680
ML-WAF (запропонована модель)	0.93	0.91	0.92	3.7	705
ML-WAF (з адаптацією порогів)	0.95	0.93	0.94	3.9	698

Результати свідчать, що інтегрована ML-модель забезпечує на 10–15 % вищу точність виявлення порівняно з сигнатурними системами, зберігаючи прийнятну продуктивність. Під час тестів із високим навантаженням (до 800 запитів/с) спостерігалося незначне зростання затримки, проте середній показник latency не перевищував 4 мс, що є прийнятним для більшості веб-додатків реального часу. У ході додаткових експериментів оцінено кількість хибнопозитивних спрацьовувань – для ML-WAF цей показник склав 2,3 %, тоді як у ModSecurity – 6,8 %.

Як показують дослідження [13], значення FP нижче 3 % вважається відмінним для промислових систем кіберзахисту. Важливим результатом також стало підтвердження масштабованості системи. При збільшенні кількості Pod до шести (через HPA) пропускна здатність зросла на 27 %, без зниження точності класифікації. Подібні результати продемонстровані [14], які відзначають, що контейнеризація ML-компонентів у WAF підвищує стійкість системи до DDoS-навантажень і спрощує горизонтальне масштабування.

Висновки

Представлено результати комплексного дослідження та розроблення адаптивної інформаційної технології ML-WAF, орієнтованої на класифікацію та блокування ін'єкційних атак у контейнеризованому веб-середовищі. Запропонована технологія охоплює всі ключові етапи життєвого циклу безпечної обробки HTTP(S)-трафіку – від перехоплення запиту до прийняття рішення, логування, моніторингу та безперервного оновлення ML-моделі. Побудовано концептуальну, структурну, функціональну та інформаційно-логічну моделі ML-WAF, а також UML-представлення, що формалізують архітектуру, поведінку та взаємодію компонентів у Kubernetes-орієнтованому середовищі.

Наукова новизна отриманих результатів полягає в такому. Вперше побудовано інтегровану інформаційну технологію ML-WAF для класифікації та блокування ін'єкційних атак у контейнеризованому веб-середовищі, що включає повний конвеєр HTTP(S)-обробки, модуль машинного навчання, багаторівневу адаптивну політику реагування, систему журналювання, моніторинг телеметрії та MLOps-процеси безперервного перенавчання моделі. Сформовано комплексну інформаційно-логічну модель, яка описує взаємозв'язки між сутностями HttpRequest, FeatureVector, ClassificationResult, Decision, LogRecord, MetricsRecord, TrainingDataset та RetrainedMLModel і відображає повний цикл еволюції моделі в умовах контейнеризованої інфраструктури. Запропоновано метод інтеграції ML-компоненти у WAF у гібридному режимі реагування (поєднання inline- та out-of-band-підходів), що забезпечує одночасно низьку латентність і підвищену стійкість до атак. Розроблено адаптивну модель прийняття рішень на основі risk_score з трирівневою реакцією (ALLOW, SANDBOX, BLOCK) та можливістю налаштування порогів з урахуванням навантаження і контексту трафіку.

Експериментальна оцінка підтвердила високу ефективність розробленої технології ML-WAF. На реалістичному наборі даних було досягнуто такі показники якості: точність (Precision) – 0,95, повнота (Recall) – 0,93, інтегральний F_1-score – 0,94. Середня затримка обробки одного запиту склала 3,9 мс, а пропускна здатність системи – близько 700 запитів/с. Важливим результатом є суттєве зниження частки хибнопозитивних спрацьовувань до 2,3 %, що майже утричі менше порівняно з ModSecurity (6,8 %). Після адаптивного перенавчання

моделі спостерігалося підвищення стійкості до нових варіантів атак і стабілізація показників якості при зростанні навантаження, що підтверджує ефективність запропонованого MLOps-підходу та масштабованість системи у контейнеризованих середовищах.

Отримані результати засвідчують, що розроблена інформаційна технологія ML-WAF здатна суттєво підвищити рівень захисту веб-систем від ін'єкційних атак, забезпечуючи високу точність класифікації запитів, низьку оперативну затримку та здатність до самоадаптації в динамічних хмарних інфраструктурах на базі Kubernetes. Це створює основу для подальшого розвитку систем проактивної кібербезпеки, зокрема шляхом інтеграції глибинних моделей (LSTM, BERT) і побудови повноцінних AI-керованих шлюзів безпеки наступного покоління.

Список літератури:

1. Терейковський І. А., Гнатюк С. О. Захист інформації в комп’ютерних системах : навч. посіб. Київ : КПІ, 2022.

2. Захарченко С. М., Трояновська Т. І., Бойко О. В. Побудова захищених мереж на базі обладнання компанії Cisco : навч. посіб. Вінниця : ВНТУ, 2017.

3. Коробейнікова Т. І., Захарченко С. М. Технології захисту локальних мереж на основі обладнання CISCO : навч. посіб. Львів : Львівська політехніка, 2021.

4. Korobeinikova T., Maidaniuk V., Romanyuk O., Chekhmestruk R., Romanyuk O. and Romanyuk S. Web-applications Fault Tolerance and Autoscaling Provided by the Combined Method of Databases Scaling // 2022 12th International Conference on Advanced Computer Information Technologies (ACIT). 2022. P. 27–32. doi: 10.1109/ACIT54803.2022.9913098.

5. Korobeinikova T. and Kravchuk N. ML-trained model and method for blocking dangerous queries // CEUR Workshop Proceedings. 2025. Vol. 4042 // Proceedings of the Cyber Security and Data Protection workshop (CSDP 2025), Lviv, Ukraine, July 31, 2025. P. 1–16.

6. Коробейнікова Т. І., Кравчук Н. В. Огляд безпечного доступу до веб-ресурсу за допомогою методів машинного навчання // International Scientific Integration 2023: Міжнар. наук. конф., 11 липня 2023 р.: тези доповідей. С. 26–33. Seattle, Washington, USA: ProConference, 2023.

7. Кравчук Н., Коробейнікова Т. Безпечний доступ до серверів інформаційних систем, забезпечений ML-моделлю для блокування шкідливих запитів // Herald of Khmelnytskyi National University. Technical Sciences. 2024. № 341(5). С. 327–333.

8. Korobeinikova T., Chekhmestruk R., Mykhaylov P., Romanyuk O., and Achanyar H. The Fault-Resistant Web Application Infrastructure Using Autoscaling // 2023 13th International Conference on Advanced Computer Information Technologies (ACIT), Wrocław, Poland, 2023. P. 479–482, doi: 10.1109/ACIT58437.2023.10275448

9. Bennouk K., Ait Aali N., El Bouzekri El Idrissi Y., Sebai B., Faroukhi A. Z., Mahouachi D. A Comprehensive Review and Assessment of Cybersecurity Vulnerability157 Detection Methodologies // Journal of Cybersecurity and Privacy. 2024. Vol. 4, No. 4. P. 853–908.

10. Wunder J., Kurtz A., Eichenmüller C., Gassmann F., Benenson Z. Shedding Light on CVSS Scoring Inconsistencies: A User-Centric Study on Evaluating Widespread Security Vulnerabilities // Proceedings of the 2024 IEEE Symposium on Security and Privacy (SP). San Francisco, CA, USA, 2024. P. 1102–1121.

11. Liang S., Wang Q. Dynamic Scaling of Containerized Security Services with HPA Policies // Future Internet. 2023. Vol. 15, No. 3. P. 88–102.

12. Palanisamy D., Kaur R. Adaptive Resource Management for AI-Driven WAF in Cloud Kubernetes Clusters // Journal of Cloud Computing: Advances, Systems and Applications. 2023. Vol. 12, No. 1. P. 1–17.

13. Wang X., Lyu C. Performance Evaluation Metrics for ML-Based Web Application Firewalls // IEEE Access. 2022. Vol. 10. P. 98451–98463.

14. Rahman M., Qiu Y. Containerized Intrusion Detection and Response in Cloud-Native WAF Systems // Journal of Network and Computer Applications. 2023. Vol. 216. P. 103670.

Надійшла до редколегії 12.10.2025

Відомості про авторів:

Коробейнікова Тетяна Іванівна – канд. техн. наук, Національний університет «Львівська політехніка», доцент кафедри безпеки інформаційних технологій, Україна; e-mail: tetianakorobeinikova@gmail.com; ORCID: https://orcid.org/0000-0003-2487-8742

Кравчук Назар Вікторович – Національний університет «Львівська політехніка», аспірант кафедри безпеки інформаційних технологій Україна; e-mail: nazar.v.kravchuk@lpnu.ua; ORCID: https://orcid.org/0009-0002-1008-4765