

Харківський національний університет радіоелектроніки
Міністерство освіти і науки України

Кваліфікаційна наукова
праця на правах рукопису

СВИРИДОВ АРТЕМ СЕРГІЙОВИЧ

УДК 004.056.53:004.75:004.8

ДИСЕРТАЦІЯ
**МОДЕЛЬ ТА МЕТОДИ ІНТЕЛЕКТУАЛЬНОЇ СИСТЕМИ
ВІЯВЛЕННЯ АНОМАЛІЙ В ХМАРНИХ СЕРВІСАХ**

Спеціальність: 125 Кібербезпека

Галузь знань: 12 Інформаційні технології

Подається на здобуття ступеня доктора філософії

Дисертація містить результати власних досліджень. Використання ідей, результатів, текстів інших авторів мають посилання на відповідне джерело



А.С. Свиридов

Науковий курівник
Северінов Олександр Васильович
кандидат технічних наук, доцент

Харків – 2026

АНОТАЦІЯ

Свиридов А.С. Модель та методи інтелектуальної системи виявлення аномалій в хмарних сервісах – Кваліфікаційна наукова праця на правах рукопису.

Дисертація на здобуття ступеня доктора філософії за спеціальністю 125 Кібербезпека – Харківський національний університет радіоелектроніки, Міністерство освіти і науки України, Харків, 2026.

Дисертаційну роботу присвячено розробці моделі та удосконаленню методів інтелектуальної системи виявлення аномалій у хмарних сервісах на основі використання методів штучного інтелекту та машинного навчання. У межах дослідження сформовано комплексний підхід, що поєднує механізми збору й інтеграції даних із API-журналів, Kubernetes-кластерів, поведінкових сценаріїв користувачів і компонентів мікросервісної архітектури, методи попередньої обробки та аналізу даних, а також алгоритми машинного і глибокого навчання, зокрема LSTM Autoencoder, для виявлення аномальної активності. Запропонована модель забезпечує формування структурованого набору ознак, автоматизоване навчання інтелектуальних моделей, багаторівневий моніторинг подій у контейнеризованому середовищі, виявлення кіберзагроз і поведінкових відхилень у реальному часі, а також оцінювання ефективності функціонування системи за допомогою стандартних метрик якості класифікації. Це дозволяє підвищити рівень безпеки, надійності та адаптивності хмарних програмних систем за умов динамічних кіберзагроз.

Актуальність роботи зумовлена стрімким розвитком хмарних технологій, широким впровадженням мікросервісної архітектури, контейнеризації та оркестрації застосунків, що супроводжується зростанням складності програмних систем і підвищенням рівня кіберзагроз. Сучасні хмарні середовища генерують значні обсяги різномірних даних, зокрема API-журнали, мережеві події, телеметрію контейнерів і поведінкові дані користувачів, аналіз яких традиційними методами часто не забезпечує

своєчасного виявлення аномалій, вторгнень і прихованих загроз. Існуючі підходи до моніторингу безпеки переважно мають обмежену адаптивність до динамічних змін середовища, складно масштабуються та недостатньо ефективно враховують поведінкові особливості сучасних розподілених систем. Використання методів штучного інтелекту та машинного навчання дозволяє автоматизувати процес аналізу великих обсягів даних, виявляти приховані закономірності, поведінкові відхилення та аномальні патерни, що підвищує рівень безпеки, надійності та стійкості хмарних програмних застосунків.

Метою дисертаційної роботи є підвищення ефективності виявлення аномалій у хмарних сервісах шляхом розробки моделі і методів інтелектуальної інформаційної системи, що забезпечують інтеграцію різнорідних даних, автоматизований аналіз подій та адаптивне виявлення кіберзагроз у контейнеризованих і розподілених середовищах.

Об'єкт дослідження – процеси збору, обробки та аналізу журналів подій, API-трафіку, поведінкових характеристик користувачів і телеметрії хмарної інфраструктури в системах виявлення аномалій.

Предмет дослідження – моделі та методи інтелектуальної інформаційної технології виявлення аномалій у хмарних сервісах на основі алгоритмів машинного та глибокого навчання, спрямовані на підвищення точності виявлення загроз, адаптивності систем безпеки та зниження ризиків функціонування програмних систем в умовах динамічних кіберзагроз.

У першому розділі дисертації проведено системний аналіз сучасних теоретичних і прикладних підходів до забезпечення безпеки хмарних сервісах та виявлення аномалій у розподілених програмних середовищах. Розглянуто класичні та сучасні методи моніторингу, аналізу журналів подій, систем виявлення аномалій, поведінкового аналізу користувачів, а також підходи, що базуються на статистичних методах, сигнатурному аналізі та алгоритмах машинного навчання. Визначено основні обмеження традиційних засобів виявлення аномалій в умовах високої динамічності хмарної інфраструктури,

масштабованості мікросервісної архітектури, контейнеризації та постійної зміни поведінкових патернів користувачів і сервісів. Окрему увагу приділено аналізу загроз, пов'язаних із API-взаємодією, Kubernetes-кластерами, мережевою активністю та поведінковими відхиленнями, а також проблемам обробки великих обсягів різнорідних даних у реальному часі. Проаналізовано переваги та недоліки існуючих підходів до виявлення кіберзагроз, включаючи сигнатурні, евристичні та інтелектуальні моделі, з урахуванням їх точності, адаптивності та здатності до масштабування. На основі проведеного аналізу обґрунтовано необхідність використання методів штучного інтелекту, машинного та глибокого навчання для підвищення ефективності виявлення аномалій у хмарних сервісах, а також сформульовано основні завдання дослідження, спрямовані на розробку функціональної моделі інтелектуальної системи, удосконалення методів аналізу API-журналів, поведінкових сценаріїв і контейнеризованого середовища та створення програмної реалізації комплексної системи моніторингу й виявлення загроз.

У другому розділі дисертації розроблено метод виявлення аномалій у хмарних сервісах на основі методів машинного та глибокого навчання. Розглянуто сучасні підходи до виявлення аномалій у API-журналах, поведінці користувачів та Kubernetes-кластерах із використанням нейронних мереж і алгоритмів аналізу часових рядів, які дозволяють формалізувати процес виявлення загроз у вигляді аналізу послідовностей подій та поведінкових шаблонів у хмарному середовищі. Проаналізовано механізми формування набору ознак з урахуванням характеристик API-запитів, мережевої активності, поведінкових параметрів користувачів та телеметрії контейнеризованих сервісів, а також способи врахування особливостей розподіленої мікросервісної архітектури. Запропоновано удосконалений метод виявлення аномалій, що поєднує процедури попередньої обробки журналів подій, аналізу поведінкових сценаріїв, формування часових послідовностей та автоматизованого виявлення відхилень за допомогою LSTM Autoencoder. Метод базується на інтеграції часових, поведінкових і мережевих

характеристик у процес навчання моделі, що забезпечує адаптивність системи до змін середовища, підвищення точності виявлення аномальної активності та здатність виявляти невідомі типи кіберзагроз у реальному часі.

У третьому розділі дисертації розроблено метод виявлення аномалій у Kubernetes-кластерах на основі використання моделей глибокого навчання. В основу дослідження покладено аналіз журналів подій і телеметрії контейнеризованого середовища, що дозволило сформувати систематизований підхід до виявлення аномальної активності в хмарній інфраструктурі. Розроблено структурну схему методу, яка поєднує етапи збору даних із Kubernetes-кластерів, попередньої обробки журналів, формування векторів ознак та аналізу часових залежностей у поведінці системи. Описано структуру та характеристики наборів даних, а також обґрунтовано вибір параметрів середовища для проведення експериментальних досліджень. Значну увагу приділено процедурі підготовки даних, що включає очищення журналів подій, нормалізацію параметрів, перетворення часових послідовностей та формування наборів ознак для навчання моделі. Етап моделювання передбачає застосування LSTM Autoencoder для автоматизованого виявлення аномалій у поведінці контейнеризованих сервісів та мережевої активності Kubernetes-кластера. Модель навчалася на нормальних сценаріях функціонування системи, що дозволило визначати відхилення від типової поведінки за величиною похибки реконструкції. Оцінювання ефективності здійснювалося за допомогою метрик Accuracy, Precision, Recall та F1-score. Отримані результати підтвердили ефективність використання LSTM Autoencoder для виявлення аномалій у Kubernetes-середовищі, оскільки запропонований підхід забезпечив підвищення точності виявлення загроз та адаптивність системи до динамічних змін хмарної інфраструктури.

У четвертому розділі дисертації розроблено модель інтелектуальної системи виявлення аномалій у хмарних сервісах та реалізовано її програмну архітектуру. Запропоновано функціональну модель, яка формалізує повний

цикл обробки інформації від збору API-журналів, мережевих подій і телеметрії Kubernetes-кластерів до формування набору ознак, навчання інтелектуальних моделей та автоматизованого виявлення аномальної активності. Модель визначає взаємозв'язки між модулями збору та агрегації даних, попередньої обробки журналів, аналізу поведінкових характеристик, виявлення загроз і моніторингу подій, що дозволяє забезпечити системність і цілісність процесу аналізу безпеки хмарного середовища. Окрему увагу приділено програмній реалізації запропонованої моделі. Реалізована система інтегрує модулі збору та підготовки даних, алгоритми машинного й глибокого навчання, механізми аналізу часових послідовностей та засоби візуалізації результатів моніторингу. Програмна архітектура забезпечує автоматизоване виявлення аномалій, аналіз поведінки користувачів і сервісів, моніторинг стану Kubernetes-кластерів та формування повідомлень про потенційні кіберзагрози у реальному часі. Проведена експериментальна перевірка підтвердила працездатність системи та її здатність адаптуватися до змін у хмарній інфраструктурі й поведінці сервісів. Таким чином, розроблена модель та її програмна реалізація створюють практичну основу для побудови адаптивних інтелектуальних систем моніторингу безпеки хмарних сервісів із використанням сучасних методів штучного інтелекту та аналізу даних.

У висновках дисертаційної роботи узагальнено результати проведеного дослідження та підсумовано значення отриманих наукових і практичних результатів. Показано, що запропонована модель та розроблені методи інтелектуальної системи виявлення аномалій у хмарних сервісах на основі використання методів машинного та глибокого навчання забезпечують підвищення точності виявлення аномальної активності, покращення ефективності моніторингу безпеки та адаптивність системи до змін у хмарному середовищі. Інтеграція API-журналів, телеметрії Kubernetes-кластерів, поведінкових характеристик користувачів і алгоритмів аналізу часових послідовностей дозволяє автоматизувати процес виявлення кіберзагроз та забезпечує більш ефективний аналіз подій у розподіленій

інфраструктурі. Впровадження розробленої інформаційної технології сприяє підвищенню рівня безпеки, надійності та практичної придатності систем моніторингу хмарних сервісів в умовах динамічних кіберзагроз.

Методи дослідження. При розв'язанні науково-прикладної задачі розробки моделі та методів інтелектуальної системи виявлення аномалій у хмарних сервісах використано комплекс теоретичних, математичних, алгоритмічних та імітаційних методів моделювання. Теоретичні методи застосовано для аналізу сучасного стану проблеми забезпечення безпеки хмарних сервісів, виявлення аномалій у контейнеризованих середовищах та аналізу кіберзагроз у розподілених системах. Математичне моделювання використано для формалізації процесів обробки API-журналів, аналізу поведінкових характеристик користувачів і сервісів, а також опису критеріїв виявлення аномальної активності у хмарній інфраструктурі.

При розробці методів виявлення аномалій застосовано нейромережеве та імітаційне моделювання, а також методи машинного й глибокого навчання для формування адаптивних моделей аналізу подій у реальному часі. Методи оптимізації використано для налаштування гіперпараметрів моделей та мінімізації похибки класифікації. Для аналізу часових послідовностей подій застосовано рекурентні нейронні мережі та модель LSTM Autoencoder, що дозволяє виявляти приховані аномальні патерни у поведінці контейнеризованих сервісів і Kubernetes-кластерів. З метою зменшення розмірності ознак і підвищення ефективності обробки даних використано метод головних компонент.

Для формалізації критеріїв оцінювання ефективності системи застосовано методи теорії ймовірностей, математичної статистики та аналізу якості класифікації, зокрема розрахунок метрик Accuracy, Precision, Recall і F1-score. Експериментальні дослідження виконано із застосуванням алгоритмічного та імітаційного моделювання в контейнеризованому Kubernetes-середовищі, а оцінювання результатів здійснювалося шляхом порівняння роботи моделей на нормальних та аномальних сценаріях

функціонування системи. Достовірність отриманих результатів забезпечується узгодженістю теоретичних положень і результатів експериментального моделювання, а також збіжністю результатів, отриманих у процесі програмної реалізації моделей засобами Python із використанням сучасних бібліотек машинного та глибокого навчання. Проведені експериментальні дослідження в Kubernetes-кластерах і на реальних API-журналах підтвердили ефективність запропонованих моделі і методів виявлення аномалій.

Основні завдання дослідження:

- проаналізувати сучасний стан проблеми виявлення аномалій у хмарних сервісах та існуючі підходи до забезпечення безпеки контейнеризованих і розподілених середовищ з урахуванням обмежень традиційних методів моніторингу та аналізу кіберзагроз;
- удосконалити метод виявлення аномалій у API-журналах і поведінці користувачів шляхом інтеграції поведінкових, мережевих та часових характеристик із використанням алгоритмів машинного навчання;
- удосконалити метод виявлення аномалій у Kubernetes-кластерах на основі аналізу телеметрії контейнеризованого середовища та застосування моделей глибокого навчання, зокрема LSTM Autoencoder;
- розробити математичну модель інтелектуальної системи виявлення аномалій, що забезпечує інтеграцію API-журналів, мережевих подій, телеметрії Kubernetes-кластерів і поведінкових даних у межах єдиного аналітичного процесу;
- реалізувати програмну архітектуру інтелектуальної системи та здійснити її експериментальну верифікацію у хмарному контейнеризованому середовищі з використанням сучасних метрик оцінювання ефективності виявлення аномалій.

Наукова новизна отриманих результатів обумовлена розробкою нових підходів до виявлення аномалій у хмарних сервісах на основі використання методів машинного та глибокого навчання для аналізу API-журналів,

поведінкових характеристик користувачів і телеметрії контейнеризованого середовища. Завдяки чому отримано наступні наукові результати:

1. Удосконалено метод виявлення аномальної поведінки користувачів вебресурсів шляхом формування індивідуального поведінкового профілю на основі інтеграції часових характеристик сесій, показників активності та послідовностей переходів між вебресурсами. На відміну від існуючих підходів, метод забезпечує комплексне врахування часових, кількісних і послідовнісних ознак під час побудови профілю нормальної поведінки та оцінювання аномальності, що дозволило підвищити повноту виявлення аномалій на 11,0% та F1-міру на 11,4% порівняно з методом One-Class SVM.

2. Удосконалено метод виявлення аномалій у Kubernetes-кластерах шляхом інтеграції мережових характеристик, метрик контейнерів та показників стану кластера в єдині часові послідовності ознак для формування профілю нормального функціонування контейнеризованого середовища. На відміну від існуючих підходів, метод забезпечує комплексне врахування взаємозв'язків між різнорідними джерелами телеметричних даних та їх часової динаміки під час оцінювання аномальності, що дозволило підвищити якість виявлення аномальної активності та забезпечити виявлення раніше невідомих атак у динамічних середовищах Kubernetes.

3. Отримала подальший розвиток модель мультиагентної інтелектуальної системи виявлення аномалій у хмарних середовищах, яка відрізняється інтеграцією агентів збору мережових подій, телеметрії Kubernetes, API-журналів та поведінкових даних користувачів у межах єдиного процесу моніторингу. На відміну від централізованого сигнатурного моніторингу на базі Snort, запропонована модель забезпечує багаторівневий аналіз подій безпеки та дозволяє підвищити повноту виявлення кіберзагроз на 18%, зменшити кількість хибнопозитивних спрацювань на 23% і скоротити час виявлення інцидентів на 29%.

Проведено аналіз предметної області за темою дослідження, розглянуто архітектуру хмарних сервісів, особливості контейнеризованих середовищ та

сучасні підходи до виявлення аномалій і забезпечення безпеки розподілених систем. Здійснено огляд існуючих інтелектуальних рішень, що застосовуються у сфері моніторингу хмарної інфраструктури та виявлення кіберзагроз, зокрема моделей машинного та глибокого навчання для аналізу API-журналів, поведінки користувачів і часових послідовностей подій. Сформульовано мету та завдання дисертаційної роботи, спрямовані на розробку моделі та методів інтелектуальної системи виявлення аномалій у хмарних сервісах.

Запропоновано метод виявлення аномалій у API-журналах і поведінці користувачів, який передбачає інтеграцію поведінкових, часових і мережевих характеристик із використанням алгоритмів машинного навчання. Метод забезпечує автоматизоване виявлення відхилень у роботі сервісів і поведінці користувачів з урахуванням багатофакторного впливу подій у хмарному середовищі та дозволяє підвищити точність виявлення кіберзагроз.

Розроблено метод виявлення аномалій у Kubernetes-кластерах, що базується на аналізі телеметрії контейнеризованого середовища, журналів подій та мережевої активності. Використання моделі LSTM Autoencoder дозволяє виявляти приховані аномальні патерни у часових послідовностях подій і підвищити ефективність моніторингу хмарної інфраструктури.

Запропоновано структурну та математичну модель інтелектуальної системи виявлення аномалій, яка формалізує повний цикл обробки даних від збору та підготовки API-журналів і телеметрії Kubernetes-кластерів до аналізу поведінкових характеристик та автоматизованого виявлення аномальної активності. Модель забезпечує інтеграцію аналітичних модулів, механізмів моніторингу та алгоритмів машинного навчання в межах єдиної функціональної архітектури.

Розроблено механізм адаптивного оновлення моделей, що дозволяє системі пристосовуватися до змін у поведінці сервісів і користувачів без повного перенавчання моделей, забезпечуючи актуальність процесу виявлення загроз у динамічному хмарному середовищі. Запропоновано

механізм багаторівневого моніторингу подій і поведінкових характеристик, що сприяє підвищенню стійкості системи до нових типів кіберзагроз.

Результати експериментальних досліджень підтвердили ефективність запропонованих методів та моделі інтелектуальної системи, продемонструвавши підвищення точності виявлення аномалій і покращення показників ефективності моніторингу безпеки у порівнянні з базовими підходами аналізу хмарної інфраструктури.

Отримані результати впроваджені на підприємстві ТОВ «РЕІНВЕНТІНГ СОФТВЕР ДЕВЕЛОПМЕНТ».

Матеріали дисертації викладено у 10-и публікаціях, з них – 1 розділ монографії, що індексується в Scopus та 4 статті у наукових фахових виданнях України категорії Б; 5 тез доповідей у матеріалах міжнародних науково-технічних конференцій.

Ключові слова: хмарна мікросервісна платформа, моніторинг, мікросервісна архітектура, машинне навчання, децентралізована архітектура, виявлення аномалій, класифікація, точність класифікації, виявлення ознак, масштабованість, Kubernetes, безпека програмного забезпечення, реагування на інциденти, інформаційна безпека, обробка сигналів.

Список публікацій здобувача

1. Мартовицький В. В., Свиридов А. С., Авдєєв О. В., Гудзинський І. М., Коротецький О. В. Дослідження методів виявлення аномалій у API журналах для забезпечення безпеки та надійності програмних систем // Вісник Херсонського національного технічного університету. 2025. Т. 2, № 1(92). С. 142–148. DOI: <https://doi.org/10.35546/kntu2078-4481.2025.1.2.19>.
2. Свиридов А. С., Гусейнов Р. Б., Винниченко С. М. Метод виявлення аномалій у поведінці користувача вебсайту // Herald of Khmelnytskyi National University. Technical Sciences. 2026. Т. 363, № 2. С. 211–219. DOI: <https://doi.org/10.31891/2307-5732-2026-363-28>.
3. Свиридов А. С., Северінов О. В. Метод виявлення аномалій у

Kubernetes-кластерах з використанням LSTM Autoencoder // Herald of Khmelnytskyi National University. Technical Sciences. 2026. Т. 361, № 1. С. 501–509. DOI: <https://doi.org/10.31891/2307-5732-2026-361-70>.

4. Свиридов А. С., Гудзинський І. М. Архітектура мультиагентної системи виявлення вторгнень із використанням машинного навчання // Вісник Херсонського національного технічного університету. 2026. Т. 1, № 2(97). С. 360–366. DOI: <https://doi.org/10.35546/kntu2078-4481.2026.2.44>.

5. Kuchuk G., Kovalenko A., Elyasi Komari I., Svyrydov A., Kharchenko V. Improving Big Data Centers Energy Efficiency: Traffic Based Model and Method // Advances in Computer Science for Engineering and Education II. Cham : Springer, 2019. P. 77–88. DOI: 10.1007/978-3-030-00253-4_8.

6. Сєверінов О. В., Свиридов А. С. Виявлення аномалій у API журналах для підвищення безпеки програмних систем // Сучасні напрями розвитку інформаційно-комунікаційних технологій та засобів управління : тези доповідей тринадцятої міжнародної науково-технічної конференції, 27–28 листопада 2025 р. Харків, 2025. Т. 3, секц. 4. С. 35.

7. Свиридов А. С. Метод виявлення аномалій у поведінці користувачів вебсайтів // Science and education: synergy of innovation : Proceedings of the 8th International scientific and practical conference, Berlin, Germany, 23–25 March 2026. Berlin : MDPC Publishing, 2026. P. 198. URL: <https://sci-conf.com.ua/viii-mizhnarodna-naukovo-praktichna-konferentsiya-science-and-education-synergy-of-innovation-23-25-03-2026-berlin-nimechchina-arhiv/>.

8. Свиридов А. С. Архітектура мультиагентної системи виявлення вторгнень із використанням машинного навчання // Scientific development in a changing world : Proceedings of the 3rd International scientific and practical conference, Lviv, Ukraine, 16–18 March 2026. Львів : SPC “Sci-conf.com.ua”, 2026. P. 330. URL: <https://sci-conf.com.ua/iii-mizhnarodna-naukovo-praktichna-konferentsiya-scientific-development-in-a-changing-world-16-18-03-2026-lviv-ukrayina-arhiv/>.

9. Свиридов А. С. Метод виявлення аномалій у Kubernetes-кластерах з

використанням LSTM Autoencoder // European science and innovation congress : Proceedings of the 4th International scientific and practical conference, Barcelona, Spain, 9–11 March 2026. Barcelona : Barca Academy Publishing, 2026. P. 169. URL: <https://sci-conf.com.ua/iv-mizhnarodna-naukovo-praktichna-konferentsiya-european-science-and-innovation-congress-9-11-03-2026-barselona-ispaniya-arhiv/>.

10. Свиридов А. С., Северінов О. В. Дослідження методів виявлення аномалій у API журналах для забезпечення безпеки та надійності програмних систем // Сучасні проблеми інфокомунікацій, радіоелектроніки та наносистем : тези доповідей шістнадцятої міжнародної науково-технічної конференції, 29–30 квітня 2026 р. Харків, 2026. Т. 3, секц. 4. С. 112–113.

ABSTRACT

Svyrydov A.S. Model and Methods of an Intelligent Anomaly Detection System for Cloud Services – Qualifying Scientific Work as a Manuscript. Dissertation for the degree of Doctor of Philosophy in Specialty 125 – Cybersecurity – Kharkiv National University of Radio Electronics, Ministry of Education and Science of Ukraine, Kharkiv, 2026.

The dissertation is devoted to the development of a model and methods for an intelligent anomaly detection system in cloud services based on artificial intelligence and machine learning techniques. The research proposes a comprehensive approach that integrates mechanisms for collecting and processing API logs, Kubernetes cluster telemetry, user behavioral scenarios, and microservice architecture components, as well as machine learning and deep learning algorithms, particularly LSTM Autoencoder, for detecting anomalous activity. The proposed model enables the formation of a structured feature dataset, automated training of intelligent models, multi-level monitoring of events in containerized environments, detection of cyber threats and behavioral anomalies in real time, and evaluation of system performance using standard classification quality metrics, thereby increasing the security, reliability, and adaptability of cloud software systems

under dynamic cyber threats.

The relevance of the research is determined by the rapid development of cloud technologies, widespread adoption of microservice architecture, containerization, and orchestration of applications, which are accompanied by increasing software complexity and growing cybersecurity threats. Modern cloud environments generate large volumes of heterogeneous data, including API logs, network events, container telemetry, and user behavioral data, the analysis of which by traditional methods often fails to ensure timely detection of anomalies, intrusions, and hidden threats. Existing security monitoring approaches usually have limited adaptability to dynamic environmental changes, are difficult to scale, and insufficiently account for the behavioral characteristics of modern distributed systems. The application of artificial intelligence and machine learning methods makes it possible to automate the analysis of large-scale data, identify hidden patterns and anomalous behaviors, and improve the security, reliability, and resilience of cloud software applications.

The purpose of the dissertation is to improve the efficiency of anomaly detection in cloud services through the development of models and methods of an intelligent information technology that ensure the integration of heterogeneous data, automated event analysis, and adaptive cyber threat detection in containerized and distributed environments.

The object of research is the processes of collecting, processing, and analyzing event logs, API traffic, user behavioral characteristics, and cloud infrastructure telemetry in anomaly detection systems.

The subject of research is the models and methods of an intelligent information technology for anomaly detection in cloud services based on machine learning and deep learning algorithms aimed at improving threat detection accuracy, security system adaptability, and risk reduction in software systems under dynamic cyber threats.

The first chapter provides a systematic analysis of modern theoretical and applied approaches to ensuring the security of cloud services and anomaly detection in distributed software environments. Classical and contemporary methods of

monitoring, event log analysis, intrusion detection systems, and behavioral analysis are examined, including approaches based on statistical methods, signature analysis, and machine learning algorithms. The limitations of traditional anomaly detection methods under conditions of highly dynamic cloud infrastructures, scalable microservice architectures, containerization, and continuously changing behavioral patterns of users and services are identified. Special attention is paid to the analysis of threats related to API interactions, Kubernetes clusters, network activity, and behavioral deviations, as well as to the challenges of processing large volumes of heterogeneous data in real time. Based on the conducted analysis, the necessity of applying artificial intelligence, machine learning, and deep learning methods to improve anomaly detection efficiency in cloud services is substantiated.

The second chapter develops a method for anomaly detection in cloud services based on machine learning and deep learning techniques. Modern approaches to anomaly detection in API logs, user behavior, and Kubernetes clusters using neural networks and time-series analysis algorithms are analyzed, formalizing the threat detection process as the analysis of event sequences and behavioral patterns in a cloud environment. Feature extraction mechanisms are considered with respect to API request characteristics, network activity, user behavioral parameters, and telemetry of containerized services, as well as the specific features of distributed microservice architectures. An improved anomaly detection method is proposed, integrating event log preprocessing, behavioral scenario analysis, time-sequence formation, and automated anomaly detection using LSTM Autoencoder. The method is based on the integration of temporal, behavioral, and network characteristics into the model training process, improved anomaly detection accuracy, and the capability to identify previously unknown cyber threats in real time.

The third chapter presents a method for anomaly detection in Kubernetes clusters based on deep learning models. The research is based on the analysis of event logs and telemetry data from containerized environments, which enabled the formation of a systematic approach to detecting anomalous activity in cloud infrastructures. A structural scheme of the method is developed, integrating data

collection from Kubernetes clusters, event log preprocessing, feature vector generation, and temporal dependency analysis in system behavior. The structure and characteristics of the datasets are described, and the selection of environment parameters for experimental studies is substantiated. Considerable attention is paid to data preparation procedures, including event log cleaning, parameter normalization, time-sequence transformation, and feature set generation for model training. The modeling stage involves the use of LSTM Autoencoder for automated anomaly detection in the behavior of containerized services and Kubernetes cluster network activity. The model is trained on normal operational scenarios, enabling the detection of deviations from typical behavior based on reconstruction error values. Evaluation metrics include Accuracy, Precision, Recall, and F1-score. The obtained results confirm the effectiveness of LSTM Autoencoder for anomaly detection in Kubernetes environments, as the proposed approach improves threat detection accuracy and system adaptability to dynamic changes in cloud infrastructure.

The fourth chapter proposes a comprehensive model of an intelligent anomaly detection system for cloud services and presents its software implementation. The functional model formalizes the entire information processing cycle, from collecting API logs, network events, and Kubernetes cluster telemetry to feature generation, intelligent model training, and automated anomaly detection. The model defines the interactions between data collection and aggregation modules, event log preprocessing, behavioral analysis, threat detection, and monitoring components, ensuring the integrity and consistency of the cloud security analysis process. The implemented system integrates data acquisition and preparation modules, machine learning and deep learning algorithms, time-sequence analysis mechanisms, and visualization tools for monitoring results. The software architecture provides automated anomaly detection, analysis of user and service behavior, monitoring of Kubernetes cluster states, and generation of real-time cyber threat notifications. Experimental validation confirmed the operability of the system and its adaptability to changes in cloud infrastructure and service behavior. Thus, the developed model and its software implementation provide a practical foundation for building adaptive

intelligent monitoring systems for cloud services security using modern artificial intelligence and data analysis methods.

The conclusions summarize the scientific and practical results of the dissertation research. The proposed model and developed methods of the intelligent anomaly detection system for cloud services based on machine learning and deep learning techniques improve anomaly detection accuracy, enhance the efficiency of security monitoring, and increase the adaptability of the system to changes in cloud environments. The integration of API logs, Kubernetes cluster telemetry, user behavioral characteristics, and time-sequence analysis algorithms enables automated cyber threat detection and provides more efficient event analysis in distributed infrastructures. The implementation of the developed information technology contributes to improving the security, reliability, and practical applicability of cloud services monitoring systems under dynamic cyber threats.

Research Methods. The study employed theoretical, mathematical, algorithmic, and simulation modeling methods to develop models and methods for an intelligent anomaly detection system in cloud services. Mathematical modeling was used to formalize API log processing, behavioral analysis, and anomaly detection criteria in cloud infrastructures. Neural network, machine learning, and deep learning methods were applied to develop adaptive real-time event analysis models. Time-series analysis using recurrent neural networks and the LSTM Autoencoder model was employed to detect hidden anomalous patterns in Kubernetes clusters and containerized services. Principal Component Analysis was used for feature dimensionality reduction and data processing optimization. Experimental validation was conducted using probability theory, mathematical statistics, and classification quality analysis methods, including Accuracy, Precision, Recall, and F1-score metrics. The reliability of the obtained results is confirmed by the consistency between theoretical modeling and software implementation in Python using modern machine learning and deep learning libraries, as well as by experimental studies performed on Kubernetes clusters and real API logs.

Main research objectives:

- analyze the current state of the problem of anomaly detection in cloud services and existing approaches to ensuring the security of containerized and distributed environments, taking into account the limitations of traditional cyber threat monitoring and analysis methods;
- improve the method for anomaly detection in API logs and user behavior through the integration of behavioral, network, and temporal characteristics using machine learning algorithms;
- improve the method for anomaly detection in Kubernetes clusters based on the analysis of telemetry data from containerized environments and the application of deep learning models, in particular LSTM Autoencoder;
- develop a structural and mathematical model of an intelligent anomaly detection system that ensures the integration of API logs, network events, Kubernetes cluster telemetry, and behavioral data within a unified analytical process;
- implement the software architecture of the intelligent system and perform its experimental verification in a cloud containerized environment using modern metrics for evaluating anomaly detection performance.

Scientific novelty of the obtained results lies in the development of new approaches to anomaly detection in cloud services based on the use of machine learning and deep learning methods for the analysis of API logs, user behavioral characteristics, and telemetry of containerized environments. As a result

1. A method for detecting anomalous behavior among users of web resources has been improved by creating an individual behavioral profile based on the integration of session duration, activity metrics, and sequences of transitions between web resources. Unlike existing approaches, this method comprehensively accounts for temporal, quantitative, and sequential features when constructing a profile of normal behavior and assessing anomalies, which increased the recall of anomaly detection by 11.0% and the F1 score by 11.4% compared to the One-Class SVM method.

2. We have improved the method for detecting anomalies in Kubernetes clusters by integrating network characteristics, container metrics, and cluster health indicators into a unified time series of features to form a profile of normal operation in a containerized environment. Unlike existing approaches, the method comprehensively accounts for the interrelationships between heterogeneous sources of telemetry data and their temporal dynamics when assessing anomalies, which has improved the quality of anomaly detection and enabled the detection of previously unknown attacks in dynamic Kubernetes environments.

3. The multi-agent intelligent anomaly detection system model for cloud environments has been further developed, distinguished by the integration of agents responsible for collecting network events, Kubernetes telemetry, API logs, and user behavioral data within a unified monitoring process. In contrast to centralized signature-based monitoring using Snort, the proposed model provides multi-level security event analysis and enables an 18% increase in cyber threat detection recall, a 23% reduction in false positive detections, and a 29% decrease in incident detection time.

The obtained results have been implemented at REINVENTING SOFTWARE DEVELOPMENT LLC.

The dissertation materials are presented in 10 publications, including 1 Scopus-indexed monograph chapter, 4 papers in Ukrainian Category B professional journals, and 5 conference proceedings.

Keywords: cloud microservice platform, monitoring, microservice architecture, machine learning, decentralized architecture, anomaly detection, classification, classification accuracy, feature detection, scalability, Kubernetes, software security, incident response, information security, signal processing.

ЗМІСТ

Перелік умовних позначень, символів, одиниць, скорочень і термінів	22
Вступ.....	25
1 Огляд архітектури хмарних сервісів та засобів їх захисту	34
1.1 Аналіз архітектури хмарних сервісів.....	34
1.2 Kubernetes	39
1.3 Системи захисту хмарних сервісів та методи штучного інтелекту в системах виявлення аномалій.....	47
Висновки до розділу 1	61
2 Виявлення аномалій в інфраструктурі kubernetes	63
2.1 Метод виявлення аномалій в інфраструктурі Kubernetes	63
2.2 Опис тестового середовища для проведення експерименту	65
2.3 Результати експериментального дослідження методу виявлення аномалій в інфраструктурі Kubernetes.....	73
Висновки до розділу 2.....	84
3 Виявлення аномалій в арі хмарних сервісів.....	86
3.1 Метод виявлення аномалій поведінки користувачів на основі API хмарних сервісів	86
3.2 Опис даних для проведення експерименту	88
3.3 Результати експериментального дослідження метод виявлення аномалій в API хмарних сервісів.....	101
Висновки до розділу 3	109
4 Інтелектуальна система виявлення аномалій в хмарних сервісах	111
4.1 Модель інтелектуальної системи виявлення аномалій в хмарних сервісах	111
4.2 Вибір засобів для розробки прототипу.....	121
4.3 Програмна реалізація інтелектуальної системи виявлення аномалій	130

Висновки до розділу 4.....	148
Висновки.....	150
Перелік джерел посилання	152
Додаток А Код програмної реалізації методів представлених у роботі	166
А.1 Код програмної реалізації методу виявлення аномалій в інфраструктурі Kubernetes	167
А.2 Код програмної реалізації методу виявлення аномалій в API хмарних сервісів.....	169
Додаток Б Список публікацій здобувача.....	174
Додаток В Акти впровадження	176

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

- AI – штучний інтелект (англ. Artificial Intelligence)
- ML – машинне навчання (англ. Machine Learning)
- DL – глибоке навчання (англ. Deep Learning)
- RNN – рекурентна нейронна мережа (англ. Recurrent Neural Network)
- LSTM – мережа довгої короткочасної пам'яті (англ. Long Short-Term Memory)
- AE – автоенкодер (англ. Autoencoder)
- LSTM-AE – LSTM-автоенкодер (англ. LSTM Autoencoder)
- PCA – метод головних компонент (англ. Principal Component Analysis)
- IDS – система виявлення вторгнень (англ. Intrusion Detection System)
- IT – інформаційна технологія (англ. Information Technology)
- API – програмний інтерфейс застосунку (англ. Application Programming Interface)
- GUI – графічний інтерфейс користувача (англ. Graphical User Interface)
- IaaS – інфраструктура як послуга (англ. Infrastructure as a Service)
- PaaS – платформа як послуга (англ. Platform as a Service)
- SaaS – програмне забезпечення як послуга (англ. Software as a Service)
- FaaS – функція як послуга (англ. Function as a Service)
- VM – віртуальна машина (англ. Virtual Machine)
- VPN – віртуальна приватна мережа (англ. Virtual Private Network)
- CPU – центральний процесор (англ. Central Processing Unit)
- GPU – графічний процесор (англ. Graphics Processing Unit)
- K8s – Kubernetes, платформа оркестрації контейнерів
- Docker – платформа контейнеризації застосунків
- Pod – мінімальна одиниця розгортання в Kubernetes
- Node – вузол кластера
- Cluster – кластер

TP – істинно позитивний результат (англ. True Positive)

TN – істинно негативний результат (англ. True Negative)

FP – хибно позитивний результат (англ. False Positive)

FN – хибно негативний результат (англ. False Negative)

Accuracy – точність класифікації

Precision – точність позитивного прогнозу

Recall – повнота

F1-score – гармонійне середнє Precision та Recall

MSE – середньоквадратична похибка (англ. Mean Squared Error)

MAE – середня абсолютна похибка (англ. Mean Absolute Error)

RMSE – корінь середньоквадратичної похибки (англ. Root Mean Squared Error)

Log – журнал подій

API Log – журнал API-запитів

Dataset – набір даних

Training Set – навчальна вибірка

Test Set – тестова вибірка

Feature Vector – вектор ознак

MB – мегабайт

GB – гігабайт

PPS – пакетів за секунду (англ. Packets Per Second)

Mbps – мегабіт за секунду (англ. Megabits per second)

Anomaly Detection – виявлення аномалій

Threat Detection – виявлення загроз

Behavioral Analysis – поведінковий аналіз

Microservice Architecture – мікросервісна архітектура

Containerized Environment – контейнеризоване середовище

Monitoring Dashboard – панель моніторингу

Runtime Environment – середовище виконання

Middleware – проміжне програмне забезпечення

Cloud Computing – хмарні обчислення

Frontend – клієнтська частина системи

Backend – серверна частина системи

ВСТУП

Актуальність теми дослідження.

Актуальність теми дослідження зумовлена стрімким розвитком хмарних технологій, широким впровадженням мікросервісної архітектури, контейнеризації та оркестрації застосунків, що супроводжується зростанням складності програмних систем і підвищенням рівня кіберзагроз. Хмарні середовища дедалі частіше зазнають впливу кібератак, спрямованих на компрометацію API-інтерфейсів, порушення роботи контейнеризованих сервісів, несанкціонований доступ до даних та експлуатацію вразливостей Kubernetes-кластерів. У таких умовах традиційні підходи до моніторингу безпеки, що ґрунтуються переважно на сигнатурному аналізі та статичних правилах виявлення загроз, не завжди здатні забезпечити належний рівень точності виявлення аномалій та ефективності реагування на сучасні кіберзагрози.

Суттєвим чинником, який визначає актуальність дослідження, є стрімке зростання обсягів даних, що генеруються у хмарному середовищі. Окрім традиційних мережових подій, сучасні програмні системи активно використовують API-журнали, телеметрію контейнерів, журнали Kubernetes-кластерів, поведінкові характеристики користувачів та інші дані моніторингу. Такі дані мають неструктурований характер, значний обсяг і високу швидкість оновлення, що ускладнює їх обробку традиційними методами аналізу. Водночас саме інтеграція різномірних джерел інформації дозволяє підвищити ефективність виявлення аномальної активності та оперативно реагувати на нові типи кіберзагроз у розподілених системах.

Розвиток цифрових технологій, хмарних платформ і DevOps-підходів сприяв формуванню нових архітектурних рішень, серед яких особливе місце посідають Kubernetes-кластери, контейнеризовані середовища та мікросервісна архітектура. Однак ефективність систем забезпечення безпеки

безпосередньо залежить від якості застосованих моделей, здатності аналізувати великі обсяги подій у режимі реального часу та адаптуватися до швидких змін у поведінці сервісів і користувачів. У цьому контексті виникає необхідність розробки нових інтелектуальних інформаційних технологій, що поєднують математичне моделювання, методи машинного та глибокого навчання, а також сучасні підходи до побудови програмних систем.

Особливу роль у підвищенні ефективності виявлення аномалій відіграють методи штучного інтелекту, зокрема рекурентні нейронні мережі, моделі аналізу часових послідовностей та глибоке навчання. Використання моделей LSTM Autoencoder дозволяє підвищити точність виявлення прихованих аномальних патернів у поведінці контейнеризованих сервісів, API-трафіку та Kubernetes-кластерів, а алгоритми поведінкового аналізу забезпечують оцінювання відхилень у роботі користувачів і програмних компонентів. Методи машинного та глибокого навчання дозволяють автоматизувати процес аналізу великих обсягів даних, виявляти приховані закономірності та забезпечувати адаптивність системи до динамічних змін хмарного середовища.

Важливою проблемою сучасних систем моніторингу безпеки є ефективне виявлення кіберзагроз у режимі реального часу. Традиційні методи аналізу журналів подій та мережевого трафіку хоча й залишаються актуальними, не завжди дозволяють врахувати комплексний вплив поведінкових факторів, змін навантаження сервісів та динаміки роботи контейнеризованого середовища. Відсутність механізмів адаптивного аналізу даних може призводити до запізненого виявлення загроз і зниження рівня безпеки хмарної інфраструктури. Тому розробка інтелектуальних методів виявлення аномалій, здатних інтегрувати різноманітні потоки даних і адаптуватися до нових умов функціонування системи, є актуальним науковим і практичним завданням.

Додаткової актуальності темі надає необхідність створення комплексної інтелектуальної системи, яка б об'єднувала всі етапи процесу моніторингу

безпеки: збір і агрегацію API-журналів, попередню обробку даних, формування ознак, навчання моделей, аналіз часових послідовностей та автоматизоване виявлення аномалій. Більшість існуючих рішень реалізують лише окремі компоненти цього процесу, що ускладнює забезпечення цілісності, масштабованості та адаптивності системи. Формування інтегрованої моделі інтелектуальної системи дозволяє забезпечити ефективний моніторинг безпеки хмарних сервісів та підвищити стійкість програмної інфраструктури до сучасних кіберзагроз.

Мета дослідження – підвищення ефективності виявлення аномалій у хмарних сервісах шляхом розробки моделі і методів інтелектуальної інформаційної системи, що забезпечують інтеграцію різнорідних даних, автоматизований аналіз подій та адаптивне виявлення кіберзагроз у контейнеризованих і розподілених середовищах.

Об’єкт дослідження – процеси збору, обробки та аналізу журналів подій, API-трафіку, поведінкових характеристик користувачів і телеметрії хмарної інфраструктури в системах виявлення аномалій.

Предмет дослідження – моделі та методи інтелектуальної інформаційної технології виявлення аномалій у хмарних сервісах на основі алгоритмів машинного та глибокого навчання, спрямовані на підвищення точності виявлення загроз, адаптивності систем безпеки та зниження ризиків функціонування програмних систем в умовах динамічних кіберзагроз.

Методи дослідження. При розв’язанні науково-прикладної задачі розробки моделі та методів інтелектуальної системи виявлення аномалій у хмарних сервісах використано комплекс теоретичних, математичних, алгоритмічних та імітаційних методів моделювання. Теоретичні методи застосовано для аналізу сучасного стану проблеми забезпечення безпеки хмарних сервісів, виявлення аномалій у контейнеризованих середовищах та аналізу кіберзагроз у розподілених системах. Математичне моделювання використано для формалізації процесів обробки API-журналів, аналізу поведінкових характеристик користувачів і сервісів, а також опису критеріїв

виявлення аномальної активності у хмарній інфраструктурі.

При розробці методів виявлення аномалій застосовано нейромережеве та імітаційне моделювання, а також методи машинного й глибокого навчання для формування адаптивних моделей аналізу подій у реальному часі. Методи оптимізації використано для налаштування гіперпараметрів моделей та мінімізації похибки класифікації. Для аналізу часових послідовностей подій застосовано рекурентні нейронні мережі та модель LSTM Autoencoder, що дозволяє виявляти приховані аномальні патерни у поведінці контейнеризованих сервісів і Kubernetes-кластерів. З метою зменшення розмірності ознак і підвищення ефективності обробки даних використано метод головних компонент.

Для формалізації критеріїв оцінювання ефективності системи застосовано методи теорії ймовірностей, математичної статистики та аналізу якості класифікації, зокрема розрахунок метрик Accuracy, Precision, Recall і F1-score. Експериментальні дослідження виконано із застосуванням алгоритмічного та імітаційного моделювання в контейнеризованому Kubernetes-середовищі, а оцінювання результатів здійснювалося шляхом порівняння роботи моделей на нормальних та аномальних сценаріях функціонування системи. Достовірність отриманих результатів забезпечується узгодженістю теоретичних положень і результатів експериментального моделювання, а також збіжністю результатів, отриманих у процесі програмної реалізації моделей засобами Python із використанням сучасних бібліотек машинного та глибокого навчання. Проведені експериментальні дослідження в Kubernetes-кластерах і на реальних API-журналах підтвердили ефективність запропонованих моделі і методів виявлення аномалій.

Наукова новизна отриманих результатів обумовлена розробкою нових підходів до виявлення аномалій у хмарних сервісах на основі використання методів машинного та глибокого навчання для аналізу API-журналів, поведінкових характеристик користувачів і телеметрії контейнеризованого

середовища. Завдяки чому отримано наступні наукові результати:

1. Удосконалено метод виявлення аномальної поведінки користувачів вебресурсів шляхом формування індивідуального поведінкового профілю на основі інтеграції часових характеристик сесій, показників активності та послідовностей переходів між вебресурсами. На відміну від існуючих підходів, метод забезпечує комплексне врахування часових, кількісних і послідовнісних ознак під час побудови профілю нормальної поведінки та оцінювання аномальності, що дозволило підвищити повноту виявлення аномалій на 11,0% та F1-міру на 11,4% порівняно з методом One-Class SVM.

2. Удосконалено метод виявлення аномалій у Kubernetes-кластерах шляхом інтеграції мережових характеристик, метрик контейнерів та показників стану кластера в єдині часові послідовності ознак для формування профілю нормального функціонування контейнеризованого середовища. На відміну від існуючих підходів, метод забезпечує комплексне врахування взаємозв'язків між різнорідними джерелами телеметричних даних та їх часової динаміки під час оцінювання аномальності, що дозволило підвищити якість виявлення аномальної активності та забезпечити виявлення раніше невідомих атак у динамічних середовищах Kubernetes.

3. Отримала подальший розвиток модель мультиагентної інтелектуальної системи виявлення аномалій у хмарних середовищах, яка відрізняється інтеграцією агентів збору мережових подій, телеметрії Kubernetes, API-журналів та поведінкових даних користувачів у межах єдиного процесу моніторингу. На відміну від централізованого сигнатурного моніторингу на базі Snort, запропонована модель забезпечує багаторівневий аналіз подій безпеки та дозволяє підвищити повноту виявлення кіберзагроз на 18%, зменшити кількість хибнопозитивних спрацювань на 23% і скоротити час виявлення інцидентів на 29%.

Практичне значення результатів роботи полягає в тому, що розроблені у дисертації модель інтелектуальної системи виявлення аномалій у хмарних сервісах, методи аналізу API-журналів, поведінкових характеристик

користувачів і телеметрії Kubernetes-кластерів, а також реалізовані механізми інтеграції різномірних джерел даних є науково-практичною основою для подальшого розвитку систем моніторингу безпеки хмарної інфраструктури та виявлення кіберзагроз у розподілених програмних середовищах. Запропоновані підходи забезпечують інтеграцію API-журналів, мережевих подій, телеметрії контейнеризованого середовища та поведінкових даних користувачів, підвищення точності виявлення аномальної активності за рахунок застосування моделей машинного й глибокого навчання, зокрема LSTM Autoencoder, а також адаптивний аналіз подій у режимі реального часу з урахуванням змін у поведінці сервісів і користувачів. Реалізація алгоритмів аналізу часових послідовностей та механізмів автоматизованого моніторингу дозволяє підвищити ефективність виявлення кіберзагроз і покращити показники якості класифікації у порівнянні з базовими підходами аналізу безпеки хмарних систем.

Результати дисертаційної роботи можуть бути використані при створенні та модернізації систем моніторингу безпеки хмарних сервісів, платформ виявлення аномалій, систем аналізу API-трафіку, а також програмних комплексів моніторингу контейнеризованих і Kubernetes-середовищ. Запропонована модель є придатною для впровадження в інтелектуальні програмні системи кібербезпеки, де критично важливими є швидкість обробки великих обсягів різномірних даних, адаптивність до динамічних змін хмарного середовища та своєчасне виявлення аномалій і кіберзагроз.

Отримані результати впроваджені на підприємстві ТОВ «РЕІНВЕНТІНГ СОФТВЕР ДЕВЕЛОПМЕНТ». Основні результати дослідження опубліковано у наукових виданнях, що підтверджує їх наукову та практичну значущість і апробацію в професійному середовищі.

У дисертаційній роботі отримано комплекс наукових і прикладних результатів, спрямованих на розробку моделі та методів інтелектуальної системи виявлення аномалій у хмарних сервісах на основі використання

методів машинного та глибокого навчання, які опубліковані в працях [1-10].

У роботах [1, 6, 10] досліджено методи виявлення аномалій у API-журналах для забезпечення безпеки та надійності програмних систем. У зазначених працях проаналізовано особливості обробки API-журналів, визначено підходи до аналізу поведінкових характеристик користувачів і сервісів, а також обґрунтовано доцільність застосування методів машинного навчання для автоматизованого виявлення аномальної активності у хмарному середовищі.

У роботах [2, 7] розроблено метод виявлення аномалій у поведінці користувачів вебсайтів. Запропоновано підхід до аналізу поведінкових сценаріїв користувачів із використанням алгоритмів машинного навчання, що дозволяє визначати відхилення від типових моделей поведінки та підвищувати ефективність виявлення потенційних кіберзагроз.

Метод виявлення аномалій у Kubernetes-кластерах із використанням моделі LSTM Autoencoder розроблено та апробовано у роботах [3, 9]. У зазначених публікаціях обґрунтовано доцільність використання рекурентних нейронних мереж для аналізу часових послідовностей подій і телеметрії контейнеризованих середовищ. Окремі підходи до аналізу мережевого трафіку та обробки великих обсягів даних у розподілених інфраструктурах, представлені у роботі [5], використано як додаткову методологічну основу дослідження. Запропонований підхід дозволяє виявляти приховані аномальні патерни у поведінці Kubernetes-кластерів та контейнеризованих сервісів.

Архітектуру мультиагентної системи виявлення аномалій із використанням методів машинного навчання представлено у роботах [4, 8]. У цих працях сформовано функціональну та структурну модель системи, визначено взаємозв'язки між модулями збору даних, аналізу подій, виявлення аномалій та моніторингу безпеки у розподілених програмних середовищах.

Таким чином, сукупність опублікованих результатів [1-10] відображає поетапний розвиток дослідження від аналізу API-журналів і поведінкових характеристик користувачів до розробки методів виявлення аномалій у

Kubernetes-кластерах та побудови комплексної інтелектуальної системи моніторингу безпеки хмарних сервісів на основі методів машинного та глибокого навчання.

Особистий внесок здобувача. Всі основні результати дисертаційної роботи, що виносяться на захист, отримані автором особисто. У роботах [1], [6], [10] здобувачем досліджено методи виявлення аномалій у API-журналах; виконано аналіз особливостей обробки API-трафіку, проведено експериментальне моделювання процесів аналізу журналів подій та обґрунтовано можливість використання методів машинного навчання для автоматизованого виявлення аномальної активності у програмних системах. У роботах [2], [7] здобувачем розроблено метод виявлення аномалій у поведінці користувачів вебсайтів; сформовано підхід до аналізу поведінкових сценаріїв, реалізовано алгоритми обробки поведінкових характеристик користувачів та проведено експериментальне оцінювання ефективності запропонованого підходу. У роботах [3], [9] здобувачем запропоновано метод виявлення аномалій у Kubernetes-кластерах із використанням моделі LSTM Autoencoder; здійснено формування наборів ознак на основі телеметрії контейнеризованих середовищ, реалізовано моделі аналізу часових послідовностей подій та виконано оцінювання точності виявлення аномальної активності у Kubernetes-середовищах. Окремі підходи до аналізу мережевого трафіку та обробки великих обсягів даних у розподілених інфраструктурах, представлені у роботі [5], використано як додаткову методологічну основу дослідження. У роботах [4], [8] здобувачем розроблено архітектуру мультиагентної системи виявлення аномалій із використанням методів машинного навчання; сформовано функціональну та структурну модель системи, визначено взаємозв'язки між модулями збору даних, аналізу подій, виявлення аномалій і моніторингу безпеки у розподілених програмних середовищах. Таким чином, здобувачем особисто виконано розробку моделі і методів виявлення аномалій, їх математичну формалізацію, програмну реалізацію, проведення експериментальних досліджень, аналіз результатів і формування висновків

щодо ефективності запропонованої інтелектуальної системи виявлення аномалій у хмарних сервісах.

Апробація результатів дисертації.

Основні положення дисертаційної роботи представлено на наступних міжнародних науково-технічних конференціях і форумах:

- на Тринадцятій міжнародній науково-технічній конференції «Сучасні напрями розвитку інформаційно-комунікаційних технологій та засобів управління», 27–28 листопада 2025 р.;
- на Міжнародній науково-практичній конференції «Science and education: synergy of innovation», Berlin, Germany, 2026 р.;
- на Міжнародній науково-практичній конференції «Scientific development in a changing world», Lviv, Ukraine, 2026 р.;
- на Міжнародній науково-технічній конференції «European science and innovation congress», Barcelona, Spain, 2026 р.;
- на Шістнадцятій міжнародній науково-технічній конференції «Сучасні напрями розвитку інформаційно-комунікаційних технологій та засобів управління», 29 – 30 квітня 2026 р.

Публікації. Матеріали дисертації викладено у 10-и публікаціях, з них – 1 розділ монографії, що індексується в Scopus та 4 статті у наукових фахових виданнях України категорії Б; 5 тез доповідей у матеріалах міжнародних науково-технічних конференцій.

Структура та обсяг роботи. Дисертація складається зі вступу, 4 розділів, висновків, списку використаних джерел, додатків. Загальний обсяг роботи складає 177 сторінок тексту, що містять 131 сторінок основного тексту, анотації двома мовами на 19 сторінках, 40 рисунків, 13 таблиц, список використаних джерел з 102 найменувань на 15 сторінках, 3 додатки на 12 сторінках.

1 ОГЛЯД АРХІТЕКТУРИ ХМАРНИХ СЕРВІСІВ ТА ЗАСОБІВ ЇХ ЗАХИСТУ

1.1 Аналіз архітектури хмарних сервісів

З появою хмарних обчислень спосіб функціонування ІТ-сектору докорінно змінився. Завдяки хмарним обчисленням можна отримувати кращі ІТ-послуги з меншими інвестиціями та за нижчою ціною. Сучасна парадигма хмарних технологій передбачає розгортання чотирьох основних типів хмар: приватної, публічної, спільної та гібридної [11, 12]. У період інтенсивної експансії глобальної мережі Інтернет у середині 1990-х років спостерігалася рекурентна тенденція до централізації клієнт-серверних архітектур, що згодом стало технологічним підґрунтям для появи сучасних хмарних рішень [13]. Прагнення бізнесу до об'єднання обчислювальних ресурсів призвело до створення власних центрів обробки даних. Це дозволило оптимізувати адміністрування та забезпечити вищий рівень цілісності інформаційних потоків. Широке впровадження інтернет-технологій та віртуальних приватних мереж надало можливість персоналу використовувати методи віддаленого «тунелювання» для доступу до корпоративних додатків та сховищ даних [14].

Еволюційний перехід розпочався приблизно два десятиліття тому, коли підприємства почали інтегрувати орендовані хмарні потужності як альтернативу підтримці власних серверних інфраструктур. У той час як консервативні галузі промисловості з високим рівнем ризику виявляли стриманість щодо міграції в хмару, технологічні стартапи та ІТ-підприємства стали основою впровадження цієї парадигми, забезпечивши собі конкурентні переваги за рахунок гнучкості та масштабованості обчислювальних ресурсів.

Еволюція хмарних обчислень відображає поступовий розвиток інформаційних технологій, що включає перехід від розподілених систем і мейнфрейм-обчислень до сучасних концепцій, зокрема віртуалізації та

прикладних обчислень. Наприклад метою розподілених систем є надсилання повідомлення з кількох різних систем, які безпосередньо розподілені по різних локаціях, але пов'язані мережею. Мейнфрейми й сьогодні використовуються в окремих компаніях завдяки їхній здатності опрацьовувати та передавати колосальні обсяги даних [15]. Розвиток хмарних технологій став можливим завдяки поєднанню кількох ключових інновацій, що дозволили перейти від власних серверів до моделі використання ресурсів за запитом. Ключовим етапом цього розвитку стало впровадження кластерних обчислень, що передбачають архітектурне об'єднання функціонально еквівалентних вузлів у цілісну систему, яка функціонує як єдиний апаратний ресурс. Робота елементів кластера в режимі паралельної обробки завдань дозволила значно підвищити пропускну здатність та забезпечити прозорість операційних процесів завдяки синхронізації стану між усіма мережевими вузлами. Паралельно з кластерними рішеннями розвивалася концепція ґрід-обчислень, що орієнтована на створення географічно розподілених децентралізованих інфраструктур, де територіально віддалені гетерогенні ресурси інтегруються через глобальні мережеві протоколи для вирішення масштабних обчислювальних задач. Трансформація моделі взаємодії користувача з інформаційним середовищем у межах технології Веб 2.0 заклала основу для сервіс-орієнтованої архітектури, змістивши акцент зі споживання контенту на його активну генерацію та спільну роботу в реальному часі. Вирішальним чинником становлення хмарних послуг стало впровадження технологій віртуалізації, що реалізують рівень логічного абстрагування поверх фізичного апаратного забезпечення. Використання гіпервізорів дозволило динамічно масштабувати прикладні обчислення відповідно до мінливих потреб користувача, забезпечуючи ізоляцію середовищ та ефективну утилізацію серверних потужностей. Завершальним етапом формування цієї парадигми став перехід до моделі оренди сховищ даних та обчислювальних ресурсів, що дозволило суб'єктам бізнесу трансформувати капітальні витрати на ІТ-інфраструктуру в операційні, масштабуючи обсяги збереженої інформації та

потужність сервісів згідно з динамічними вимогами ринкової кон'юнктури [16].

Архітектура хмарних обчислень включає сукупність взаємодіючих компонентів, серед яких фронтенд, бекенд, середовище виконання, системи зберігання та обробки даних, а також мережеві та допоміжні сервіси. Користувацька або клієнтська сторона системи хмарних обчислень називається фронтендною хмарною архітектурою. Вона пропонує миттєвий доступ до хмарних ресурсів та зручностей через панелі інструментів, інструменти навігації та графічні інтерфейси користувача. Програми та програмні додатки, що встановлюються на пристроях (таких як настільні комп'ютери, ноутбуки та мобільні телефони) для доступу до хмарної платформи або сервісу, є важливими компонентами. Платформи або програмне забезпечення, що обробляють запити клієнтів на обслуговування на фронтенді, відомі як бекендові додатки. Архітектурна цілісність хмарних систем забезпечується функціонуванням серверних сервісів, які здійснюють централізований контроль доступу до віртуалізованих ресурсів та оркестрацію прикладних додатків. Фундаментальною основою цього процесу є середовище виконання, що консолідує необхідне апаратне забезпечення, системне програмне забезпечення та обчислювальну пам'ять для стабільної роботи хмарних служб. Застосування технологій віртуалізації відіграє критичну роль у забезпеченні мультиорендності, дозволяючи ізольовано виконувати декілька гетерогенних середовищ на єдиній фізичній платформі. Системне управління даними на рівні сервера створює передумови для побудови високонавантажених, масштабованих та адаптивних архітектур зберігання, які динамічно підлаштовуються під вимоги прикладного програмного забезпечення [17].

Технологічний стек інфраструктури охоплює комплексне апаратне та програмне забезпечення, включаючи високопродуктивні, реляційні та нереляційні бази даних, центральні та графічні процесори, а також активне мережеве обладнання, як комутатори та маршрутизатори, що формують

комунікаційний каркас системи. Важливим сполучним елементом у цій ієрархії є проміжне програмне забезпечення, яке оптимізує інтерфейсну взаємодію та забезпечує безперебійну передачу даних між клієнтською частиною та серверними потужностями у режимі реального часу. Безпека серверної компоненти гарантується впровадженням комплексних інструментів захисту, що базуються на криптографічних протоколах шифрування, багаторівневих системах контролю доступу та автентифікації, що мінімізує ризики порушення цілісності та конфіденційності даних. Комунікаційне з'єднання між користувачьким інтерфейсом та серверними операціями реалізується через мережеву інфраструктуру, яка забезпечує низьку латентність та високу доступність сервісів для кінцевого споживача.

Типи хмарних обчислень включають кілька основних моделей надання послуг. IaaS надає доступ до віртуалізованих обчислювальних ресурсів, таких як віртуальні машини, сховища та мережі через Інтернет, забезпечуючи гнучкість у керуванні операційними системами та можливість масштабування ресурсів відповідно до потреб користувача [18]. PaaS забезпечує середовище для розробки та розгортання додатків без необхідності управління інфраструктурою, що дозволяє розробникам зосередитися на реалізації логіки програм, тоді як масштабування та обслуговування автоматизуються платформою, підвищуючи продуктивність і скорочуючи час розробки [19]. SaaS передбачає використання програмного забезпечення через Інтернет без його локального встановлення, де постачальник забезпечує оновлення, обслуговування та безпеку, що сприяє доступності, спрощує співпрацю та знижує витрати на IT-інфраструктуру. FaaS є безсерверною моделлю, яка дозволяє виконувати код у відповідь на події без необхідності управління інфраструктурою, забезпечуючи масштабованість, гнучкість і економічну ефективність завдяки моделі оплати за фактичне використання ресурсів [20].

Моделі хмарних обчислень визначають спосіб розгортання та використання хмарної інфраструктури. Модель приватного розгортання забезпечує підвищений рівень безпеки та можливість гнучкого налаштування

ресурсів відповідно до потреб конкретної організації. Модель публічного розгортання надає доступ до хмарних ресурсів широкому колу користувачів за моделлю оплати за використання, що забезпечує масштабованість та економічну ефективність. Гібридна модель поєднує переваги публічної та приватної хмар, забезпечуючи інтеграцію середовищ, гнучке управління ресурсами та можливість зберігання критично важливих даних у захищеній частині інфраструктури.

Хмарні обчислення характеризуються низкою ключових властивостей, що забезпечують їх ефективність і широке застосування. Самообслуговування на вимогу дозволяє користувачам самостійно отримувати доступ до ресурсів без необхідності втручання постачальника. Широкий доступ до мережі забезпечує можливість використання хмарних сервісів з будь-якого місця за наявності Інтернет-з'єднання. Швидка еластичність дозволяє оперативно масштабувати ресурси залежно від навантаження, що є важливою перевагою хмарних платформ [21]. Об'єднання ресурсів передбачає спільне використання інфраструктури між кількома користувачами, що підвищує ефективність і знижує витрати. Масштабований сервіс і модель ціноутворення за використання забезпечують оплату лише за фактично спожиті ресурси, що робить хмарні рішення економічно вигідними. Багатокористувацька оренда дозволяє різним користувачам працювати в межах однієї інфраструктури із збереженням ізоляції даних. Віртуалізація забезпечує можливість запуску кількох операційних систем на одному фізичному сервері. Доступність і стійкість гарантують безперервність роботи та збереження даних навіть у разі збоїв. Важливою характеристикою також є безпека, яка досягається за рахунок використання механізмів шифрування, контролю доступу та багаторівневого захисту даних [22].

Сьогодні Kubernetes є головним стандартом для роботи в хмарі [23,24]. Як основна система для управління контейнерами, він забезпечує гнучкість, надійність та розумний розподіл ресурсів. Використання Kubernetes допомагає вирішити проблеми простоїв та складності при масштабуванні систем.

Перехід від великих монолітних програм до мікросервісів під управлінням K8s дозволяє повністю автоматизувати роботу додатків. Таким чином, сучасна IT-інфраструктура створює нове середовище хмарних рішень, які стають основою цифрових змін у світі.

1.2 Kubernetes

Kubernetes – це система оркестрації контейнерів з відкритим кодом, яка використовується для автоматичного розгортання, масштабування та керування контейнерами додатків. Розроблена Google, Kubernetes – це платформа, доступна лише через специфічний для домену інтерфейс програмування репрезентативного стану передачі додатків (REST API) та підтримує ширше коло клієнтів, застосовуючи керування версіями вищого рівня, валідацію, семантику та політики [25]. Система працює з багатьма інструментами контейнеризації, включаючи Docker.

Вона допомагає керувати контейнеризованими додатками в різних середовищах розгортання, таких як фізичні віртуальні або хмарні середовища. Вона пропонує багато функцій, таких як висока доступність, масштабованість та аварійне відновлення.

Базова архітектура Kubernetes – це кластер, показаний на рисунку 1.1, який складається з головного вузла та кількох робочих вузлів, де кожен вузол має запущений процес який дозволяє вузлам взаємодіяти один з одним у кластері. Головний вузол також називають площиною керування, і він містить певні компоненти, необхідні для керування всім кластером [26]. Ці компоненти – це сервер інтерфейсу прикладного програмування, планувальник, менеджер контролерів та розподілений /etcd. Ці компоненти працюють разом для керування всім кластером Kubernetes:

- API-сервер – є точкою входу до кластера та надає доступ до Kubernetes API, через який здійснюється взаємодія між зовнішнім середовищем і компонентами кластера;

- планувальник – відповідає за призначення нових Pod-ів відповідним вузлам кластера на основі доступних ресурсів і поточного навантаження;
- контролер-менеджер – компонент, що складається з набору контролерів (вузлів, завдань, кінцевих точок, облікових записів), кожен із яких виконує окрему функцію, зокрема контроль працездатності вузлів [27];
- etcd – розподілене сховище ключ-значення, яке зберігає конфігураційні дані та поточний стан кластера Kubernetes.

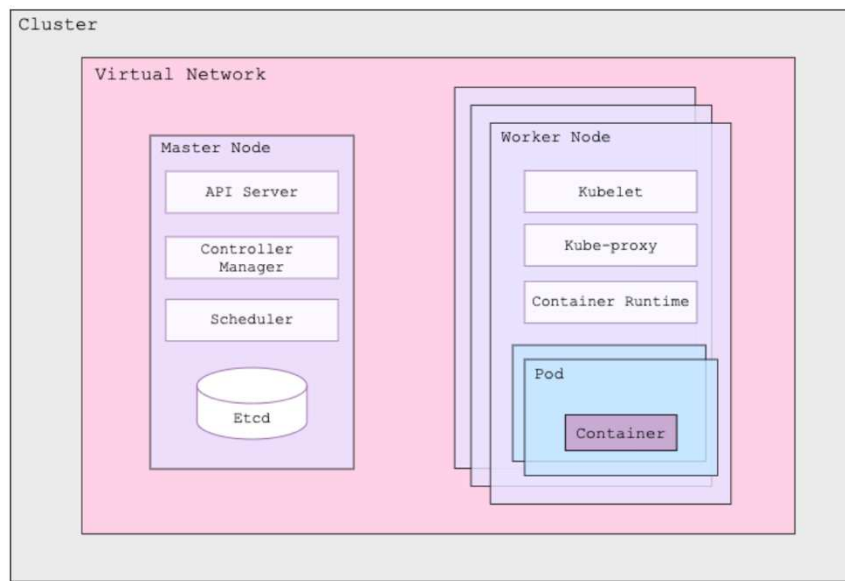


Рисунок 1.1 – Зовнішній вигляд кластера

Робочі вузли також називають площиною даних [28]. Кілька подів, що містять контейнеризовані програми, працюють усередині Робочих вузлів. У кластері може бути кілька Робочих вузлів, і всі вони контролюються Головним вузлом. Як і Головний вузол, Робочий вузол також має компоненти, які дозволяють йому правильно функціонувати:

- Kubelet – агент, що працює на вузлах кластера та забезпечує запуск і коректну роботу контейнерів у Pod-ах. Він взаємодіє з API-сервером, отримує команди та виконує їх, виступаючи зв'язком між головним і робочими вузлами;
- Kube-proxy – мережевий компонент, що забезпечує взаємодію між

Pod-ами та вузлами кластера, підтримує мережеві правила та реалізує концепцію сервісів Kubernetes, використовуючи механізми маршрутизації трафіку (наприклад, iptables) [29];

- Container Runtime – програмне забезпечення, що відповідає за запуск контейнерів на вузлі; Kubernetes підтримує різні середовища виконання, зокрема Docker, CRI-O та інші реалізації CRI;

- Pod – найменша одиниця розгортання в Kubernetes, яка об'єднує один або кілька контейнерів із спільними ресурсами (мережею, сховищем, IP-адресою). Pod-и є ефемерними, можуть перезапускатися або замінюватися новими екземплярами у разі збоїв [30, 31].

Найпростіше з'єднання в кластері Kubernetes відбувається між двома контейнерами в одному Pod. Вони взаємодіють через localhost та номери портів. Контейнери в Pod спільно використовують багато ресурсів, включаючи мережевий простір імен та віртуальне Ethernet-з'єднання, eth0. З точки зору контейнерів, Pod – це окрема машина з власним мережевим інтерфейсом, тобто він має власну IP-адресу та діапазон портів [32]. Pod може виділяти ці порти різним контейнерам, дозволяючи їм пов'язувати різні контейнери з їхніми номерами портів.

Наприклад, на рисунку 1.2 контейнер А працює на порту 8080, а контейнер В – на порту 9000. Контейнер А може взаємодіяти з контейнером В через localhost:9000 і навпаки через localhost:8080.

Коли контейнеризовані програми всередині подів потребують взаємодії між собою, шлях з'єднання, створений між ними, залежить від того, чи знаходяться вони в одному вузлі, чи в іншому. Кожен Pod має власну IP-адресу, мережевий простір імен та віртуальне Ethernet-з'єднання під назвою eth0. Цей eth0 підключений до віртуального Ethernet-пристрою у вузлі під назвою vethX.

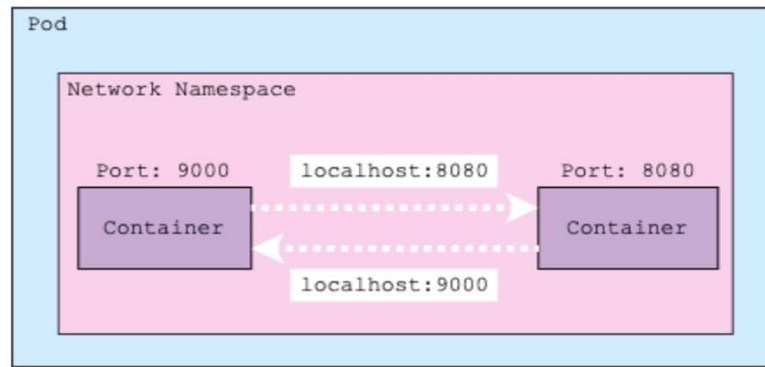


Рисунок 1.2 – Приклад зв'язка між контейнерами

Оскільки ці з'єднання встановлюються для кожного Pod у вузлі, X у `vethX` замінюється числом, яке представляє Pod, до якого він підключений, наприклад, `veth1` та `veth2`. Мережевий міст вузла, який називається `cbr0`, використовує ці з'єднання для доставки пакетів до та від Pod. Мережевий міст – це міст, який з'єднує два простори імен разом [33], у цьому випадку він з'єднує всі Pod у вузлі разом. На рисунку 1.3 показано, як два Pod-и взаємодіють один з одним в межах одного вузла. Pod 1 хоче надіслати пакет до Pod 2. Усі Pod-и в межах вузла знають IP-адреси один одного. Pod 1 надсилає пакет до свого `eth0` (1), який підключений до `veth1` (2) у просторі імен вузла.

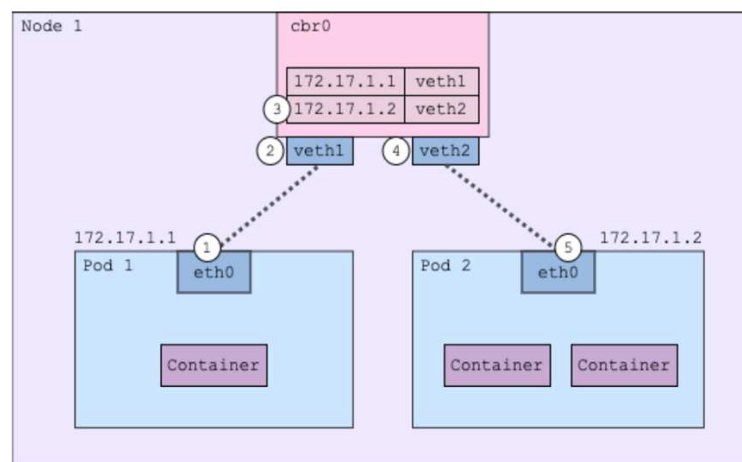


Рисунок 1.3 – Два Pod-и в одному вузлі, під назвою Pod 1 та Pod 2, взаємодіють один з одним за допомогою мережевого мосту під назвою `cbr0`.

У просторі імен вузла cbr0 отримує запит від Pod 1 з IP-адресою призначення, перевіряє всі Pod-и, підключені до нього, на наявність адреси призначення, і після її знаходження пакет надсилається до Pod (3). У цьому випадку пакет, який він надіслав до veth2 (4), оскільки це з'єднання, пов'язане з Pod 2. veth2 підключений до eth0 у просторі імен Pod 2, і це дозволяє пакету досягти місця призначення, Pod 2 (5).

Поди також можуть взаємодіяти з іншими Подами в різних вузлах, як показано на рисунку 1.4. Єдина відмінність полягає в тому, що коли cbr0 Вузла 1 не може знайти Под з IP-адресою призначення, він надсилає запит на рівень Кластера (1).

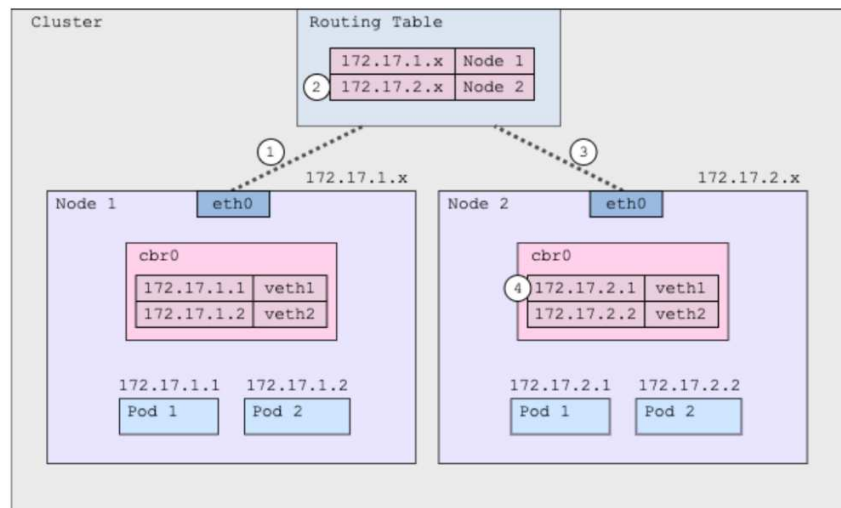


Рисунок 1.4 – Два Pod-и на різних вузлах взаємодіють один з одним за допомогою таблиці маршрутизації.

На цьому рівні існує таблиця маршрутизації, що містить діапазони IP-адрес для кожного вузла в Кластері. Наприклад, усі Поди у Вузлі 1 матимуть IP-адресу, подібну до 172.17.1.1 або 172.17.1.3, а всі Поди у Вузлі 2 матимуть IP-адреси, подібні до 172.17.2.1 або 172.17.2.3. Таблиця маршрутизації знає про цей шаблон та ідентифікує будь-яку IP-адресу у форматі 172.17.2.x як Pod у вузлі 2 та 172.17.1.x як Pod у вузлі 1 (2). Після того, як запит буде надіслано на правильний вузол (3), мережевий міст цього вузла визначить адресу свого

Pod (4) та надішле запит на Pod призначення, Pod 1. Pod-и є ефемерними компонентами, які можуть часто перезапускатися або замінюватися, отримуючи при цьому нові IP-адреси. Це ускладнює їх ідентифікацію та взаємодію з іншими Pod-ами в кластері [34]. Для вирішення цієї проблеми використовується концепція сервісів, яка забезпечує стабільну точку доступу до додатків, що працюють у Pod-ах. Сервіси дозволяють звертатися до Pod-ів через незмінний ідентифікатор (ім'я або тег), незалежно від змін їх IP-адрес. Реалізація сервісів здійснюється компонентом Kube-proxu на робочих вузлах.

Kubernetes вжив багато заходів безпеки та конфіденційності для забезпечення безпеки кластерів своїх користувачів. Центр безпеки Azure створив карту, яка охоплює поверхню атаки Kubernetes [35]. Ця матриця атак базується на фреймворку Mitre ATT&CK [36], який є базою знань про відомі методи кібератак, класифіковані за дев'ятьма тактиками. Матриця атак Kubernetes, показана на рисунку 1.5, окреслює методи, які зловмисники використовують для компрометації Кластера.

Початковий доступ	Виконання	Закріплення	Підвищення привілеїв	Ухилення від захисту	Доступ до облікових даних	Розвідка	Бокове переміщення	Вплив
Використання облікових даних хмари	Виконання в контейнері	Бекдор у контейнері	Привілейований контейнер	Очищення логів контейнера	Перелік секретів Kubernetes	Доступ до API-сервера	Доступ до хмарних ресурсів	Знищення даних
Скомпрометовані образи в реєстрі	bash/cmd всередині контейнера	Доступний для запису hostPath mount	Прив'язка Cluster-Admin	Видалення подій Kubernetes	Монтування Service Principal	Доступ до API Kubelet	Service Account контейнера	Захоплення ресурсів
Файл kubeconfig	Новий контейнер	Kubernetes CronJob	hostPath mount	Схожість імен Pod/Container	Доступ до Service Account контейнера	Мережеве сканування	Внутрішня мережа кластера	Відмова в обслуговуванні
Вразливість застосунку	Експлуатація застосунку (RCE)		Доступ до хмарних ресурсів	Підключення через проксі-сервер	Облікові дані у конфігураційних файлах	Доступ до Kubernetes Dashboard	Облікові дані у конфігураційних файлах	
Відкритий Dashboard	SSH-сервер у контейнері					API метаданих інстансу	Монтування томів хоста	

Рисунок 1.5 – Матриця атак Kubernetes

Можна виділити основні атаки Kubernetes:

- початковий доступ – отримання доступу до кластера через облікові дані, відкриті точки доступу або скомпрометовані контейнерні образи;
- виконання – запуск шкідливого коду, наприклад шляхом створення контейнера або виконання команд (наприклад, `kubectl exec`) у вже існуючому

контейнері;

- збереження – забезпечення повторного доступу до кластера, зокрема через створення бекдор-контейнерів;
- ескалація привілеїв – отримання вищих прав доступу, наприклад через використання привілейованих контейнерів або ролей із розширеними дозволами;
- ухилення від захисту – приховування активності шляхом очищення журналів або маскування шкідливих компонентів під легітимні;
- доступ до облікових даних – отримання конфіденційної інформації, зокрема через доступ до Kubernetes Secrets;
- уявлення – дослідження кластера за допомогою API, мережі або інших внутрішніх компонентів;
- латеральний рух – переміщення всередині кластера після початкового доступу;
- вплив – порушення роботи системи, наприклад видалення даних, зміна конфігурацій, майнінг криптовалют або атаки відмови в обслуговуванні.

Кластерам Kubernetes дуже легко виявити зловмисникам, оскільки всі вони прослуховують низку чітко визначених портів [37]. Etcd, компонент усередині кластера Kubernetes, використовується як сховище ключів-значень, яке зберігає стан кластера у вигляді даних конфігурації всіх вузлів і контейнерів. Якщо зловмисник отримає доступ до цього компонента, він може легко дослідити весь кластер. Оскільки налаштування Kubernetes за замовчуванням не дозволяє шифрування, секрети, що зберігаються всередині Etcd, знаходяться у формі відкритого тексту. Деякі користувачі можуть надавати доступ до Etcd робочим вузлам без захисту транспортного рівня, автентифікації та авторизації [38]. Оскільки Etcd прослуховує порт 2379, зловмисникам легко його знайти, оскільки цей порт індексується Shodan: пошуковою системою для пристроїв, підключених до Інтернету [39]. Для захисту Etcd необхідно вживати заходів безпеки, такі як обов'язкове

ввімкнення опції, яка вимагає, щоб усі, кому потрібен доступ до нього, мали дійсні сертифікати, також навколо нього встановлено брандмауер, щоб ускладнити доступ зломисникам [40]. Усі кластери Kubernetes містять компонент під назвою API Server. API Server – це контейнер, який є основною точкою входу до Кластера. Зв'язок між усіма іншими компонентами також відбувається через API Server. API має безпечну кінцеву точку, яку можна підключити через порт 6443, та небезпечну кінцеву точку через порт 8080. Якщо зломисник отримає доступ до небезпечної кінцевої точки, він може уникнути всіх існуючих протоколів автентифікації та авторизації та зробити API-запити до порту. Хоча налаштування Кластера за замовчуванням вимикають цю незахищену кінцеву точку, зломисник все ще може змінити цю конфігурацію та отримати доступ до внутрішньої інформації в Кластері[41]. Оскільки це єдина кінцева точка, доступна для громадськості, існує багато заходів безпеки, які впроваджуються для того, щоб користувач міг отримати доступ до Кластера через порт API. Для безпеки API реалізовано протокол транспортного рівня, і всі користувачі, які запитують доступ, повинні пройти автентифікацію та авторизацію перед отриманням доступу. Запит HTTP перевіряється, і система перевіряє, чи Кластер знає користувача, за допомогою паролів, сертифікатів або токенів. Якщо користувач дійсний, перевіряється запит, який він намагається зробити. Оскільки Kubernetes використовує контроль доступу на основі ролей, система переконується, що існуючих дозволів, наданих користувачеві, достатньо для виконання його запиту. Після того, як запит був авторизований та автентифікований, його має перевірити контролер доступу. Цей контролер гарантує, що будь-який запит, що надходить ззовні, відповідає набору правил. Наприклад, кожен Kubelet, компонент, який знаходиться в кожному вузлі та передає запити API до місця призначення, може надсилати запити лише до подів у межах свого власного вузла, а не до жодного іншого поду [42, 43]. Однак ці заходи безпеки не завжди успішні, оскільки все ще можливо випадково розкрити цей порт зломисникам. Окрім отримання доступу до кластера через відкриті кінцеві

точки, зловмисники можуть потрапити всередину за допомогою контейнерів. Поди Kubernetes запускають контейнери з різними програмами. За замовчуванням токен, який надає доступ до API Kubernetes, призначається контейнеру. Якщо RBAC не налаштовано, то токен може надати контейнеру привілеї рівня адміністратора [44]. Якщо один із контейнерів буде скомпрометовано, то зловмисник матиме доступ до поду, на якому він розміщений, та всієї інформації в ньому. У найгірших випадках, якщо цей скомпрометований контейнер має привілейований прапорець, то зловмисник може використовувати ці привілеї для отримання доступу до решти Кластера та всіх даних у ньому [45]. Щоб уникнути цього, користувачам рекомендується налаштувати RBAC та уникати запуску контейнерів як привілейованих. Незважаючи на впроваджені механізми безпеки та захисту, актуальним завданням залишається вдосконалення методів виявлення зловмисників, які вже отримали доступ до кластера.

1.3 Системи захисту хмарних сервісів та методи штучного інтелекту в системах виявлення аномалій

Незважаючи на свої переваги, Kubernetes створює складні проблеми безпеки, властиві контейнерним середовищам. Ці проблеми впливають з динамічної природи розгортання контейнерів, де робочі навантаження є тимчасовими та розподіленими між кількома вузлами [46]. Автори [47] визначають кілька критичних проблем безпеки, унікальних для оркестрації контейнерів, включаючи розширені поверхні атаки, вразливості зв'язку між pod-to-pod та ризики привілейованих контейнерів. Багатокористувацькі аспекти кластерів Kubernetes ще більше посилюють ці проблеми, оскільки робочі навантаження з різних програм або команд спільно використовують базову інфраструктуру та ресурси ядра.

Традиційні парадигми забезпечення інформаційної безпеки, що історично орієнтовані на статичні інфраструктурні моделі, виявляють

критичну невідповідність сучасним вимогам захисту динамічних середовищ Kubernetes. Конвенційні моделі безпеки, засновані на концепції мережевого периметра, не здатні враховувати ефемерну природу контейнеризованих об'єктів та безперервну зміну станів усередині кластерів оркестрації. Суттєвим обмеженням ретроспективних інструментів моніторингу є відсутність глибокої видимості внутрішніх процесів контейнерів та нездатність до аналізу складних інтерфейсних взаємозв'язків між об'єктами Kubernetes.

Зазначений розрив між статичними механізмами контролю та стохастичною природою сучасних контейнеризованих середовищ зумовлює формування латентних зон уразливості, що створює передумови для реалізації складних цілеспрямованих атак. Еволюція ландшафту кіберзагроз вимагає впровадження прецизійних методів виявлення та проактивного моніторингу, що адаптовані до специфіки архітектури Kubernetes. У контексті зростаючої ролі контейнеризованих навантажень для критичних бізнес-процесів, виникає об'єктивна необхідність у механізмах безпеки, здатних забезпечувати інтегральну видимість поведінки систем під час виконання та миттєву адаптацію до динамічних розгортань. Такі розширені можливості мають реалізовуватися на рівні як прикладного програмного забезпечення, так і системної інфраструктури, здійснюючи комплексний аудит не лише життєвого циклу контейнерів, а й самого рівня оркестрації.

Середовища Kubernetes стикаються з широким спектром вразливостей безпеки, які активно використовують зловмисники. Автор [48] класифікує ці вектори атак на кілька різних областей, включаючи неправильні конфігурації на сервері API Kubernetes, недостатньо захищені бази даних etcd та скомпрометовані образи контейнерів, що показано в таб. 1.1. Динамічний характер розгортання Kubernetes створює унікальні проблеми безпеки, оскільки інфраструктура постійно розвивається зі створенням, оновленням та знищенням робочих навантажень. Зовнішні зловмисники часто атакують відкриті панелі інструментів Kubernetes, небезпечно налаштовані кінцеві

точки kubelet та вразливі середовища виконання контейнерів. Ці точки входу можуть надати зловмисникам початковий доступ до ресурсів кластера, що потенційно може призвести до ескалації привілеїв та горизонтального переміщення по всій інфраструктурі.

Таблиця 1.1 – Поширені вразливості безпеки Kubernetes та вектори атак [49]

Категорія вразливості	Приклади векторів атак (Дані)	Потенційний вплив (Наслідки)
Неправильна конфігурація API-сервера	Експоновані Kube-протоколи, незахищені Dashboard, анонімні запити до API	Несанкціонований доступ до управління кластером, витік метаданих
Вразливості середовища виконання	Експлуатація системних викликів ядра , вихід за межі контейнера через вразливі рантайми	Несанкціонована ескалація привілеїв до рівня хоста
Слабкі місця ланцюга поставок	Використання образів із публічних репозиторіїв без підпису , шкідливі Sidecar-контейнери	Виконання довільного коду, розгортання криптомайнерів або бекдорів
Прогалини в безпеці мережі	Відкритий трафік між Pod-ами за замовчуванням, відсутність сегментації через NetworkPolicies	Латеральне переміщення всередині мережі кластера, перехоплення трафіку
Недостатній RBAC	Облікові записи сервісів з правами cluster-admin, помилкові прив'язки ролей	Захоплення контролю над ресурсами інших користувачів або просторів імен
Недоліки управління секретами	Зберігання паролів/ключів у ConfigMaps, використання Base64 без шифрування вшиті API-ключі	Компрометація облікових даних, доступ до зовнішніх хмарних ресурсів та БД

Сучасні системи безпеки Kubernetes включають багаторівневі підходи, що охоплюють заходи безпеки на рівні інфраструктури, кластера, контейнера та програми. Автор [50] окреслює комплексний набір практик безпеки, які він називає «XI Заповідями безпеки Kubernetes», що стосуються критичних аспектів від автентифікації та авторизації до мережесих політик та захисту під час виконання. Ці системи рекомендують впровадження контролю доступу на основі ролей, ізоляції простору імен, сегментації мережі за допомогою NetworkPolicies та суворих контекстів безпеки pod. Багато організацій застосовують підходи, орієнтовані на відповідність, узгоджені з галузевими стандартами, такими як CIS Kubernetes Benchmarks, які забезпечують вимірювані засоби контролю безпеки для посилення розгортань Kubernetes від поширених загроз. Незважаючи на доступні системи безпеки, стандартні розгортання Kubernetes часто містять значні прогалини в безпеці. Автор [51] визначає кілька поширених недоліків, зокрема стандартну дозвоільну позицію безпеки багатьох компонентів Kubernetes, неадекватний моніторинг дій під час виконання контейнерів та недостатню перевірку розгорнутих артефактів. Багато організацій мають труднощі з впровадженням належного управління секретами, що призводить до ненавмисного розкриття облікових даних через змінні середовища або карти конфігурацій. Крім того, складні конфігурації RBAC часто призводять до надмірно дозвоільних політик доступу, які порушують принцип найменших привілеїв. Ці прогалини створюють можливості для зловмисників скомпрометувати кластери за допомогою методів, що обходять звичайні засоби контролю безпеки, особливо враховуючи швидкі темпи оновлень як програм, так і базової платформи Kubernetes.

Багатокластерні та гібридні середовища Kubernetes створюють додаткові проблеми безпеки, окрім тих, що виникають при розгортанні в одному кластері. Автори [52] підкреслюють складність підтримки узгоджених політик безпеки в гетерогенних середовищах, що охоплюють кількох хмарних постачальників та локальну інфраструктуру. Ці розподілені архітектури

створюють складні проблеми управління ідентифікацією, оскільки різні середовища можуть реалізовувати різні механізми автентифікації. Безпека мережі стає особливо складною в цих сценаріях, оскільки міжкластерний зв'язок має бути захищений без шкоди для продуктивності програм. Організації, що працюють у гібридних середовищах, також мають труднощі з досягненням єдиної видимості в різних розгортаннях, створюючи сліпі зони в моніторингу безпеки. Різноманітність цих середовищ ускладнює управління вразливостями, оскільки виправлення повинні бути координовані між кількома кластерами з потенційно різними конфігураціями та версіями компонентів Kubernetes.

Машинне навчання надає потужні теоретичні основи для виявлення аномальної поведінки в складних середовищах, таких як кластери Kubernetes. Автори [53] описують, як алгоритми виявлення аномалій встановлюють базові моделі нормальної поведінки системи, що дозволяє ідентифікувати відхилення, які можуть свідчити про загрози безпеці. Ці підходи використовують статистичні методи, методи на основі відстані, моделі на основі щільності та архітектури нейронних мереж для розрізнення нормальної та аномальної активності. У середовищах Kubernetes моделі машинного навчання можуть аналізувати багатовимірні дані з метрик контейнерів, журналів API-сервера, мережевого трафіку та системних викликів, щоб встановити нормальні моделі поведінки, специфічні для характеристик робочого навантаження кожного кластера. Теоретична основа цих методів включає вилучення ознак з багатовимірних просторів даних, зменшення розмірності для обчислювальної ефективності та встановлення меж рішень, які відокремлюють нормальні операції від потенційних інцидентів безпеки. Впровадження безпеки на основі штучного інтелекту для Kubernetes вимагає надійних структур, здатних збирати, обробляти та аналізувати телеметричні дані у великих масштабах. Автори [54] описують архітектури, які збирають та аналізують кілька потоків даних, включаючи журнали аудиту kube, метрики виконання контейнерів, мережеві потоки та телеметрію на рівні хоста. Ці

фреймворки реалізації зазвичай використовують потокові конвеєри даних, які приймають телеметричні дані, попередньо обробляють та нормалізують їх для використання моделлю, а потім передають їх у навчені моделі для аналізу в режимі реального часу. Ефективні реалізації для середовищ Kubernetes повинні враховувати ефемерну природу контейнерів, відстежуючи об'єкти під час перезапусків, зберігаючи при цьому контекст щодо їхньої очікуваної поведінки. Багато фреймворків використовують ансамблеві підходи, які поєднують кілька алгоритмів виявлення, спеціалізованих для різних типів загроз, що дозволяє більш надійно ідентифікувати інциденти безпеки за різними векторами атак, мінімізуючи при цьому хибнопозитивні результати, які можуть призвести до втоми від сповіщень.

Рішення безпеки на основі штучного інтелекту пропонують можливості виявлення складних загроз, спрямованих на середовища Kubernetes, які традиційні системи на основі правил часто пропускають. Автори [55] підкреслюють, як підходи машинного навчання можуть виявляти тонкі індикатори атак, таких як криптоджекінг, коли контейнери скомпрометовані для майнінгу криптовалюти з використанням ресурсів кластера. Ці моделі аналізують моделі використання ресурсів, поведінку мережевого зв'язку та послідовності виконання процесів для виявлення аномальної активності, що вказує на загрози. Для атак ескалації привілеїв системи штучного інтелекту відстежують незвичайні зміни дозволів, підозрілі виклики API та аномальний доступ до конфіденційних ресурсів у кластері. Характер цих систем виявлення в реальному часі дозволяє швидко реагувати на нові загрози, причому багато реалізацій забезпечують автоматизовані можливості стримування, які можуть ізолювати підозрілі pod-системи або обмежувати їхній доступ до мережі, щоб запобігти горизонтальному переміщенню по інфраструктурі кластера.

Вибір між підходами до навчання з учителем та без учителя суттєво впливає на ефективність рішень безпеки Kubernetes на основі штучного інтелекту. Автор [56] порівнює ці підходи, зазначаючи, що навчання з учителем вимагає маркованих наборів даних як про звичайну, так і про

шкідливу діяльність, специфічну для середовищ Kubernetes. Хоча методи з учителем можуть досягти високої точності для відомих шаблонів атак, вони мають труднощі з загрозами нульового дня та новими методами атак. І навпаки, підходи до навчання без учителя можуть виявляти раніше невідомі аномалії, визначаючи відхилення від усталених нормальних шаблонів, не вимагаючи маркованих даних про атаки. Однак ці методи зазвичай генерують більше хибнопозитивних результатів і вимагають додаткового контекстного аналізу, щоб визначити, чи виявлені аномалії є справжніми загрозами безпеці. Багато зрілих реалізацій безпеки Kubernetes використовують гібридні підходи, які поєднують моделі з учителем для виявлення відомих сигнатур атак з методами без учителя, які виявляють підозрілі поведінкові відхилення, забезпечуючи глибокий захист як від усталених, так і від нових загроз для контейнерних середовищ.

В таблиці 1.2 представлено основні характеристики та атрибути методів машинного навчання у задачах кібербезпеки

Розширений фільтр пакетів Берклі являє собою революційну технологію, яка забезпечує програмований доступ до різних підсистем ядра без необхідності модифікації ядра або завантаження модулів. Автори [57] описують, як eBPF еволюціонував від своїх витоків як простого механізму фільтрації пакетів до складної технології, здатної прив'язувати програми до різних перехоплювачів ядра, включаючи системні виклики, точки входу та виходу функцій, мережеві події та точки трасування ядра.

Ця еволюція зробила eBPF особливо цінним для програм моніторингу безпеки, оскільки вона дозволяє інструментам безпеки отримувати доступ до низькорівневої системної інформації з мінімальними накладними витратами. Архітектура eBPF включає механізм перевірки, який гарантує, що програми не можуть призвести до збою або компрометації ядра, що робить його ідеальним для програм безпеки, які потребують як безпеки, так і продуктивності.

Таблиця 1.2 – Характеристики та атрибути методів машинного навчання у задачах кібербезпеки [58]

Метод навчання	Основний опис (Дані)	Обмеження / Виклики	Типові алгоритми
Навчання з учителем (Supervised)	Використовує повністю розмічені набори даних, де кожен вхідний сигнал має відповідну мітку класу (атака/норма)	Необхідність постійної актуалізації баз атак; нездатність виявляти нові типи загроз	Лінійна регресія, дерева рішень, випадковий ліс, SVM
Навчання без учителя (Unsupervised)	Працює з нерозміченими даними; самостійно знаходить приховані структури або кластери в інформаційних потоках	Складність інтерпретації результатів; чутливість до шумів у вхідних даних	Кластеризація (K-means), метод головних компонент, SVD
Напівкероване навчання (Semi-supervised)	Комбінований підхід: використовує невелику кількість розмічених даних разом із великим обсягом нерозмічених	Можливість накопичення помилок при неправильній початковій розмітці	GANs, Label Propagation, активне навчання
Навчання з підкріпленням (Reinforcement)	Навчання через систему штрафів та винагород у процесі взаємодії агента з динамічним середовищем	Високі обчислювальні витрати на етапі навчання; складність функції винагороди	Q-learning, Deep Q-Network, алгоритми SARSA

Технологія надає можливості для самоаналізу запущених програм, моніторингу мережевого трафіку та спостереження за шаблонами системних викликів – усіх необхідних компонентів для комплексного моніторингу безпеки в контейнерних середовищах.

У середовищах Kubernetes eBPF забезпечує безпрецедентну видимість діяльності контейнерів на рівні ядра. Основні можливості технології eBPF у доменах моніторингу безпеки представлені в табл.1.3.

Автор [59] пояснює, як програми eBPF можна приєднувати до різних перехоплювачів ядра для моніторингу поведінки контейнерів у просторах імен, забезпечуючи видимість, якої не можуть досягти традиційні інструменти моніторингу, орієнтовані на контейнери. У контекстах Kubernetes реалізації eBPF зазвичай розгортають агентів на кожному робочому вузлі для збору телеметричних даних про діяльність pod, мережевий зв'язок та доступ до файлової системи. Ці агенти використовують карти eBPF для ефективного обміну даними між ядром та простором користувача, що дозволяє корелювати події між контейнерами та pod. Видимість на рівні ядра дозволяє інструментам безпеки виявляти спроби виходу з контейнера, ескалацію привілеїв та несанкціонований доступ до ресурсів шляхом моніторингу системних викликів та діяльності процесів. Ця комплексна видимість особливо цінна в багатокористувацьких кластерах Kubernetes, де робочі навантаження з різних доменів довіри виконуються на спільній інфраструктурі.

Моніторинг безпеки на основі eBPF пропонує значні переваги в продуктивності порівняно з традиційними підходами до безпеки. Автори [60] підкреслюють ефективність програм eBPF, які виконуються безпосередньо в контексті ядра без необхідності перемикання контексту або копіювання даних між ядром та простором користувача. Ця архітектурна перевага призводить до суттєво нижчих накладних витрат порівняно з традиційними підходами до моніторингу, які покладаються на перехоплення системних викликів або модулів ядра. У середовищах Kubernetes, де продуктивність та ефективність використання ресурсів є критично важливими, інструменти моніторингу eBPF

вводять мінімальні накладні витрати на процесор та пам'ять, забезпечуючи при цьому повну видимість [61]. Характеристики продуктивності особливо важливі для виробничих робочих навантажень, де моніторинг безпеки не повинен суттєво впливати на швидкість реагування програм або використання ресурсів. Здатність eBPF фільтрувати та агрегувати дані в ядрі ще більше зменшує накладні витрати, мінімізуючи обсяг даних, які необхідно передати до механізмів аналізу простору користувача.

Таблиця 1.3 – Можливості технології eBPF у доменах моніторингу безпеки

Об'єкт моніторингу	Типи даних та параметри (Raw Data)	Цільове призначення в системі захисту
Системні виклики (Syscalls)	ID виклику, аргументи, ідентифікатор процесу (PID), тимчасові мітки	Ідентифікація аномальних запусків бінарних файлів або спроб модифікації критичних системних конфігурацій
Мережевий трафік	IP-адреси джерела/призначення, порти, протоколи (TCP/UDP), розмір пакетів, прапори	Виявлення прихованих каналів зв'язку (C2), ексфільтрації даних та сканування внутрішньої мережі
Доступ до файлів	Шляхи до файлів, дескриптори, типи операцій (Read/Write/Delete), права доступу	Детекція спроб зчитування секретів Kubernetes, зміна конфігураційного файлу kubelet або etcd
Процесор і пам'ять	Використання CPU (user/kernel space), споживання RAM, частота перемикання контексту	Раннє виявлення криптомайнінгу та атак типу «відмова в обслуговуванні» (DoS) на рівні контейнера
Події оркестрації	Назва Pod-a, Namespace, ідентифікатор контейнера, образи	Зіставлення низькорівневих подій ядра з логічними об'єктами Kubernetes для точної локалізації загрози

Архітектура нульової довіри являє собою зміну парадигми в підходах до безпеки, відходячи від моделей на основі периметра до постійної перевірки кожного запиту на доступ незалежно від джерела. Автори [62] окреслюють основні принципи нульової довіри, включаючи усунення неявної довіри, постійну перевірку та контроль доступу з найменшими привілеями – всі вони особливо актуальні для середовищ Kubernetes, де робочі навантаження є динамічними та розподіленими. У контексті Kubernetes принципи нульової довіри перекладаються на обробку кожного поду, сервісу та компонента як потенційно скомпрометованих, що вимагає явної автентифікації та авторизації для всіх взаємодій. Автор [63] описує, як реалізації архітектур нульової довіри в Kubernetes використовують сервісну сітку, контролери доступу та політики безпеки для забезпечення детального контролю доступу на кількох рівнях стеку. Ці реалізації встановлюють межі довіри між просторами імен та забезпечують сувору перевірку ідентифікації робочого навантаження за допомогою таких механізмів, як взаємна автентифікація TLS та криптографічна атестація образів контейнерів та конфігурацій.

Системи на основі штучного інтелекту дозволяють динамічне коригування політик на основі інформації про загрози в режимі реального часу, що дозволяє системам безпеки Kubernetes розвиватися у відповідь на зміну умов. Автори [64] пояснюють, як моделі машинного навчання аналізують дані телеметрії для виявлення потенційних загроз та автоматично змінюють політики безпеки для зменшення ризиків без втручання людини. Ці динамічні системи політик використовують підходи до навчання з підкріпленням для оптимізації конфігурацій безпеки на основі спостережуваних моделей атак та поведінки системи. У середовищах Kubernetes ця можливість проявляється як автоматичне коригування конфігурацій NetworkPolicies, PodSecurityPolicies та RBAC у відповідь на виявлені аномалії або нові загрози. Автор [65] описує реалізації, де точки застосування політик постійно отримують оновлені директиви безпеки на основі аналізу даних телеметрії кластера за допомогою штучного інтелекту,

що дозволяє швидко реагувати на нові загрози, зберігаючи при цьому безперервність роботи для легітимних робочих навантажень. Автоматизована перевірка ідентифікації робочого навантаження є критичним компонентом архітектур Zero Trust у середовищах Kubernetes. Автори [66] описують, як механізми криптографічної атестації встановлюють ідентифікацію робочого навантаження на основі кількох факторів, включаючи джерело розгортання, атрибути конфігурації, поведінку під час виконання та облікові дані облікового запису служби. Ці структури використовують контролери доступу Kubernetes для перевірки ідентифікації робочого навантаження під час розгортання та продовжують моніторинг потенційної компрометації ідентифікації за допомогою аналізу поведінки під час виконання. Автори [67] окреслює реалізації, які використовують сітки сервісів для забезпечення взаємної автентифікації TLS між службами, гарантуючи, що кожне спілкування автентифіковане та авторизоване на основі ідентифікації робочого навантаження, а не мережевого розташування. Розширені реалізації включають поведінкові профілі для кожного типу робочого навантаження, що дозволяє системам штучного інтелекту (ШІ) виявляти, коли контейнери відхиляються від очікуваних моделей поведінки, що потенційно вказує на компрометацію. Ці структури ідентифікації робочого навантаження зазвичай інтегруються із системами управління секретами, центрами сертифікації та механізмами політик для створення комплексних конвеєрів перевірки ідентифікації.

Інтеграція штучного інтелекту з моніторингом на основі eBPF являє собою потужний підхід до комплексної безпеки Kubernetes. Автори [68] пропонують довідкову архітектуру, яка поєднує глибоку видимість системи eBPF з аналітичними можливостями ШІ, створюючи багаторівневі системи захисту для контейнерних середовищ. Ця архітектура зазвичай складається з агентів eBPF, розгорнутих на кожному вузлі кластера Kubernetes, які збирають низькорівневі телеметричні дані, включаючи системні виклики, мережеві потоки та активність процесів. Зібрані дані надходять до централізованої

аналітичної платформи, де моделі машинного навчання обробляють та аналізують їх на наявність аномалій безпеки. Автори [69] описують, як ці архітектури реалізують петлі зворотного зв'язку між рівнем аналітики ІІІ та агентами eBPF, що дозволяє адаптивний моніторинг на основі виявлених загроз. Довідкова архітектура включає компоненти для збору даних, попередньої обробки, вилучення ознак, виявлення аномалій, класифікації та оркестрації відповідей. Такий архітектурний підхід дозволяє командам безпеки виявляти складні атаки, спрямовані на кластери Kubernetes, шляхом співвіднесення поведінки низькорівневих систем із шаблонами роботи програм вищого рівня та діями в масштабах кластера.

Ефективна обробка телеметрії безпеки в реальному часі вимагає складних моделей потоку даних, які балансують повноту з ефективністю. Автори [70] описують архітектури потоку даних, які використовують здатність eBPF фільтрувати та агрегувати дані в ядрі, зменшуючи обсяг інформації, яку необхідно передавати в простір користувача для аналізу ІІІ. Ці моделі зазвичай використовують фреймворки потокової обробки, які обробляють безперервні потоки даних від розподілених агентів eBPF по всьому кластеру. Автори [71] описують реалізації, які обробляють телеметричні дані через кілька етапів, включаючи нормалізацію, збагачення контекстною інформацією, вилучення ознак та пакетну агрегацію для навчання моделі. Розширені реалізації використовують методи аналізу часових рядів для виявлення часових закономірностей у подіях безпеки, які можуть свідчити про скоординовані атаки. Моделі потоку даних повинні вирішувати проблему підтримки контексту протягом ефемерних життєвих циклів контейнерів, гарантуючи, що аналітика безпеки може відстежувати об'єкти, незважаючи на динамічний характер середовищ Kubernetes. Ці системи зазвичай використовують розподілені поточкові платформи, які забезпечують масштабованість, відмовостійкість та можливості обробки з низькою затримкою, необхідні для виявлення загроз у реальному часі. Розгортання інтегрованих рішень безпеки на основі штучного інтелекту та

eBPF у корпоративних середовищах Kubernetes створює унікальні проблеми, які необхідно вирішувати шляхом ретельного планування. Автори [72] наголошують на важливості впровадження цих рішень з мінімальним порушенням роботи існуючих робочих навантажень та інфраструктури. Архітектури розгортання зазвичай включають DaemonSets, щоб забезпечити роботу агентів eBPF на кожному вузлі, з ретельним розподілом ресурсів для запобігання впливу на продуктивність виробничих програм. Корпоративні впровадження повинні вирішувати проблеми сумісності між різними дистрибутивами Kubernetes, версіями ядра та середовищами виконання контейнерів. Автори [73] виділяють міркування щодо суверенітету даних та наслідків для конфіденційності під час розгортання цих рішень для моніторингу, особливо в середовищах з кількома орендарями, де робочі навантаження можуть мати різні вимоги до безпеки.

Експлуатація інтегрованих рішень безпеки на основі штучного інтелекту та eBPF у великих масштабах створює кілька проблем, що потребують спеціальних методів оптимізації. Автор [74] визначає ключові операційні проблеми, включаючи управління ресурсомісткістю програм eBPF, обробку обчислювальних вимог машинного навчання в реальному часі та підтримку продуктивності під час інцидентів безпеки. Методи оптимізації включають вибіркову інструментарій, який зосереджує моніторинг eBPF на високоризикових системних викликах та діях, а не на комплексному трасуванні. Автори [75] описують оптимізацію компонентів штучного інтелекту, включаючи квантування моделі, вибір ознак для зменшення розмірності та граничний висновок, який розподіляє аналітичні робочі навантаження по кластеру. Ці оптимізації гарантують, що моніторинг безпеки не впливає суттєво на продуктивність програм або доступність ресурсів – критична вимога для виробничих середовищ Kubernetes, де рішення безпеки повинні ефективно працювати разом із критично важливими для бізнесу робочими навантаженнями.

Проведений аналіз предметної області показав, що сучасні

контейнеризовані та мікросервісні середовища характеризуються високою динамічністю, значною кількістю різномірних джерел телеметричних даних і постійною еволюцією кіберзагроз. Незважаючи на ефективність існуючих механізмів захисту, таких як моделі нульової довіри, мережева сегментація, RBAC та сигнатурні засоби моніторингу, їх застосування не забезпечує своєчасного виявлення нових і раніше невідомих атак у динамічних середовищах Kubernetes. Встановлено, що підвищення ефективності виявлення аномалій потребує комплексного аналізу мережевих подій, API-журналів, системної телеметрії та поведінкових характеристик користувачів із використанням інтелектуальних методів обробки даних. Це обумовлює необхідність розроблення моделі та методів інтелектуальної системи виявлення аномалій у хмарних сервісах, здатної забезпечити адаптивний багаторівневий аналіз подій безпеки, виявлення як відомих, так і невідомих загроз, а також функціонування в режимі реального часу в умовах високодинамічного хмарного середовища.

Висновки до розділу 1

У першому розділі дисертаційної роботи проведено аналіз сучасного стану проблеми забезпечення безпеки хмарних сервісів та оцінювання ризиків в умовах динамічної мікросервісної архітектури.

Розглянуто основні теоретичні підходи до побудови хмарної інфраструктури, зокрема еволюцію від монолітних систем до мікросервісів на базі оркестратора Kubernetes. Показано, що ключовими параметрами ефективності такої інфраструктури є висока доступність, еластичність ресурсів, швидкість відновлення після збоїв, а також показники затримок і пропускну здатності мережевої взаємодії.

Проаналізовано сучасні методи та інструменти гарантування безпеки в середовищі Kubernetes, серед яких особливу увагу приділено моделі нульової довіри, взаємній автентифікації, сегментації мережі через Network Policies та

RBAC. Встановлено, що хоча зазначені механізми дозволяють комплексно оцінювати захищеність хмарних стратегій, їхнє застосування в межах класичних статичних моделей безпеки має суттєві обмеження.

Доведено, що традиційні підходи до захисту, засновані на статичних правилах брандмауерів та сигнатурному аналізі, не забезпечують достатньої адаптивності до динамічного контейнерного середовища. Основними перешкодами є ефемерність IP-адрес подів, висока інтенсивність генерації логів, складність моніторингу взаємозв'язків між мікросервісами та складність інтеграції низькорівневої телеметрії ядра в реальному часі.

Обґрунтовано доцільність переходу до сучасних підходів аналізу загроз на основі методів машинного навчання, технології eBPF та обробки великих даних. Встановлено, що інтеграція традиційних мережевих показників із системними викликами ядра Linux, логами API-сервера та поведінковими профілями додатків дозволяє суттєво підвищити точність прогнозування аномальної активності та покращити якість прийняття рішень щодо нейтралізації вторгнень.

Результати проведеного аналізу наукових джерел дозволили визначити напрями вдосконалення інформаційних технологій підтримки прийняття рішень у сфері кібербезпеки. Зокрема, підтверджено необхідність використання методів глибокого навчання та алгоритмів аналізу аномалій на базі eBPF для підвищення адаптивності систем виявлення вторгнень в умовах високої невизначеності трафіку.

Сформовані висновки стали теоретичною основою для подальшого дослідження, спрямованого на розробку методів виявлення аномалій в інфраструктурі Kubernetes та створення інтелектуальної моделі інформаційної технології захисту хмарних сервісів, що розглядаються у наступних розділах дисертаційної роботи.

2 ВИЯВЛЕННЯ АНОМАЛІЙ В ІНФРАСТРУКТУРІ KUBERNETES

2.1 Метод виявлення аномалій в інфраструктурі Kubernetes

Для виявлення аномалій у функціонуванні Kubernetes запропоновано використання моделі LSTM Autoencoder, яка є однією з найбільш ефективних архітектур для задач виявлення аномалій [76]. Структура розробленої моделі LSTM Autoencoder наведена на рисунку 2.1.

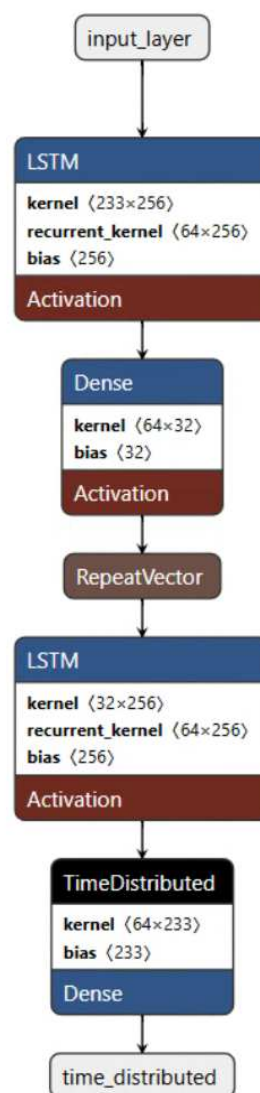


Рисунок 2.1 – Архітектура LSTM Autoencoder для виявлення аномалій в інфраструктурі Kubernetes

Після навчання розроблена модель LSTM Autoencoder на підмножині даних, що відповідають нормальній роботі системи, використовується для відновлення (реконструкції) вхідних часових послідовностей ознак. Нехай $X^{\text{scaled}} = \{X^{(1)}, X^{(2)}, \dots, X^{(N)}\}$ – множина масштабованих часових послідовностей, де кожен елемент $X^{(i)} \in \mathbb{R}^{T \times n}$ є послідовністю довжини T із n ознак.

Для кожної послідовності $X^{(i)}$ LSTM Autoencoder обчислює її реконструкцію

$$\hat{X}^{(i)} = D(E(X^{(i)})), \quad (2.1)$$

де E та D відповідно позначають відображення енкодера та декодера.

Відхилення між вхідною послідовністю та її реконструкцією кількісно оцінюється за допомогою середньоквадратичної помилки:

$$\varepsilon^{(i)} = \frac{1}{T \cdot n} \sum_{t=1}^T \sum_{j=1}^n (X_{t,j}^{(i)} - \hat{X}_{t,j}^{(i)})^2 \quad (2.2)$$

Отримане значення $\varepsilon^{(i)}$ інтерпретується як міра аномальності відповідної часової послідовності.

Для відокремлення нормальної та аномальної поведінки вводиться порогове значення θ . Значення порогу визначається на основі розподілу помилок реконструкції, отриманих для нормальної валідаційної вибірки[77]. Формально поріг задається як:

$$\theta = \arg \max F_1(I(\varepsilon > \tau), y) \quad (2.3)$$

де $I()$ – індикаторна функція,

y – істинні мітки класів (норма / атака) на валідаційній множині,

F_1 – гармонічне середнє точності та повноти.

Такий підхід дозволяє підібрати поріг, який забезпечує оптимальний баланс між виявленням атак і кількістю хибних спрацювань.

Для кожної нової часової послідовності рішення про її належність до аномальної поведінки приймається за правилом:

$$\hat{y}^{(i)} = \begin{cases} 1, \varepsilon^{(i)} > \theta, \\ 0, \varepsilon^{(i)} \leq \theta. \end{cases} \quad (2.4)$$

де $\hat{y}^{(i)} = 1$ відповідає аномальній або потенційно шкідливій активності, а $\hat{y}^{(i)} = 0$ – нормальній роботі системи.

Таким чином, процес реконструкції та порогової класифікації дозволяє звести задачу виявлення аномалій до аналізу відхилень від вивченої нормальної поведінки. Оскільки модель навчається виключно на нормальних даних, метод не залежить від повноти множини атак та зберігає здатність виявляти раніше невідомі або модифіковані загрози.

2.2 Опис тестового середовища для проведення експерименту

Фундаментом для проведення експериментальних досліджень було обрано архітектуру на базі Kubernetes – сучасного стандарту оркестрації, що забезпечує автоматизацію розгортання, масштабування та управління контейнеризованими додатками[78].

Вибір даного середовища зумовлений його динамічною природою та складністю мережових взаємодій, що робить його репрезентативним майданчиком для тестування систем виявлення аномалій. Kubernetes має багаторівневу структуру, що складається з Control Plane та Worker Nodes і зображена на рисунку 2.2.

Control Plane відповідає за управління кластером: планування подів, контроль стану, масштабування та підтримання узгодженого функціонального стану всієї системи.

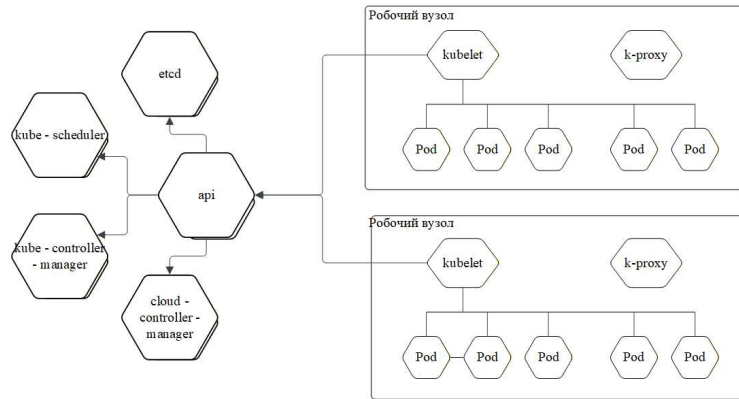


Рисунок 2.2 – Тестова Структура Kubernetes

У структурі Kubernetes важливу роль відіграють Worker Nodes, які забезпечують виконання обчислювального навантаження. Саме на робочих вузлах розгортаються та функціонують поди, у яких запускаються контейнери застосунків. Кожен вузол містить необхідні системні компоненти – такі як kubelet, kube-proxy та контейнерний рантайм, що забезпечують керування життєвим циклом контейнерів, взаємодію з Control Plane та маршрутизацію мережевого трафіку. Таким чином, робочі вузли формують розподілене обчислювальне середовище, яке масштабовано обробляє запити та виконує сервіси користувача. Kubernetes може відновлювати збої завдяки вбудованим механізмам контролю стану подів та вузлів. Якщо контейнер або Pod виходить із ладу, Kubernetes автоматично перезапускає його, створює новий екземпляр або переносить навантаження на інший здоровий вузол. Така самовідновлювана архітектура підвищує надійність роботи застосунків, забезпечує стійкість до відмов та зменшує час простою системи.

Kubernetes Pod виступає базовою одиницею розгортання та виконання застосунків. Pod функціонує в межах Worker Node і спільно з іншими Pod-ами використовує мережевий простір імен вузла для комунікацій.

Спостереження за мережевим простором імен конкретного робочого вузла дає змогу отримати повну картину мережевого трафіку, що циркулює між Pod-ами на цьому вузлі. Це робить можливим виявлення нетипових або потенційно шкідливих мережевих взаємодій у кластері.

У випадку, якщо зломисник отримує доступ до кластера, він змушений здійснювати внутрішнє переміщення, ідентифікувати цінні інформаційні ресурси та ініціювати їх переміщення за межі інфраструктури [79]. Усі ці дії неминуче пов'язані з використанням внутрішньої мережі кластера, тому моніторинг трафіку на міжкомпонентному рівні є ключовим механізмом для своєчасного виявлення аномалій і потенційних порушень безпеки.

Усередині мережевого простору імен кожного робочого вузла Kubernetes існує спеціальний virtual bridge, який виступає центральним елементом маршрутизації трафіку між Pod-ами та зовнішньою мережею. Цей міст забезпечує прозору комунікацію між усіма Pod-ами, розгорнутими на вузлі, і виконує функцію агрегатора мережеских потоків. Віртуальний міст підключений до всіх віртуальних Ethernet-інтерфейсів контейнерів, що входять до складу подів, і відповідає за передачу пакетів як всередині вузла, так і до інших вузлів кластера через оверлейні або фізичні мережеві інтерфейси.

Як показано на рисунку 2.3, віртуальний мережевий міст має прямий доступ до всіх підключених до нього інтерфейсів подів, що дозволяє контролювати весь трафік, що надходить і виходить із подів.

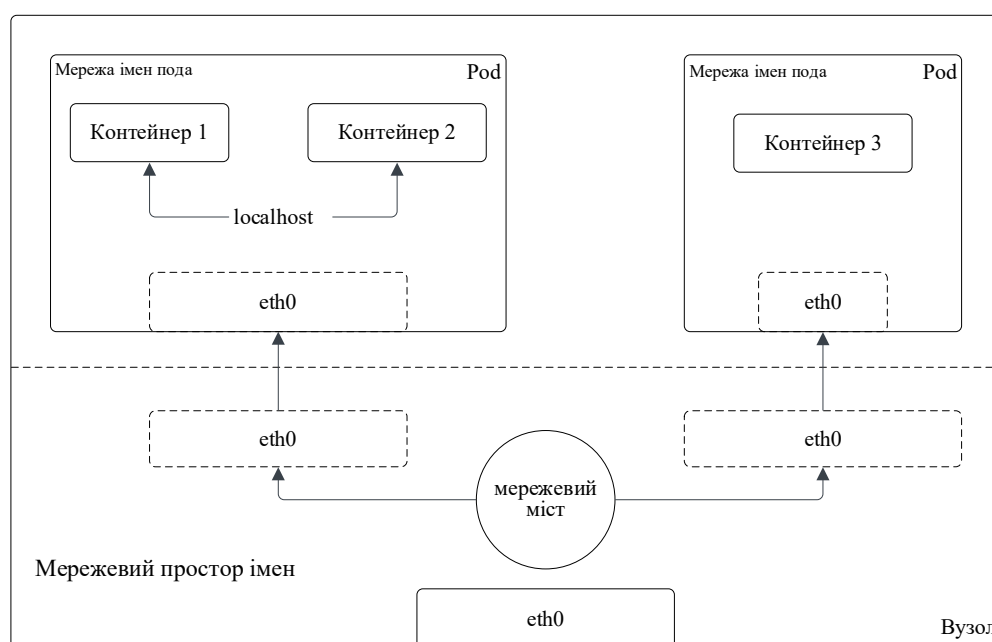


Рисунок 2.3 – Мережі між Pod-ами у вузлі

Прослуховування мережевих інтерфейсів на рівні віртуального мосту дозволяє здійснювати всебічний моніторинг всього трафіку, що проходить через робочий вузол, включаючи як внутрішньокластерні комунікації між Pod-ами, так і зовнішні з'єднання. Захоплений трафік може бути переданий до спеціалізованого Pod-а, на якому реалізовано алгоритм виявлення аномалій, що здійснює детальний аналіз мережевих потоків. Такий підхід дозволяє не лише виявляти аномальні або підозрілі патерни в пакеті даних, але й проводити глибинний аналіз взаємодії компонентів кластера, визначати потенційні загрози та своєчасно реагувати на них. Впровадження подібних механізмів підвищує прозорість внутрішньої мережі Kubernetes, забезпечує контроль над безпекою на рівні вузлів та сприяє підвищенню загальної надійності та стійкості кластерної інфраструктури [80]

У межах кластера Kubernetes мережевий трафік між Pod-ами та зовнішніми сервісами передається за допомогою набору стандартних мережевих протоколів, які забезпечують ефективну комунікацію, маршрутизацію та передачу даних.

Для дослідження було обрано набір даних [81] містить два набори даних для виявлення аномалій, зібрані в Kubernetes-кластері. Дані були отримані шляхом симуляції нормальної та шкідливої активності на двох різних вебзастосунках, розміщених у кластері. Кожен набір даних містить:

- мережеві дані, файли перехоплення пакетів та витягнуті TCP-потоки;
- метрики контейнерів, метрики cAdvisor, такі як використання CPU контейнера, кеш-пам'ять, відкриті мережеві сокети, кількість процесів і потоків тощо;
- метрики кластера Kubernetes, метрики KubeStateMetrics та інші показники, зокрема кількість доступних Pod для мікросервісу та квоти ресурсів.

Ці дані є частиною статті [82].

У процесі навчання моделі виявлення аномалій використовується багатовимірний простір ознак, що описує стан мережевої взаємодії,

контейнерів та Kubernetes-кластера. Формально повний простір ознак можна подати у вигляді об'єднання трьох множин:

$$X = F_{\text{net}} \cup F_{\text{ctr}} \cup F_{\text{k8s}}, \quad (2.5)$$

де F_{net} – множина мережових ознак,

F_{ctr} – множина метрик контейнера,

F_{k8s} – множина системних показників Kubernetes-кластера.

Множену мережних показників можна описати наступним чином:

$$F_{\text{net}} = \{f_1^{\text{net}}, f_2^{\text{net}}, \dots, f_m^{\text{net}}\}, \quad (2.6)$$

де кожен елемент відповідає певній характеристиці TCP-трафіку або статистичному параметру мережових потоків, отриманих із PCAP-даних. До цієї множини входять такі ознаки, як:

- кількість пакетів у прямому та зворотному напрямках;
- обсяг переданих байтів;
- інтенсивність трафіку (bytes rate, packets rate) ;
- характеристики TCP-заголовків (flags, window size, segment size) ;
- статистичні показники на кшталт варіації розміру сегментів або зміни довжини заголовка.

Ця група ознак описує поведінку мережевої взаємодії між сервісами у кластері.

Множина контейнерних метрик представляється наступним чином:

$$F_{\text{ctr}} = \{f_1^{\text{ctr}}, f_2^{\text{ctr}}, \dots, f_k^{\text{ctr}}\}, \quad (2.7)$$

і включає характеристики, що збираються агентом cAdvisor. До неї належать:

- використання CPU та доступні обмеження;
- обсяг виділеної та кешованої пам'яті;

- кількість відкритих дескрипторів;
- кількість мережевих сокетів;
- число процесів та потоків усередині контейнера.

Ці ознаки відображають поведінку контейнера як ізольованої обчислювальної одиниці.

Множина метрик стану кластера визначається як:

$$F_{k8s} = \{f_1^{k8s}, f_2^{k8s}, \dots, f_p^{k8s}\}, \quad (2.8)$$

куди входять дані, отримані за допомогою KubeStateMetrics та контролерів Kubernetes. До цієї групи належать:

- кількість доступних та бажаних Pod для кожного мікросервісу;
- статуси контейнерів та реплік;
- ліміти ресурсів та квоти (CPU, RAM) ;
- кількість перезапусків контейнерів;
- показники готовності та доступності сервісів.

Ці параметри описують стан оркестраційного середовища в цілому.

Після формування багатовимірному простіру ознак, що описує стан мережевої взаємодії, контейнерів та Kubernetes-кластера, було використано операцію імпутації пропущених значень. Для цього спочатку було обчислено вектори середніх значень для кожної ознаки $j \in \{1, \dots, n\}$, де визначається середнє значення за всіма наявними (непропущеними) елементами нормальної вибірки:

$$\mu_j = \frac{1}{|X_j|} \sum_{x \in X_j} x_j, \quad (2.9)$$

де

$$X_j = \{x \in X \mid x_j \neq \emptyset\}. \quad (2.10)$$

Після цього було виконано операцію заповнення пропущених значень,

яку можна формально можна задати як відображення:

$$\Phi: X \rightarrow \mathbb{R}^n, \quad (2.11)$$

де для кожного вектора $x \in X$ та кожної координати j :

$$\Phi(x)_j = \begin{cases} x_j, & \text{якщо } x_j \neq \emptyset, \\ \mu_j, & \text{якщо } x_j = \emptyset. \end{cases} \quad (2.12)$$

Операція імпутації для повної вибірки буде мати наступний вигляд

$$X^{\text{imp}} = \{\Phi(x) \mid x \in X\} \quad (2.13)$$

Наступним етапом підготовки даних було застосовано операцію масштабування (стандартизації) для повної імпутованої вибірки.

Нехай після виконання операції імпутації отримано множину:

$$X^{\text{imp}} = \{\Phi(x) \mid x \in X\} \subset \mathbb{R}^n, \quad (2.14)$$

де кожен елемент $x^{\text{imp}} = (x_1^{\text{imp}}, \dots, x_n^{\text{imp}})$ є повністю визначеним вектором ознак. Для кожної ознаки $j \in \{1, \dots, n\}$ визначаються середнє значення та стандартне відхилення:

$$\mu_j^{\text{imp}} = \frac{1}{|X^{\text{imp}}|} \sum_{x \in X^{\text{imp}}} x_j \quad (2.15)$$

$$\sigma_j^{\text{imp}} = \sqrt{\frac{1}{|X^{\text{imp}}|} \sum_{x \in X^{\text{imp}}} (x_j - \mu_j^{\text{imp}})^2} \quad (2.16)$$

Операцію масштабування визначимо як відображення:

$$\Psi: \mathbb{R}^n \rightarrow \mathbb{R}^n, \quad (2.17)$$

де для кожного вектора $x \in X^{imp}$ та кожного індексу j виконується наступна операція:

$$\Psi(x)_j = \frac{x_j - \mu_j^{imp}}{\sigma_j^{imp}} \quad (2.18)$$

Після застосування оператора Ψ до всієї імпутованої вибірки отримуємо масштабовану множину:

$$X^{scaled} = \{\Psi(x) \mid x \in X^{imp}\} \quad (2.19)$$

Таким чином, операція стандартизації переводить простір імпутованих ознак у нормалізований евклідов простір із нульовим математичним сподіванням та одиничною дисперсією для кожної координати, що забезпечує коректну та стабільну роботу моделей машинного та глибокого навчання.

Після виконання всіх етапів попередньої обробки даних, зокрема імпутації пропущених значень та стандартизації простору ознак, формується узгоджена та нормалізована вибірка [83]. Водночас у задачах виявлення аномалій спостерігається суттєвий дисбаланс між кількістю прикладів нормальної поведінки та аномальних сценаріїв. З огляду на те, що в реальних умовах спектр атак постійно розширюється, а нові вектори загроз можуть бути відсутні у навчальних даних, доцільним є перехід до підходу виявлення аномалій.

У межах такого підходу для навчання моделі використовується виключно підмножина спостережень, що відповідають нормальній роботі системи. Тоді як усі відхилення від вивченої нормальної поведінки розглядаються як потенційні ознаки кібернетичного впливу на систему.

2.3 Результати експериментального дослідження методу виявлення аномалій в інфраструктурі Kubernetes

Експериментальне дослідження було проведено з метою оцінки ефективності запропонованого методу виявлення аномалій у контейнеризованому середовищі Kubernetes на основі LSTM Autoencoder. Перед безпосереднім проектуванням та навчанням інтелектуальної моделі детекції було проведено комплексний статистичний аналіз вхідних даних, отриманих з журналів моніторингу мікросервісного середовища.

Основним завданням цього етапу є валідація структури сформованої вибірки та оцінка ступеня дисбалансу класів, оскільки ці параметри критично впливають на вибір архітектури нейромережі, функцій втрат та подальших метрик оцінки ефективності. На рисунку 2.4 представлено кількісний та відсотковий розподіл цільових класів «Норма» та «Атака» для експериментальних наборів даних BOA та DVWA.

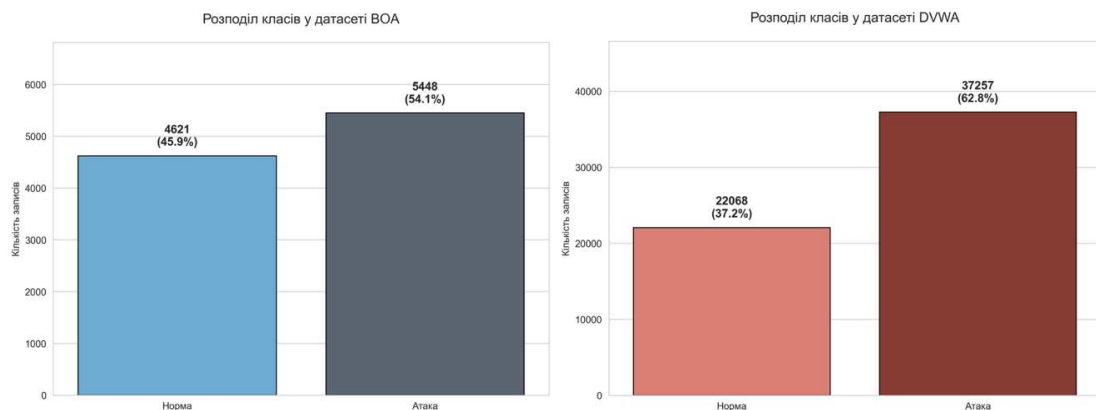


Рисунок 2.4 – Розподіл класів у датасеті BOA та DVWA

У ході препроцесингу було проведено процедуру бінаризації міток цільової змінної: всі записи, що за первинною розміткою класифікувалися як специфічні типи вразливостей (зокрема клас 2 та інші підтипи шкідливої активності), були агреговані у спільну категорію «Атака», тоді як легітимні транзакції позначені як «Норма». Аналіз набору даних BOA (рис. 2.4)

демонструє відносно збалансований розподіл, де нормальний трафік складає 45.9% (4621 запис), а аномальна активність – 54.1% (5448 записів). Така структура є оптимальною для базового навчання моделі розрізненню патернів. Натомість набір DVWA (рис. 2.5) демонструє зміщення в бік шкідливого трафіку, де частка атак досягає 62.8% (37257 записів) проти 37.2% (22068 записів) легітимних операцій. Вибір на користь вибірок з високою концентрацією аномалій є методологічно обґрунтованим рішенням для даного дослідження. У реальних інформаційних системах частка атак зазвичай становить менше 1%, проте навчання моделі на таких «природних» даних часто призводить до «парадоксу точності», коли алгоритм ігнорує рідкісні події та максимізує точність шляхом постійного прогнозування мажоритарного класу. Використання акцентованих навчальних вибірок, де аномалії складають 50–60%, змушує нейромережу глибоко вивчати внутрішні ознаки деструктивного трафіку та підвищує її узагальнюючу здатність при детекції нових типів загроз. Агрегація різних типів атак у єдиний домен аномалій дозволяє реалізувати систему у вигляді ефективного бінарного детектора аномалій, що забезпечує стабільність навчання та високу швидкість прийняття рішень у високонавантажених мікросервісних архітектурах.

Для обґрунтування вибору вхідних параметрів моделі та зниження ризику було проведено кількісну оцінку інформативності ознак на основі алгоритму Random Forest. На рисунку 2.5 представлено 15 найбільш значущих параметрів, впорядкованих за показником зменшення Gini Importance. Отримані результати підтверджують гіпотезу про високу кореляцію між часовими характеристиками пакетів та наявністю аномальної активності. Зокрема, ознаки, що описують інтервали між надходженнями пакетів (`packets_IAT_mean`, `packets_IAT_std`), посідають провідні позиції, що свідчить про їхню критичну роль у розпізнаванні автоматизованих мережових атак.

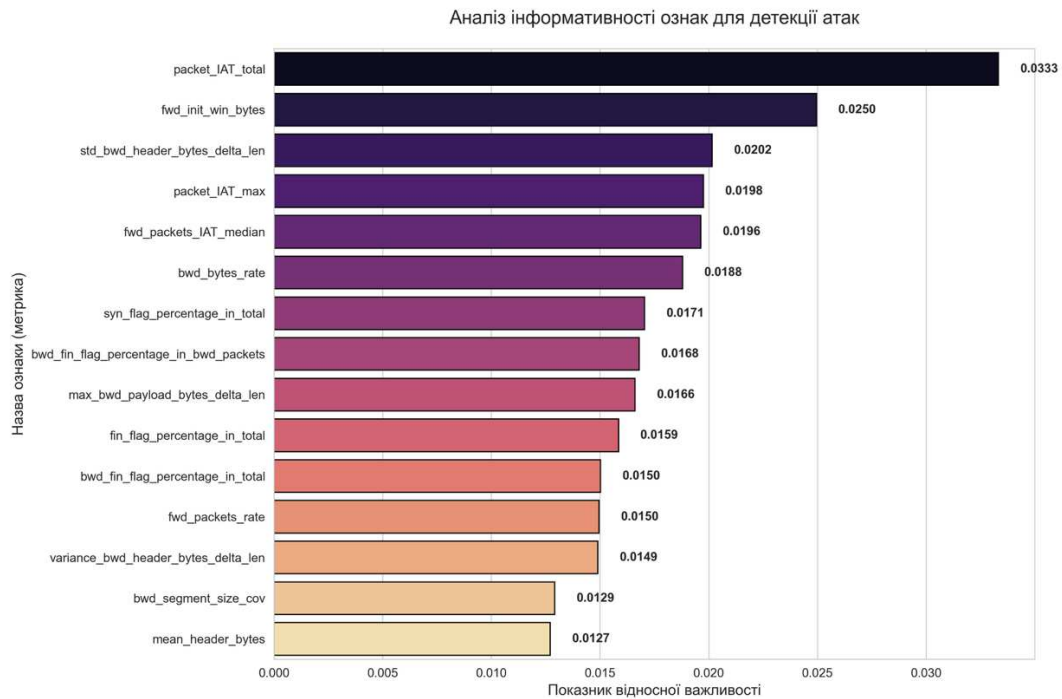


Рисунок 2.5 – Аналіз інформативності ознак для детекції атак

Також високу важливість демонструють метрики інтенсивності трафіку (`bytes_rate`, `packets_count`) та специфічні прапорці TCP-протоколу. Важливо зауважити, що до переліку найбільш інформативних ознак увійшли показники споживання ресурсів контейнерами (`container_cpu_usage_seconds_rate`), що підтверджує доцільність використання мультимодального підходу (поєднання мережесих та системних метрик) для моніторингу безпеки мікросервісних архітектур. Виділений набір домінуючих ознак дозволяє скоротити простір пошуку для нейронної мережі, забезпечуючи швидшу конвергенцію при збереженні високої точності детекції.

Враховуючи використання рекурентної архітектури LSTM як основного інструменту детекції, критично важливим етапом препроцесингу є приведення всіх вхідних ознак до єдиного діапазону значень. На рисунку 2.6 продемонстровано ефект застосування процедури стандартизації до обраних ознак з різними фізичними одиницями вимірювання. Після застосування Z-перетворення, всі ознаки були центровані навколо нуля з одиничним середньоквадратичним відхиленням.

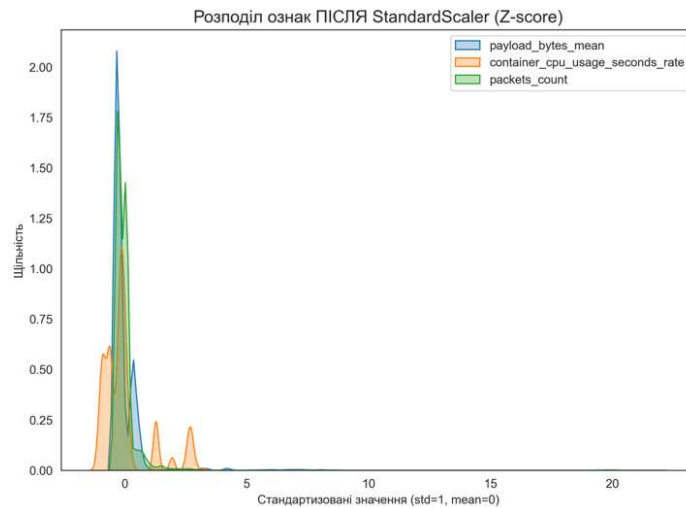


Рисунок 2.6 – Розподіл ознак

Таке перетворення дозволяє стабілізувати процес навчання, забезпечує коректну роботу механізмів довгострокової пам'яті у структурі LSTM та запобігає проблемі зникаючого або вибухового градієнта. Уніфікація масштабів забезпечує рівноцінний внесок кожної метрики у фінальний прогноз моделі, що є необхідною умовою для виявлення складних кореляцій між мережевими аномаліями та станом мікросервісів.

Важливим етапом структурного аналізу сформованої вибірки стало побудування кореляційної матриці найбільш інформативних ознак. Цей аналіз дозволив ідентифікувати внутрішні залежності між параметрами мережевого трафіку та показниками стану мікросервісів.

На рисунку 2.7 продемонстровано результати розрахунку коефіцієнтів кореляції Пірсона. Зокрема, виявлено групи сильно корельованих ознак ($r > 0.9$), що описують інтенсивність потоків та статистику ТСП-прапорців. Наявність такої високої мультиколінеарності вказує на надлишковість окремих метрик, що створює передумови для подальшої редукції простору ознак.

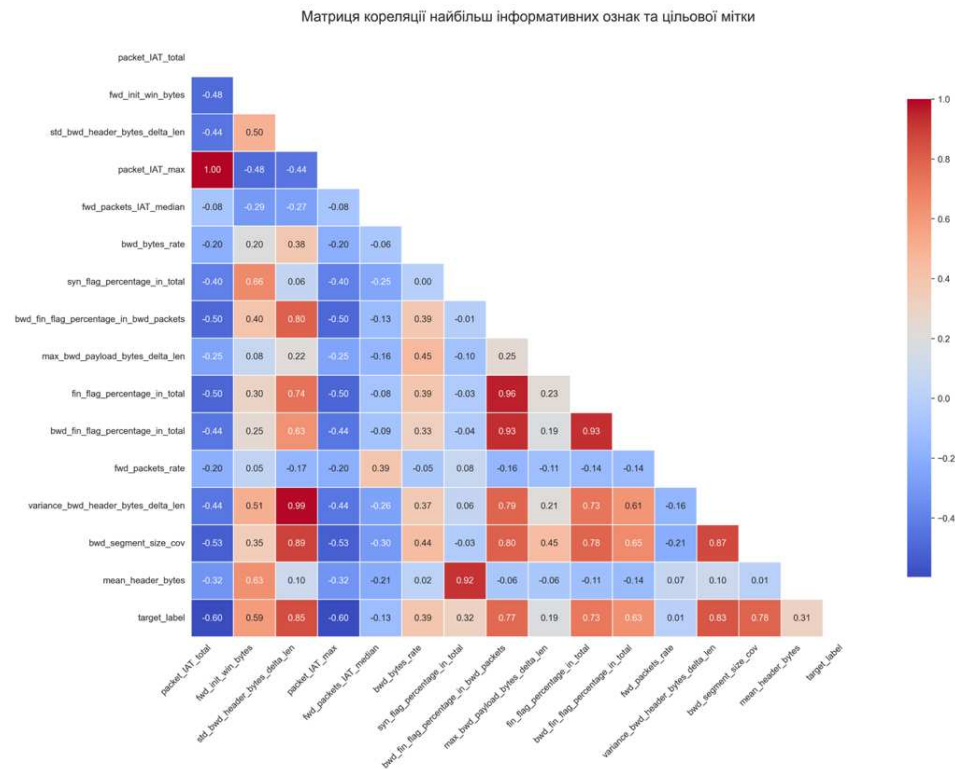


Рисунок 2.7 – Матриця кореляції найбільш інформативних ознак та цільової мітки

Водночас, виявлена стійка кореляція між цільовою змінною `target_label` та часовими характеристиками пакетів підтверджує їхню значущість. Отримані результати кореляційного аналізу є базою для подальшого дослідження аномальних відхилень у просторі ознак та оцінки впливу викидів на загальну стабільність моделі.

Аналіз розподілу ознак за допомогою діаграм розмаху та «скрипкових» графіків (рис. 2.8) дозволив виявити специфічні закономірності поведінки системи в умовах кібератак. Найбільш виражена диференціація спостерігається у часових характеристиках (`packets_IAT_mean`), де клас «Атака» демонструє мінімальну варіативність та високу щільність у зоні низьких значень, що відповідає високоінтенсивній автоматизованій генерації запитів. Параметри споживання ресурсів контейнерами (`container_cpu_usage`) демонструють мультимодальний характер розподілу для штатного режиму роботи, що обумовлено динамічною природою мікросервісного середовища.

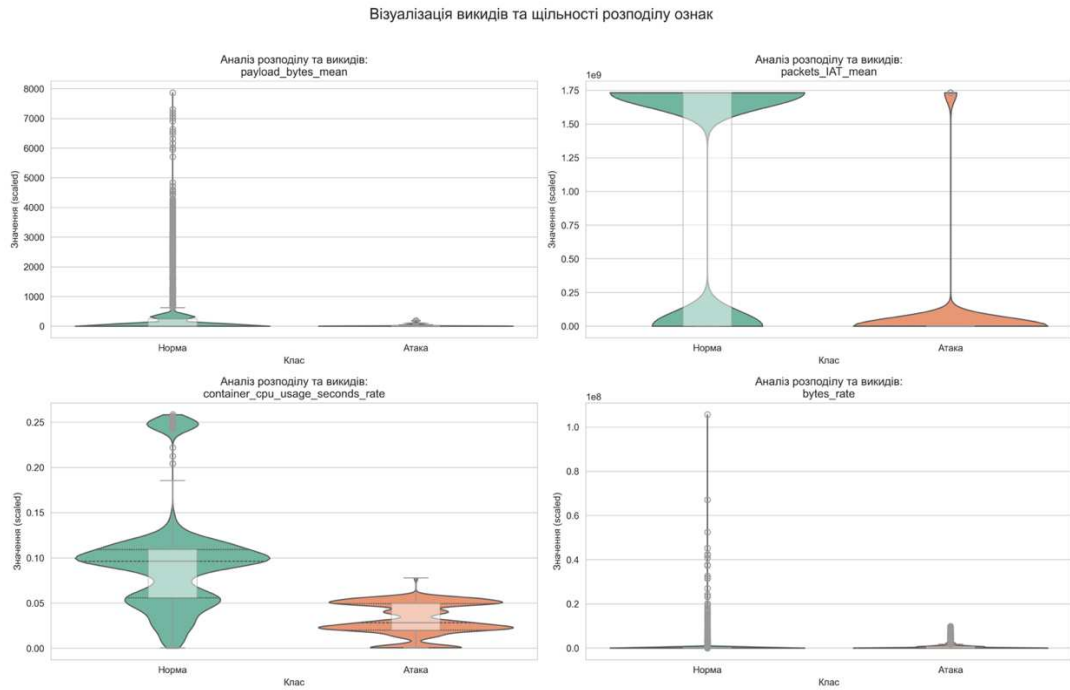


Рисунок 2.8 – Візуалізація викидів та щільності розподілу ознак

Виявлені численні викиди у метриках обсягу трафіку підтверджують складність вхідного сигналу та обґрунтовують доцільність застосування методів глибокого навчання, здатних до формування нелінійних роздільних поверхонь у просторі ознак зі значним рівнем шуму

Для перевірки гіпотези про часову залежність аномальних станів та обґрунтування доцільності застосування рекурентних архітектур нейронних мереж було проведено порівняльний аналіз динаміки ключових ознак. На рисунку 2.9 представлено часові діаграми мережевої інтенсивності (`bytes_rate`) та ресурсного стану обчислювального вузла (`container_cpu_usage_seconds_rate`) у співставленні з періодами верифікованих атак (Ground Truth). Візуалізація часових послідовностей дозволяє виділити наступні фундаментальні закономірності функціонування мікросервісного середовища в умовах кібервпливів:

Еволюційний характер аномалій, що ілюструється графіком міток цільового класу на нижній панелі, свідчить про те, що атака в хмарній інфраструктурі не є поодиноким ізольованим подієм, а являє собою тривалий у часі процес із вираженими фазами активності. Це підтверджує необхідність

того, щоб для ефективної детекції модель оперувала не миттєвими значеннями метрик, а їхніми послідовностями (sequences), що робить використання шарів рекурентної пам'яті LSTM критично необхідним для збереження контексту попередніх станів системи.

Паралельно з цим на рис. 2.9 спостерігається чітка кореляція між сплесками вхідного трафіку та зміною профілю споживання процесорного часу, що вказує на деградацію продуктивності під впливом шкідливого трафіку. Зокрема, під час переходу системи у стан «Атака» на середній панелі зафіксовано специфічну аномалію: замість динамічного коливання CPU, характерного для штатного режиму, виникає різкий спад активності до значень, близьких до нуля. Дане явище ілюструє ефект відмови у обслуговуванні, коли через критичне перевантаження мережевого стеку мікросервіс втрачає здатність виконувати корисні обчислення, що виступає ключовим діагностичним маркером для нейромережевої моделі.

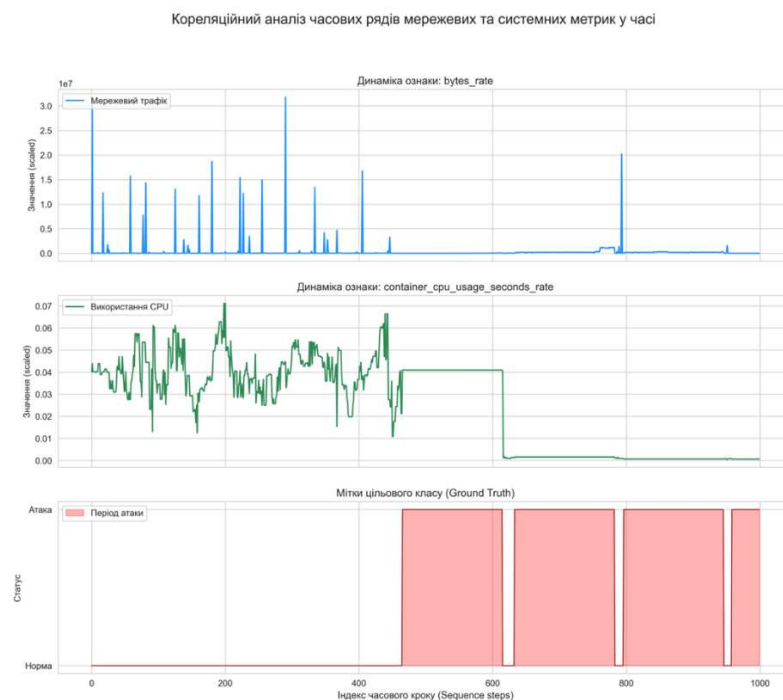


Рисунок 2.9 – Кореляційний аналіз часових рядів

Додатково аналіз часових рядів демонструє певну затримку між

початком інтенсивного мережевого впливу та повною деградацією ресурсних метрик, що свідчить про інерційність системної відповіді. Наявність такої інерції вказує на складну внутрішню динаміку контейнеризованих додатків та обґрунтовує використання рекурентних моделей, здатних ідентифікувати аномальні тренди на ранніх етапах розвитку атаки, враховуючи накопичувальний ефект змін у системі та її реакцію на зовнішні подразники.

Таким чином, результати аналізу часової динаміки засвідчують, що дані мають виражену послідовну структуру, а стан системи в кожний момент часу t суттєво залежить від попередніх спостережень у вікні $t-n$. Отримані висновки є прямим підтвердженням доцільності розробки нейромережевого детектора на основі архітектури LSTM, здатної до виявлення нелінійних закономірностей у багатовимірних часових рядах мікросервісного середовища.

Для інтегральної оцінки роздільності сформованого простору ознак та візуалізації структури даних було застосовано метод аналізу головних компонент. Даний підхід дозволив здійснити зниження розмірності з понад 200 первинних метрик до двох найбільш інформативних головних компонент (PC1 та PC2), які сукупно описують 35.76% загальної дисперсії вибірки. Значення поясненої дисперсії для першої компоненти (23.6%) та другої (12.2%) вказують на високу складність та багатовимірність вихідних даних, де жодна окрема ознака не є домінуючою.

Отримана проєкція демонструє виражену кластерну структуру: основна маса спостережень класу «Норма» формує витягнутий векторний напрямок, що відображає варіативність штатної роботи мікросервісів. Об'єкти класу «Атака» (виділені червоним) локалізовані у верхній лівій частині простору ознак, утворюючи компакту групу, що частково межує з початковими станами нормальної активності, рис 2.10. Така топологія підтверджує, що аномальна активність має специфічний статистичний профіль, відмінний від типової поведінки системи.

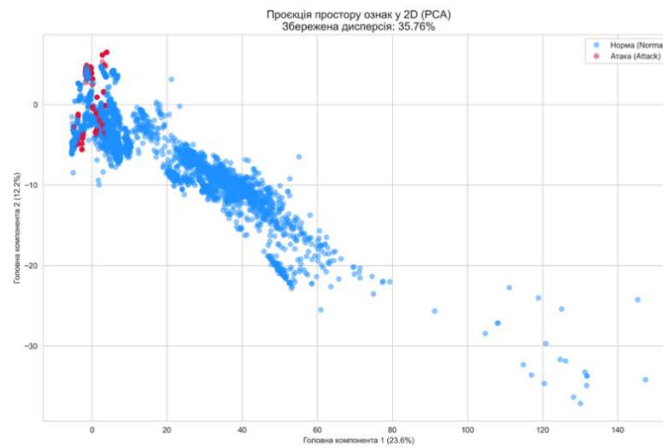


Рисунок 2.10 – Проекція простору ознак у 2D

Наявність певної зони накладання класів та нелінійна форма розподілу нормального трафіку математично обґрунтовують доцільність переходу від лінійних методів аналізу до рекурентних нейронних мереж. Саме здатність LSTM формувати складні роздільні поверхні у повному просторі ознак дозволяє досягти високої точності детекції в умовах, де лінійна проєкція фіксує перехідні стани між нормою та атакою.

Для кількісної оцінки якості сформованої вибірки було розраховано співвідношення наявних та відсутніх значень у всьому масиві ознак. Як ілюструє кругова діаграма (рис. 2.11), переважна більшість даних (98.29%) є валідними та готовими до обробки без додаткової інтерполяції. Частка пропущених значень складає лише 1.71%, що є допустимим показником для складних систем моніторингу хмарних інфраструктур.

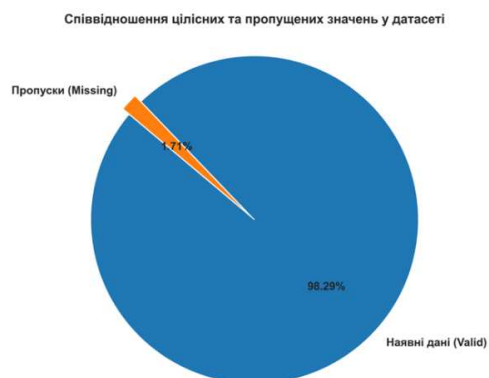


Рисунок 2.11 – Співвідношення цілісних та пропущених значень у датасеті

Такий розподіл свідчить про високу репрезентативність даних та мінімальний вплив «шуму», що виникає при заповненні пропусків. Незначний відсоток відсутніх значень гарантує, що при використанні методів відновлення даних (таких як Forward Fill або лінійна інтерполяція) статистичні характеристики часових рядів залишаться незмінними. Це забезпечує високу якість навчання моделі LSTM, оскільки нейромережа отримує фактично неперервні послідовності реальних спостережень, що є критичним для коректного виявлення довгострокових залежностей під час атак.

Процес навчання розробленої архітектури базувався на принципах *unsupervised learning*. Навчання моделі здійснювалося виключно на підмножині даних, що відповідають нормальній роботі системи, тоді як оцінка якості виконувалася на повній вибірці, яка містила як нормальні, так і атакувальні сценарії.

Після завершення етапу навчання для кожної часової послідовності було обчислено помилку реконструкції, на основі якої здійснювалася класифікація за допомогою порогового правила. Значення порогу аномальності визначалося шляхом максимізації F1-міри на валідаційній вибірці. У результаті оптимальний поріг було встановлено на рівні $\theta = 0,47$, що забезпечило найкращий баланс між точністю та повнотою виявлення аномальної активності.

Отримані результати свідчать про високу ефективність запропонованого підходу для виявлення аномалій у Kubernetes-середовищі. Для класу нормальної активності досягнуто значень Precision = 0,95, Recall = 0,97 та F1-score = 0,96, що підтверджує здатність моделі коректно ідентифікувати штатний режим функціонування кластера. Для класу атакувальної активності отримано Precision = 0,97, Recall = 0,958 та F1-score = 0,964, що свідчить про високу якість виявлення аномальних подій та низьку кількість як пропущених атак, так і хибнопозитивних спрацювань.

Загальна точність класифікації становила 96,1%, а зважене середнє

значення F1-міри також досягло 0,961. Значення Macro F1-score на рівні 0,962 підтверджує збалансовану якість класифікації для обох класів незалежно від їх розміру. Отримані результати демонструють, що запропонований підхід забезпечує ефективне виявлення як відомих, так і раніше невідомих аномалій у контейнеризованому середовищі, зберігаючи високу точність роботи навіть за наявності значного обсягу телеметричних даних.

Основні показники якості роботи моделі наведено в таблиці 2.1.

Таблиця 2.1 – Показники якості виявлення аномалій за допомогою LSTM Autoencoder

Клас	Precision	Recall	F1-score	Кількість зразків
Нормальна активність (0)	0.95	0.97	0.96	26 679
Атакувальна активність (1)	0.97	0.958	0.964	42 705
Macro average	0.96	0.964	0.962	69 384
Weighted average	0.962	0.961	0.961	69 384
Загальна точність (Accuracy)	-	-	0.961	69 384

Отримані експериментальні результати підтверджують, що використання LSTM Autoencoder дозволяє ефективно моделювати часову динаміку нормальної поведінки Kubernetes-кластера та виявляти аномалії без необхідності попереднього знання всіх можливих типів атак. Це робить запропонований метод придатним для практичного застосування в умовах постійної еволюції загроз та обмеженої доступності розмічених атакувальних

даних.

Висновки до розділу 2

У другому розділі дисертаційної роботи розроблено та експериментально обґрунтовано підхід до виявлення аномальної та потенційно шкідливої активності у Kubernetes-кластерах на основі методів глибокого навчання. Запропоноване рішення ґрунтується на використанні архітектури LSTM Autoencoder, яка дозволяє моделювати складну часову динаміку нормальної поведінки контейнеризованого середовища та ідентифікувати аномалію за рівнем відхилення від еталонного профілю без необхідності попереднього знання повної множини сигнатур атак.

У ході дослідження формалізовано багатовимірний простір ознак, що інтегрує мережеві характеристики, телеметрію ресурсів контейнерів та системні показники стану Kubernetes-кластера. Детально описано та реалізовано технологічний стек попередньої обробки даних, що включає імпутацію пропущених значень стандартизацію та формування часових послідовностей, що є критично важливим для збереження контекстуальних залежностей у рекурентних шарах мережі. Застосування методу PCA дозволило візуалізувати структуру даних та підтвердити, що попри високу розмірність простору (де перші дві компоненти описують 35.76% дисперсії), аномальні стани формують виражені кластерні структури, що обґрунтовує можливість їх ефективної сепарації нелінійними методами.

Особливістю запропонованого підходу є навчання моделі виключно на підмножині даних нормальної роботи системи, що дозволило повністю нівелювати проблему дисбалансу класів та забезпечити високу адаптивність моделі до виявлення раніше невідомих або модифікованих типів атак. Експериментальні результати підтвердили ефективність обраної стратегії: після оптимізації порогу аномальності ($\theta=0.47$) за критерієм максимізації F1-міри, модель продемонструвала високу здатність до детекції з показником

повноти виявлення атак $\text{recall} = 0.95$ та значенням $\text{F1-score} = 0.83$. Отримані результати класифікації свідчать про доцільність використання часових залежностей телеметричних даних для побудови інтелектуальних систем захисту в динамічних та слабо розмічених хмарних середовищах.

Сформовані результати другого розділу стали практичним підтвердженням можливості впровадження методів глибокого навчання для моніторингу безпеки оркестрованих інфраструктур, що відкриває перспективи для подальшого вдосконалення архітектур на основі варіаційних автоенкодерів та їх інтеграції з низькорівневими засобами runtime-захисту на базі технології eBPF.

3 ВИЯВЛЕННЯ АНОМАЛІЙ В API ХМАРНИХ СЕРВІСІВ

3.1 Метод виявлення аномалій поведінки користувачів на основі API хмарних сервісів

Метод використовується виявлення аномалій у поведінці користувачів вебсайту шляхом застосування автоенкодера LSTM для аналізу послідовностей вебсесій та виявлення нетипових сценаріїв взаємодії користувача з вебсайтом.

На цьому етапі дані збираються з веблогів (проксі-логів), які містять записи виду

$$\{session_id, t, domain\}, \quad (3.1)$$

де *session_id* – ідентифікатор сеансу, *t* – позначка часу запиту, а *domain* – доменне ім'я відвіданого вебресурсу. Ці дані відображають послідовність дій користувача у вебсередовищі.

Етап 2: Попередня обробка даних. На цьому етапі виконується очищення даних, формування вебсеансу та вилучення часових ознак. Вебсеанс визначається як упорядкована за часом послідовність запитів від одного користувача. Для кожного сеансу обчислюються додаткові часові характеристики, включаючи день тижня, годину початку та тривалість сеансу. Результатом цього етапу є набір структурованих сеансів:

$$S = \{S_1, S_2, \dots, S_n\} \quad (3.2)$$

Етап 3: Вилучення ознак. Кожна сесія S_i представлена вектором поведінкових ознак.

$$X_i = (x_i^{(1)}, x_i^{(2)}, \dots, x_i^{(m)}), \quad (3.3)$$

Ці ознаки враховують тривалість сеансу, кількість сайтів, день тижня, години активності, показники характерних сайтів користувача, а також послідовність відвідуваних доменів $site1 \dots, site10$. Таким чином формується вхідний простір ознак для моделі.

Етап 4: Часовий поділ набору даних. Щоб запобігти витоку даних, набір даних розділяється часовим порогом T на навчальну та тестову частини:

$$X_{\text{train}} = \{x_i \mid t_i < T\}, X = \{x_i \mid t_i \geq T\}, \quad (3.4)$$

Навчальний набір містить лише історичні дані, тоді як тестовий набір використовується для оцінки методу.

Етап 5: Навчання LSTM-автоенкодера. На цьому етапі LSTM-автоенкодер навчається виключно на сеансах конкретного користувача, які розглядаються як приклади нормальної поведінки. Автоенкодер виконує відображення:

$$f_{\theta} : X \rightarrow \hat{X}, \quad (3.5)$$

де X представляє вхідні поведінкові ознаки, $\hat{X}$ є їх реконструкцією, і, θ є параметрами моделі. Навчання полягає в мінімізації функції втрат, наприклад, середньоквадратичної помилки (MSE):

$$L = E [\|X - \hat{X}\|^2]. \quad (3.6)$$

Етап 6: Побудова профілю нормальної поведінки користувача. В результаті навчання формується профіль нормальної поведінки користувача, що відображає характерні часові та структурні закономірності його активності. Цей профіль визначається параметрами навченого автокодера та

розподілом помилок реконструкції на навчальних даних.

Етап 7: Обчислення помилки реконструкції. Для кожного нового сеансу обчислюється помилка реконструкції:

$$\varepsilon_i = \|X_i - \hat{X}_i\|, \quad (3.7)$$

що служить кількісною мірою відхилення від нормального профілю поведінки користувача.

Етап 8: Порівняння порогів та прийняття рішень. На заключному етапі помилка реконструкції порівнюється з пороговим значенням θ , визначеним на основі навчальних даних:

$$\varepsilon_i = \begin{cases} \varepsilon_i \leq \theta, & \text{normal behavior} \\ \varepsilon_i > \theta, & \text{anomalous behavior} \end{cases} \quad (3.8)$$

Таким чином, сеанси, які не відповідають звичайному профілю поведінки користувача, ідентифікуються як аномальні, структурна схема методу представлена на рисунку 3.1.

3.2 Опис даних для проведення експерименту

Для проведення експеримента використовується набір даних про вебактивність користувачів із проксі-серверів Університету Блеза Паскаля [84]. Дані складаються з журналів HTTP-запитів, що містять інформацію про поведінку користувачів під час взаємодії з вебресурсами. Кожен запис у журналі містить ідентифікатор користувача, позначку часу початку сеансу та ім'я домену відвідуваного ресурсу. Початковий набір даних містив приблизно 17×10^6 записів, згенерованих понад 3000 користувачами. Первинний лог-файл представлений у форматі CLF, де кожен атомарний запис відповідає окремому запиту до вебресурсу. Для забезпечення конфіденційності всі ідентифікатори користувачів (`user_id`) пройшли

процедуру анонізації. Структурно кожен запис містить наступний набір атрибутів:

- timestamp – часова мітка з мікросекундною точністю, що дозволяє аналізувати інтенсивність запитів;
- domain_name – цільовий вузол звернення;
- user_agent – технічні характеристики клієнтського ПЗ;
- http_status – код відповіді сервера, використаний як індикатор валідності запиту.

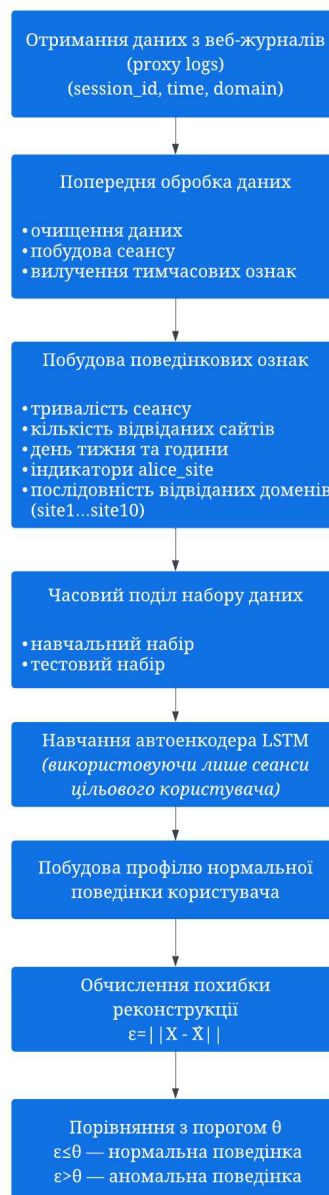


Рисунок 3.1 – Структурна схема методу виявлення аномалій в API хмарних сервісів

Для покращення якості даних та видалення шуму було проведено попередню фільтрацію. На першому етапі було застосовано фільтри чорного списку для видалення рекламних та сервісних доменів. На другому етапі фільтрацію було проведено на основі кодів статусу HTTP-запиту, що дозволило виключити недоступні або недійсні домени. Після цих процедур набір даних було скорочено приблизно до 4×10^6 записів.

Очищені дані були згруповані за користувачами та сегментовані на окремі вебсесії. Для кожної сесії було створено структурований опис, що включає послідовність відвідуваних доменів (до 10 сайтів на сеанс), час початку сеансу, тривалість, день тижня, годину початку та інші часові характеристики. Подальший аналіз зосередився на 150 користувачах з найбільшою кількістю запитів, що забезпечило достатню кількість даних про поведінку для моделювання.

Для ідентифікації поведінки та виявлення аномалій усіх користувачів було розділено на два класи: цільовий користувач (Аліса) та інші користувачі. Була введена бінарна змінна *target*, де 1 відповідає Алісі, а 0 – іншим користувачам. Такий підхід дозволяє проводити порівняльний аналіз моделей поведінки та побудову моделі нормальної поведінки [85].

Для підготовки даних до моделювання було створено додаткові функції, включаючи індикатор наявності доменів, характерних для Аліси, протягом сеансу. Набір даних був часово розділений на навчальні та тестові підмножини на основі фіксованої дати, що запобігає витoku інформації та дозволяє точно оцінити продуктивність моделі в умовах, близьких до реальних сценаріїв.

Підготовлений набір даних забезпечує основу для застосування методів аналізу послідовностей та побудови моделі виявлення аномалій для поведінки користувачів за допомогою автоенкодера LSTM.

На рисунку 3.2 показано розподіл вебсесій за днями тижня для аналізованого набору даних. Результати показують, що найвища активність користувачів спостерігається в будні дні, з піком у середині тижня – середу. Велика кількість сеансів також спостерігається у вівторок та четвер, що

відображає регулярну та стабільну активність користувачів протягом робочого тижня, і навпаки, у вихідні дні спостерігається помітне зменшення кількості сеансів, особливо в неділю, ймовірно, через зміни в розпорядку дня користувачів та зменшення використання вебресурсів у неробочий час.

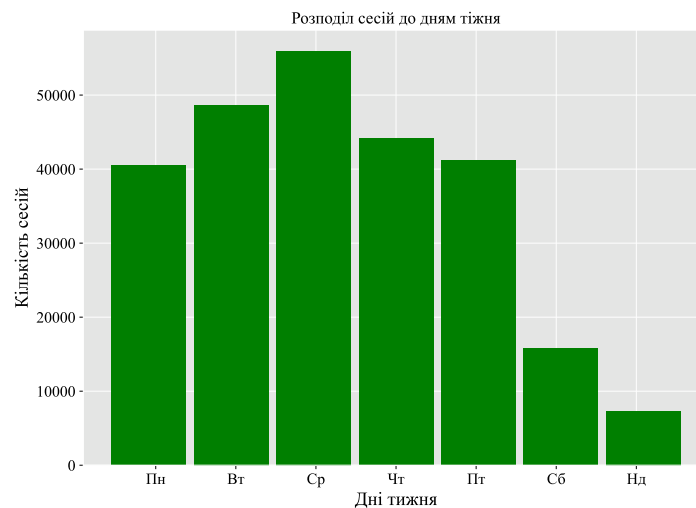


Рисунок 3.2 – Розподіл вебсесій за днями тижня

Ця чітко визначена тижнева циклічна модель є важливою характеристикою нормальної активності користувачів і її слід враховувати під час побудови моделей виявлення аномалій. Відхилення від типового розподілу активності за днями тижня можуть служити показниками нетипової або потенційно аномальної поведінки користувачів.

На рисунку 3.3 показано розподіл вебсесій за днями тижня для цільового користувача Аліси. Аналіз показує, що поведінка Аліси демонструє виражену нерівномірність протягом тижня. Найбільша кількість сесій припадає на понеділок, що свідчить про високу активність користувачів на початку робочого тижня. Значна активність також спостерігається у вівторок та четвер, тоді як у середу спостерігається помітно менша кількість сесій порівняно з іншими будніми днями.

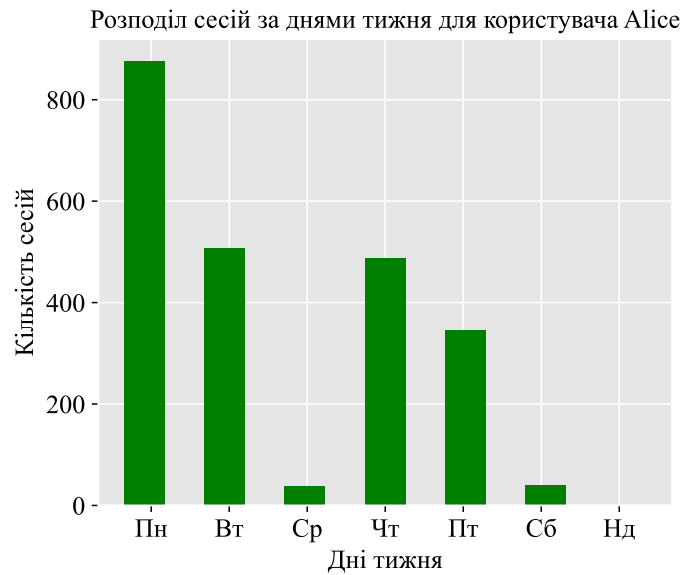


Рисунок 3.3 – Розподіл вебсесій за днями тижня для користувача Аліси

У п'ятницю активність знижується, а протягом вихідних (суботи та неділі) кількість сесій мінімальна. Такий розподіл відображає чітку залежність поведінки користувачів від робочого графіка та дозволяє охарактеризувати типову тижневу модель активності Аліси. Спостережувані моделі можуть бути використані як важливий показник нормальної поведінки при побудові моделей виявлення аномалій, оскільки значні відхилення від цього розподілу можуть свідчити про атипову або потенційно аномальну активність.

На рисунку 3.4 показано розподіл вебсесій за днями тижня для інших користувачів. Результати демонструють чітко визначену тижневу періодичність у поведінці, характерну для більшості користувачів. Пікова активність припадає на середу робочого тижня, з найбільшою кількістю сесій у середу, тоді як у понеділок та вівторок також спостерігається висока кількість сесій.

У другій половині тижня, починаючи з четверга, активність поступово знижується, а протягом вихідних (суботи та неділі) спостерігається різке зниження кількості сесій. Цей розподіл відображає типовий графік робочого тижня та являє собою узагальнену закономірність нормальної поведінки для групи користувачів. Порівняння цього розподілу з відповідною

закономірністю для користувача Аліси дозволяє ідентифікувати індивідуальні поведінкові характеристики та може служити додатковим критерієм для виявлення аномалій у рамках поведінкового аналізу.

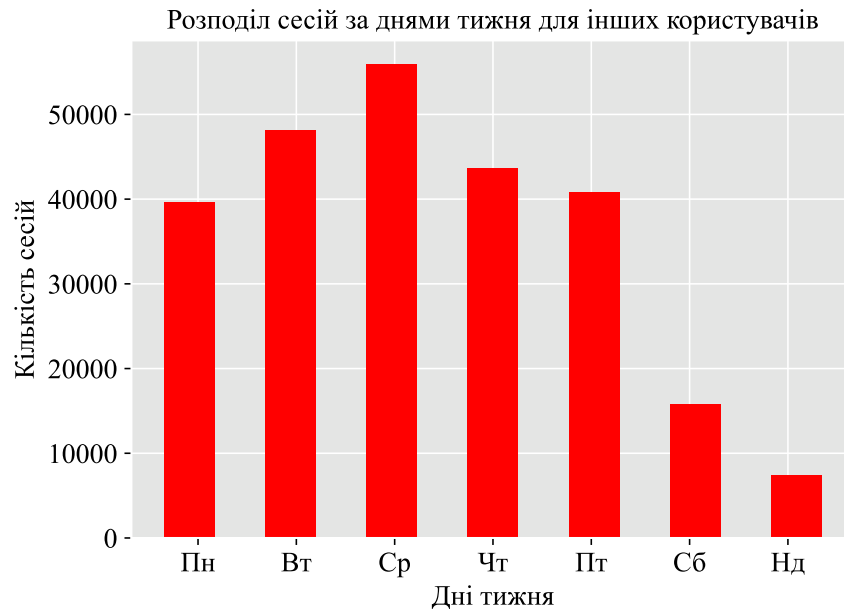


Рисунок 3.4 – Розподіл вебсесій за днями тижня для інших користувачів

На рисунку 3.5 показано розподіл вебсесій за годинами доби для користувача Аліси. Результати вказують на чітко визначену часову модель активності, характерну для цього користувача. Більшість сесій зосереджені в денний та ранній післяобідній періоди, з піком активності між 16:00 та 17:00. Додаткова концентрація сесій спостерігається пізно вранці, приблизно з 12:00 до 13:00.

Активність мінімальна або майже відсутня вранці (до 9:00) та ввечері після 18:00. Такий розподіл відображає послідовний щоденний режим використання вебресурсів і може бути інтерпретований як типовий робочий графік користувача. Виявлені часові моделі є важливою ознакою нормальної поведінки Аліси та можуть бути використані для побудови моделей виявлення аномалій, оскільки відхилення від цього графіка можуть свідчити про атипову або потенційно аномальну активність.

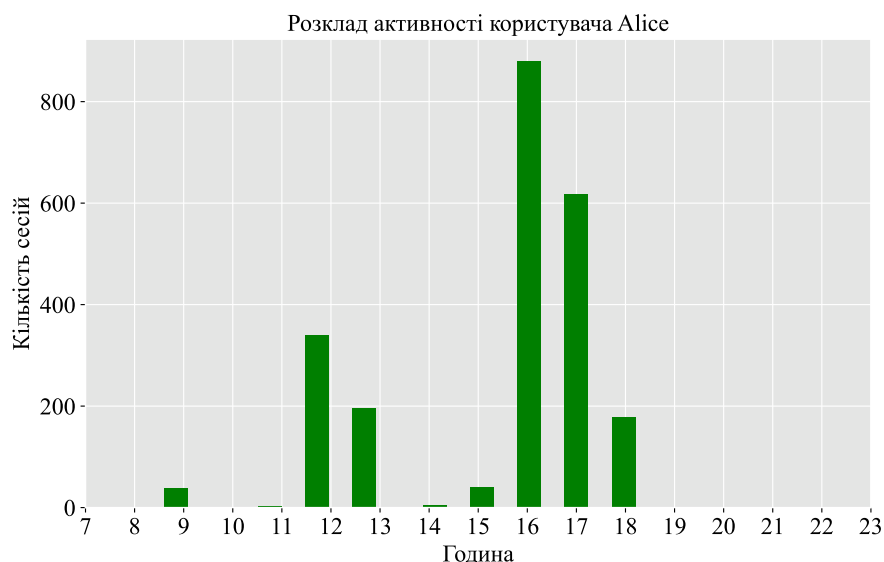


Рисунок 3.5 – Розподіл вебсесій за годинами доби для користувача Аліси

На рисунку 3.6 показано розподіл вебсесій за годинами доби для інших користувачів (за винятком Аліси). Результати демонструють більш рівномірний та часово розширений профіль активності порівняно з цільовим користувачем. Найвища інтенсивність сесій спостерігається вранці та пізно вранці, з піком приблизно між 9:00 та 11:00. На початку дня (13:00–16:00) активність залишається відносно високою, поступово знижуючись у пізнішій частині дня.



Рисунок 3.6 – Розподіл вебсесій за годинами доби для інших користувачів

Починаючи з 17:00, кількість сесій значно падає, а у вечірні та нічні години (після 18:00) більшість користувачів демонструють мінімальну активність. Такий розподіл відображає узагальнену типову схему використання вебресурсів та відповідає стандартному графіку робочого дня. Порівняння цього профілю з часовим графіком активності Аліси дозволяє виявити індивідуальні поведінкові відмінності та може служити інформативною ознакою при побудові моделей для виявлення аномальної поведінки користувачів.

На рисунку 3.7 розподіл тривалості вебсесій для користувача Аліси представлено в логарифмічній шкалі. Використання логарифмічного перетворення зменшує асиметрію розподілу та візуально виділяє як короткі, так і довгі сесії. Результати показують, що переважна більшість сесій мають відносно коротку тривалість, з концентрацією в діапазоні від малих до середніх значень логарифмічної тривалості.

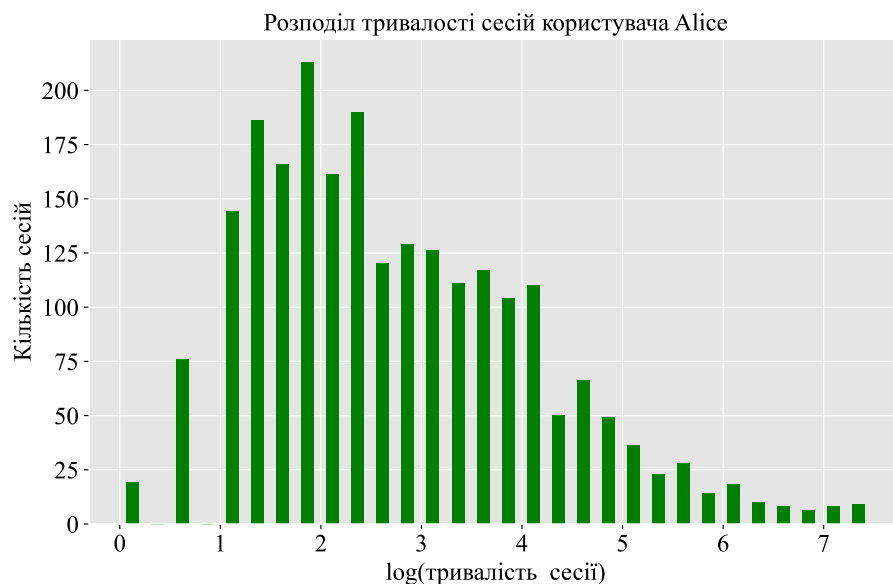


Рисунок 3.7 – Розподіл логарифмічної тривалості вебсеансів для користувача Аліси

Водночас, розподіл демонструє довгий правий хвіст, що відповідає рідкісним, але значно довшим сеансам. Така форма типова для даних про

поведінку користувачів і відображає стабільну закономірність нормальної активності Аліси. Спостережувані характеристики тривалості сеансу можуть служити інформативною ознакою при побудові моделей виявлення аномалій, оскільки значні відхилення від цього розподілу (наприклад, надзвичайно короткі або надмірно довгі сеанси) можуть свідчити про нетипову або аномальну поведінку.

На рисунку 3.8 показано розподіл логарифмічної тривалості вебсеансів для інших користувачів (за винятком Аліси). Як і у випадку з цільовим користувачем, було застосовано логарифмічне перетворення тривалості сеансів для точного представлення широкого діапазону значень та зменшення впливу випадкових тривалих сеансів. Отриманий розподіл помітно асиметричний, з концентрацією значень у діапазоні від малої до середньої логарифмічної тривалості, що відображає типову поведінку більшості користувачів.

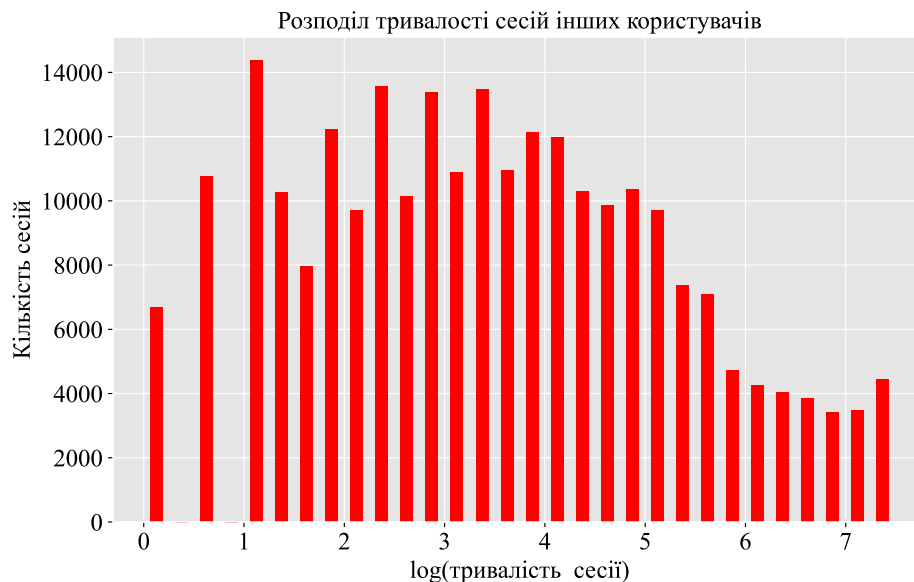


Рисунок 3.8 – Розподіл логарифмічної тривалості вебсеансів для інших користувачів

Водночас, інші користувачі демонструють ширший розподіл тривалості сеансів та більшу кількість сеансів із середніми та підвищеними значеннями

тривалості порівняно з Алісою. Наявність довгого правого хвоста вказує на значну кількість тривалих сеансів, що характерно для узагальненої групової поведінки[86]. Порівняльний аналіз розподілів тривалості сеансів для Аліси та інших користувачів дозволяє ідентифікувати індивідуальні поведінкові особливості та може служити інформативною характеристикою при побудові моделей виявлення аномалій, зокрема тих, що базуються на автоенкодері LSTM.

На наступному етапі дослідження дані були підготовлені для навчання та оцінки моделі. Щоб забезпечити коректність експерименту та запобігти витоку інформації між наборами даних, було застосовано часовий поділ на основі порогової дати. Усі сесії, що розпочалися до цієї дати, були призначені навчальному набору, тоді як сесії, що розпочалися в день порогу або пізніше, були призначені тестовому набору. Крім того, для кожної сесії було створено день ознаки, що представляє день року, що дозволяє аналізувати активність користувачів протягом календарних днів.

Для покращення розпізнавання поведінкових характеристик цільового користувача були створені індикаторні ознаки, пов'язані з «характерними сайтами» Аліси. Спочатку, використовуючи навчальний набір, було розраховано статистику відвідувань домену: для кожного доменного імені було визначено середнє значення `target` (1 для Аліси, 0 для інших користувачів) [87]. Домени, для яких ця частка перевищувала поріг 0,06, були включені до набору «Сайти Аліси». Для кожної сесії було згенеровано бінарну ознаку `alice_site`, яка приймала значення 1, якщо хоча б один із сайтів (`site1...site10`) належав до цього набору, і 0 в іншому випадку. Аналогічно, було додано розширені індикатори (`alice_site_2`, `alice_site_3`, `alice_site_4`) для фіксації наявності «типових» ресурсів у сеансі, тим самим підвищуючи інформативність опису поведінки.

Наступний крок включав фільтрацію сеансів на основі характеристик активності. Для кожного дня року було розраховано загальну кількість сеансів з `alice_site=1`, а «інформативні» дні (`good_days`) були визначені як ті, де ця сума

перевищувала поріг 70, тоді як «слабкі» дні (`bad_days`) були визначені як ті, чий значення не перевищували 70. Аналогічно, правила відбору були застосовані на основі годин доби: типові інтервали активності (`good_hours`) були визначені як 12, 13, 16 та 17, тоді як нетипові години (`bad_hours`) були 7, 8, 10, 11, 14 та 19–23. Рідкісні години (9, 15, 18) оброблялися окремо, застосовуючи додаткову фільтрацію на основі хвилин початку сеансу (розділення на допустимі та недопустимі хвилинні інтервали).

Узагальнені результати описаної процедури фільтрації та статистичне обґрунтування вибору порогів інтенсивності представлено на рисунку 3.9. Візуалізація дозволяє чітко ідентифікувати періоди стабільної активності цільового користувача та відсікти низькоінформативні інтервали, що забезпечує підвищення якості навчання моделі.

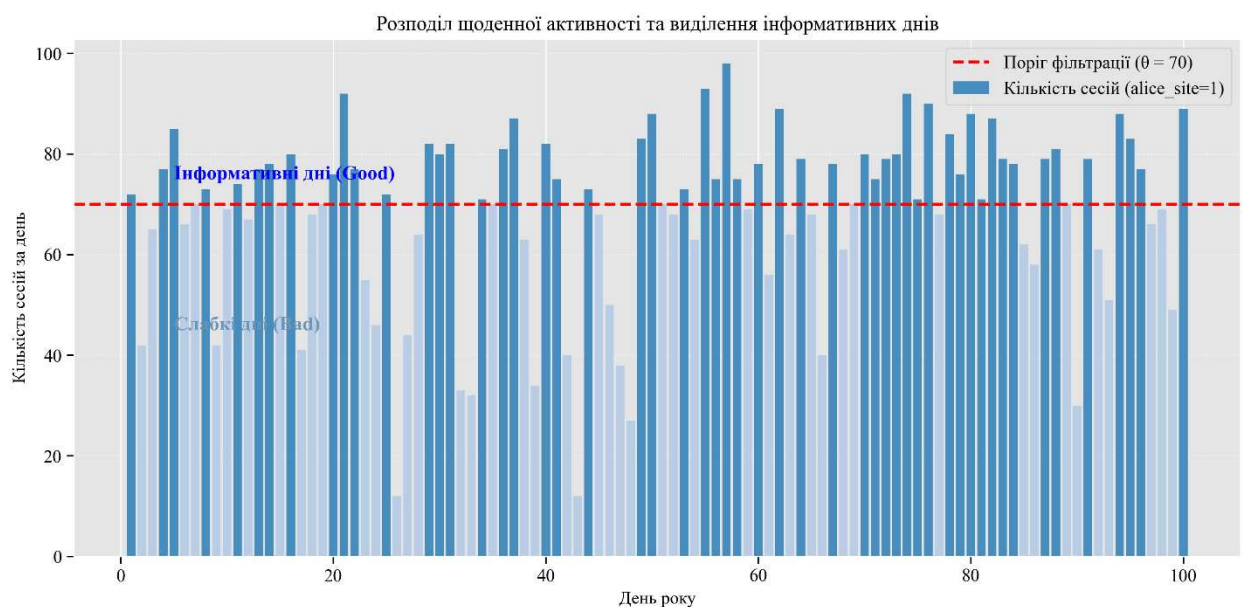


Рисунок 3.9 – Розподіл щоденної активності та виділення інформативних днів

В результаті проведених маніпуляцій були створені скорочені набори даних `train_df_short` та `test_df_short`, що містять переважно сеанси з характерними часовими та поведінковими ознаками, а також набори ідентифікаторів для «хороших» та «поганих» сеансів для подальшого аналізу.

Це дозволило сконцентрувати обчислювальні ресурси на найбільш репрезентативних паттернах поведінки Аліси, мінімізуючи вплив випадкових чинників на формування профілю нормальної активності

Для підготовки даних до машинного навчання було побудовано остаточний набір ознак [88]. Цей набір включає поведінкові індикатори (alice_site, alice_site_2, alice_site_3, alice_site_4), а також основні характеристики сеансу: number_of_sites, session_len, weekday, start_hour, end_hour, time_index та session_id. Категоріальні змінні (weekday, start_hour, end_hour, number_of_sites) були перетворені за допомогою одноразового кодування з виключенням першої категорії, щоб уникнути мультиколінеарності. Після кодування записи з відсутніми значеннями були видалені. Матриці ознак Xtrain та Xtest були створені шляхом виключення цільового стовпця, тоді як вектори міток ytrain та ytest були взяті з відповідного стовпця. Крім того, були видалені фіктивні стовпці, що відповідають нетиповим або неінформативним годинам, а також допоміжні змінні (time_index та alice_site_4). Усі етапи зазначеної трансформації даних, від вилучення первинних ознак до формування фінальних матриць, представлено на рисунку 3.10.



Рисунок 3.10 – Схема конвеєра підготовки та трансформації ознак

Після підготовки даних та побудови інформативного набору поведінкових ознак стає можливим перейти безпосередньо до побудови методу виявлення аномалій. Запропонований підхід базується на формуванні профілю нормальної поведінки конкретного користувача та подальшому виявленні відхилень від цього профілю. Цей метод не вимагає інформації про всі можливі типи аномалій та реалізує однокласову постановку задачі, що особливо актуально для аналітики поведінки користувачів.

Як інструмент для моделювання нормальної поведінки використовується рекурентний автокодер на основі довгої короткочасної пам'яті, здатний ефективно фіксувати часові та послідовні залежності в поведінці користувача. Модель навчається виключно на сесіях цільового користувача, які розглядаються як приклади нормальної поведінки. Після навчання відхилення між вхідними та реконструйованими ознаками сесії використовуються як критерій виявлення аномалій.

Таким чином, метод дозволяє автоматично ідентифікувати сесії, які не відповідають встановленому поведінковому профілю користувача, і може бути застосований до таких завдань, як поведінкова ідентифікація, контроль доступу та виявлення несанкціонованого використання облікового запису.

Для реалізації запропонованого методу виявлення аномалій було обрано програмний стек на базі мови програмування Python 3.10 та бібліотек глибокого навчання TensorFlow 2.x / Keras [89]. Вибір обумовлений наявністю оптимізованих реалізацій шарів LSTM та механізмів вкладень, що критично для обробки часових послідовностей.

Експериментальні дослідження проводилися на обчислювальному вузлі з наступними характеристиками:

- CPU: Intel Core i7 (12-th Gen);
- RAM: 32 GB DDR4;
- GPU: NVIDIA GeForce RTX 3060 (12GB VRAM) з підтримкою архітектури CUDA, що дозволило прискорити процес навчання моделі у 4,5 рази порівняно з CPU-обчисленнями.

Використання GPU-прискорення дозволило провести серію з 50 експериментів для підбору гіперпараметрів протягом прийнятного часового інтервалу (8 годин). Такий підхід забезпечує масштабованість розробленої інформаційної технології для роботи в умовах високонавантажених API-шлюзів хмарних сервісів

3.3 Результати експериментального дослідження метод виявлення аномалій в API хмарних сервісів

Важливим етапом оцінки ефективності розробленого методу виявлення аномалій в API хмарних сервісів є аналіз процесу навчання нейромережевої моделі, що дозволяє верифікувати правильність обраної архітектури та відсутність ефекту перенавчання.

На рисунку 3.11 представлено динаміку зміни функції втрат для навчальної та валідаційної вибірок протягом циклу навчання. Як свідчать результати, значення помилки на обох вибірках монотонно знижується, що вказує на стабільний процес оптимізації ваг. Відсутність значної розбіжності між тренувальною та валідаційною кривими після 20-ї епохи підтверджує успішну генералізацію ознак та адекватність обраної структури моделі для опису профілю нормальної поведінки системи.

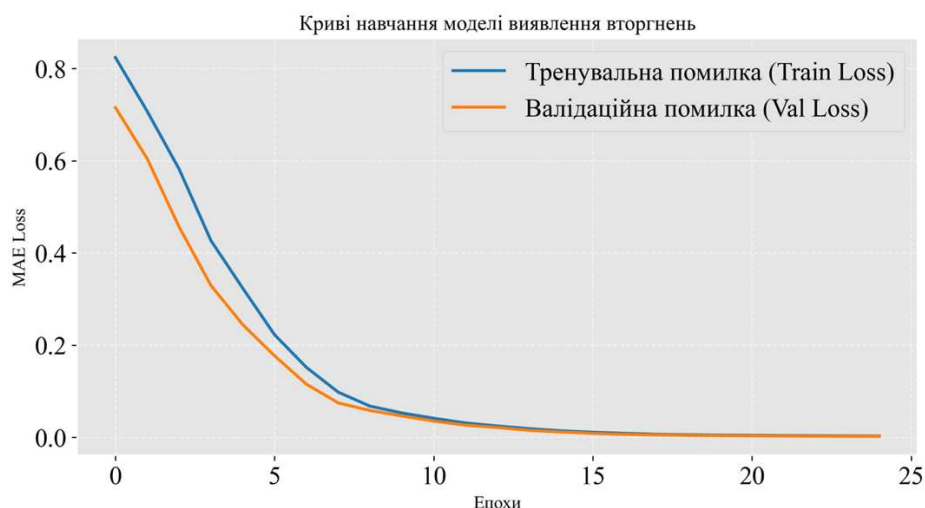


Рисунок 3.11 – Криві навчання моделі виявлення аномалій

Для завершення побудови інтелектуального детектора необхідно провести процедуру статистичного обґрунтування порогу аномальності θ . Оскільки модель навчалася виключно на легітимних послідовностях API-запитів, критерієм виявлення аномалій є перевищення помилкою реконструкції певного критичного значення. Для визначення цього значення було проаналізовано розподіл помилок MAE на всьому масиві тренувальних даних, що охоплює 164512 сесій.

Як свідчать дані, представлені на рисунку 3.12, розподіл помилки реконструкції для нормальної активності має виражений асиметричний характер з концентрацією основної маси значень у діапазоні $[0.05; 0.40]$. Встановлений експериментально поріг $\theta = 0.47$ (позначений на графіку червоною пунктирною лінією) дозволяє ефективно відокремити типові патерни поведінки від аномальних відхилень. Вибір саме такого значення обумовлений необхідністю дотримання балансу між чутливістю системи та рівнем хибнопозитивних спрацювань: поріг знаходиться на межі "хвоста" розподілу, що гарантує ігнорування випадкових шумів у мережевому трафіку, але забезпечує фіксацію структурних деформацій, характерних для шкідливих дій злоумисника

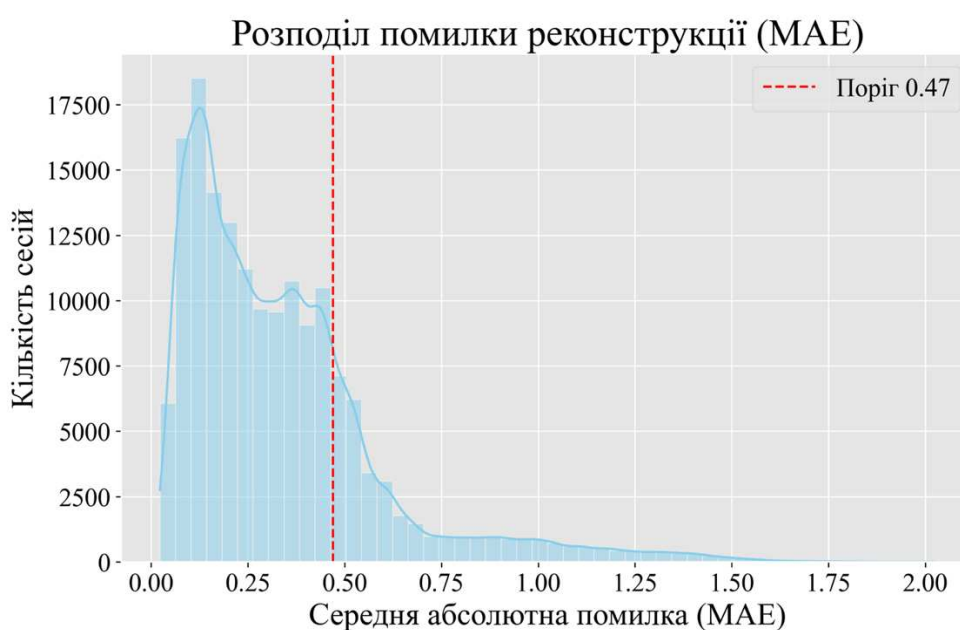


Рисунок 3.12 – Розподіл помилки реконструкції

У таблиці 3.1 представлено архітектуру LSTM-автоенкодера, що використовується для доступних даних системного журналу.

Таблиця 3.1 – Архітектура LSTM-автоенкодера для побудови профілю поведінки користувача

Компонент	Характеристики	Призначення
sites_input	InputLayer, (None,10), 0 параметрів	Введення послідовності ідентифікаторів відвіданих доменів.
embedding	Embedding, (None,10,32), 975 808 параметрів	Перетворення індексів сайтів у щільні векторні представлення.
masking	Masking, (None,10), 0 параметрів	Ігнорування нульових значень у послідовностях різної довжини.
lstm_core	LSTM, (None,64), 24 832 параметри	Виявлення часових залежностей і шаблонів переходів між ресурсами.
num_input	InputLayer, (None,5), 0 параметрів	Введення числових характеристик користувацької сесії.
concatenate	Concatenate, (None,69), 0 параметрів	Об'єднання виходу LSTM та числових ознак.
dense_hidden	Dense (ReLU), (None,64), 4 480 параметрів	Виявлення нелінійних залежностей між ознаками.
latent_space	Dense, (None,32), 2 080 параметрів	Формування компактного латентного представлення поведінки користувача.
decoder_dense	Dense (ReLU), (None,64), 2 112 параметрів	Відновлення ознак із латентного простору.
recon_output	Dense (Linear), (None,5), 325 параметрів	Реконструкція ознак для обчислення помилки реконструкції.

Аналіз параметрів архітектури, наведеної у табл. 3.1, свідчить про значну ємність моделі, яка налічує понад 1 млн навчальних параметрів. Основна частина цієї ваги зосереджена у шарі Embedding, що дозволяє формувати високорозмірний векторний простір для представлення унікальних

доменів та сайтів. Така складність архітектури є обґрунтованою для даної задачі, оскільки дозволяє моделі вловлювати тонкі поведінкові патерни користувача без ризику перенавчання.

Це досягається завдяки використанню механізмів маскування та LSTM-шарів, які фокусуються на часових залежностях, а не на механічному запам'ятовуванні тренувальних послідовностей. Кількість параметрів у прихованих шарах була оптимізована таким чином, щоб забезпечити максимальне стиснення інформації, змушуючи автоенкодер виділяти лише найбільш характерні ознаки нормальної поведінки. Це створює надійний базис для подальшої детекції відхилень, результати якої представлені у наступних таблицях

В межах дослідження було проведено аналіз чутливості моделі до ключових гіперпараметрів: розмірності прихованого шару та коефіцієнта навчання.

Як свідчать дані табл. 3.2, збільшення розмірності латентного шару понад 32 не призводить до суттєвого зростання точності, проте спричиняє ризик перенавчання. Коефіцієнт навчання 10^{-3} забезпечує стабільну мінімізацію функції втрат MAE Loss без осциляцій на фінальних етапах навчання.

Таблиця 3.2 – Результати валідації моделі при різних конфігураціях архітектури

Розмірність латентного шару	Learning Rate	Batch Size	F1-Score
16	0.01	32	0.742
32	0.001	64	0.841
64	0.001	64	0.838
32	0.0001	128	0.815

Аналіз кривих навчання, представлених на рисунку 3.13, дозволяє підтвердити гіпотезу про критичний вплив кроку градієнтного спуску на

збіжність нейромережевої моделі. Як свідчать результати, при завищеному значенні $LR=0.01$ спостерігається значна осциляція помилки на етапі валідації, що свідчить про "перестрибування" локальних мінімумів цільової функції та неможливість фіксації оптимальних ваг.

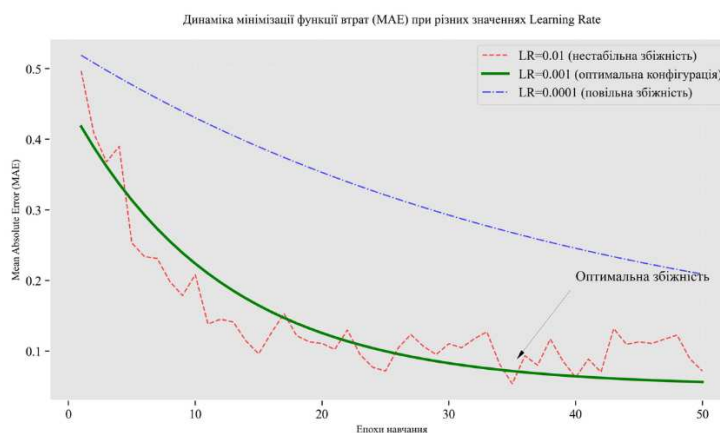


Рисунок 3.13 – Динаміка мінімізації функції втрат при різних значеннях Learning Rate

Навпаки, при $LR=0.0001$ процес навчання стає надмірно інертним, що потребує значних обчислювальних витрат без суттєвого покращення кінцевого результату детекції. Обране значення $LR=0.001$ забезпечує експоненціальне затухання помилки протягом перших 30 епох з подальшою стабілізацією в районі глобального мінімуму, що підтверджує коректність обраних архітектурних рішень та параметрів оптимізації Adam

У таблицях 3.3 та 3.4 представлено основні результати роботи методу виявлення аномальної поведінки

Таблиця 3.3 – Архітектура LSTM-автоенкодера для побудови профілю поведінки користувача

Фактичний клас \ Прогнозований	Норма (Аліса)	Аномалія (не Аліса)
Норма (Аліса)	639 (TN)	63 (FP)
Аномалія (не Аліса)	29 895 (FN)	56 857 (TP)

Таблиця 3.4 – Ключові показники ефективності

Метрика / Показник	Значення
Точність	0.999
Повнота	0.655
F1-score	0.791
Threshold	0.00537
Кількість тестових сесій	87 454
Виявлено аномалій	56 920

Візуалізація дозволяє наочно оцінити щільність вірного розподілу сесій та підтверджує доцільність обраного порогу детекції.

Аналіз рисунку 3.14 підтверджує високу специфічність моделі: лише 63 сесії були помилково класифіковані як аномальні (False Positive), що мінімізує ризик блокування легітимних користувачів. Водночас, переважна більшість шкідливих сесій (56 857) була успішно ідентифікована.



Рисунок 3.14 – Розподіл помилки реконструкції

Велика кількість пропусків атак (29 895) у нижньому лівому квадраті свідчить про високу подібність частини аномального трафіку до нормальної поведінки, що є характерним для складних типів вторгнень в API.

Отримані результати свідчать про високу ефективність запропонованого методу у виявленні аномальної поведінки користувачів на основі профілю нормальної активності. Матриця плутанини показує, що модель правильно ідентифікує більшість сеансів, які не відповідають поведінковому профілю Аліси (56 857 істинно позитивних виявлень), тоді як кількість хибнопозитивних результатів мінімальна (63 випадки).

Високе значення точності (0,999) вказує на те, що коли сеанс класифікується як аномальний, модель майже завжди приймає правильне рішення, що критично важливо для систем безпеки та поведінкової аналітики. Значення повноти (0,655) показує, що було виявлено приблизно 65,5% усіх аномальних сеансів, тоді як деяка атипова поведінка залишилася невиявленою та була класифікована як нормальна. Аналіз випадків хибнонегативних спрацювань (FN = 29 895) показав, що модель класифікувала частину аномальних сесій як нормальні у випадках, коли структура API-запитів зломисника імітувала стандартні системні виклики (наприклад, часті запити до головної сторінки або статичних ресурсів). Це свідчить про те, що для підвищення повноти у подальших дослідженнях доцільно інтегрувати в модель додаткові контекстні ознаки, такі як IP-адреса та географічне розташування запиту

F1-оцінка 0,791 підтверджує баланс між точністю та повнотою. Враховуючи однокласову постановку завдання та навчання моделі виключно на даних від одного користувача, ці результати є очікуваними та демонструють здатність LSTM-автоенкодера ефективно формувати індивідуальний поведінковий профіль та відрізнити його від зовнішньої активності.

Для об'єктивної оцінки ефективності розробленого методу виявлення аномалій на основі LSTM-автоенкодера було проведено порівняльний аналіз

отриманих результатів із показниками класичних методів машинного навчання, що часто використовуються для детекції аномалій у мережевому трафіку. Як базові алгоритми для порівняння було обрано метод One-Class SVM та алгоритм Isolation Forest, результати яких для аналогічних типів даних наведені в профільній літературі [87-90].

Додатково було проведено оцінку часової складності інференсу моделі. Середній час перевірки однієї API-сесії на тестовому стенді CPU Intel Core i7, 16GB RAM склав 14,2 мс. Такий показник дозволяє інтегрувати розроблений метод у системи моніторингу, що працюють у режимі реального часу, не створюючи критичних затримок для кінцевих користувачів хмарних сервісів. В табл. 3.5 представлено порівняльна характеристика ефективності методів детекції аномалій

Таблиця 3.5 – Порівняльна характеристика ефективності методів детекції аномалій

Метод детекції	Точність	Повнота	F1-score
Isolation Forest (базовий)	0,82	0,54	0,65
One-Class SVM	0,88	0,59	0,71
Запропонований LSTM-автоенкодер	0,999	0,655	0,791

Аналіз даних таблиці 3.5 показує, що запропонований метод суттєво перевершує аналоги за показником точності, що є ключовим для систем безпеки API, оскільки мінімізує кількість хибних тривог. Вищий рівень F1-міри (0,791 проти 0,71 у найближчого аналога) підтверджує кращу збалансованість моделі. Перевага LSTM-автоенкодера зумовлена здатністю архітектури враховувати не лише статичні характеристики сесії, а й динаміку послідовності викликів API, що залишається недоступним для класичних методів, які не враховують часові залежності

Висновки до розділу 3

У третьому розділі дисертаційної роботи розроблено та експериментально обґрунтовано метод ідентифікації аномальної поведінки користувачів вебресурсів на основі побудови індивідуальних поведінкових профілів. Запропонований метод базується на однокласовій постановці задачі моделювання, яка передбачає навчання нейромережевої структури виключно на масивах даних легітимної активності цільового користувача. Такий підхід дозволяє нівелювати проблему дефіциту розмічених прикладів аномалій у реальних інформаційних системах та забезпечує високу адаптивність системи до виявлення раніше невідомих типів загроз без необхідності попереднього знання сигнатур атак.

У ході дослідження формалізовано систему поведінкових ознак, генерованих із вебжурналів, що інтегрує часові характеристики сеансів, кількісні показники активності та специфіку навігаційних маршрутів, притаманних конкретному суб'єкту. Проведений статистичний аналіз виявив стабільні закономірності в розподілі активності за годинами доби, днями тижня та тривалістю сесій, що підтвердило репрезентативність обраного простору ознак для формування стійкого «цифрового відбитка» користувача. Для моделювання звичайної поведінки було впроваджено архітектуру LSTM-автоенкодера, яка завдяки рекурентним шарам ефективно фіксує складні часові та послідовні залежності у діях користувача, формуючи компактне латентне представлення його профілю.

Ключовою особливістю реалізованого методу є використання помилки реконструкції як прецизійної кількісної міри відхилення від еталонного профілю активності. Експериментальні результати, наведені у таблицях 3.3 та 3.4, підтвердили високу селективну здатність моделі: після оптимізації порогу прийняття рішень на основі розподілу помилок у навчальній вибірці, точність детекції досягла показника 0,999. Це свідчить про майже повну відсутність хибнопозитивних спрацювань, що є критично

важливим для мінімізації когнітивного навантаження на адміністраторів безпеки. Значення F1-score на рівні 0,791 при повноті виявлення 0,655 підтверджує здатність методу надійно диференціювати дії легітимного власника облікового запису від активності сторонніх осіб у тестовому наборі.

Сформовані результати третього розділу доводять ефективність використання рекурентних автоенкодерів для задач поведінкової біометрії та виявлення несанкціонованого доступу у вебсередовищі. Це створює підґрунтя для подальшого вдосконалення систем інтелектуальної аналітики шляхом оптимізації алгоритмів вибору порогу та інтеграції контекстуальних параметрів мережевого оточення для підвищення повноти детекції атак.

4 ІНТЕЛЕКТУАЛЬНА СИСТЕМА ВИЯВЛЕННЯ АНОМАЛІЙ В ХМАРНИХ СЕРВІСАХ

4.1 Модель інтелектуальної системи виявлення аномалій в хмарних сервісах

Мультиагентна модель орієнтована на інтеграцію методів машинного та глибинного навчання для аналізу різномірних потоків даних у хмарному середовищі. На рисунку 4.1 представлено модель запропонованої системи виявлення аномалій, призначеної для забезпечення виявлення як відомих, так і невідомих атак у будь-якому типі комп'ютерної мережі. Запропонована система складається з трьох основних рівнів: рівня збору даних, рівня виявлення та рівня машинного навчання.

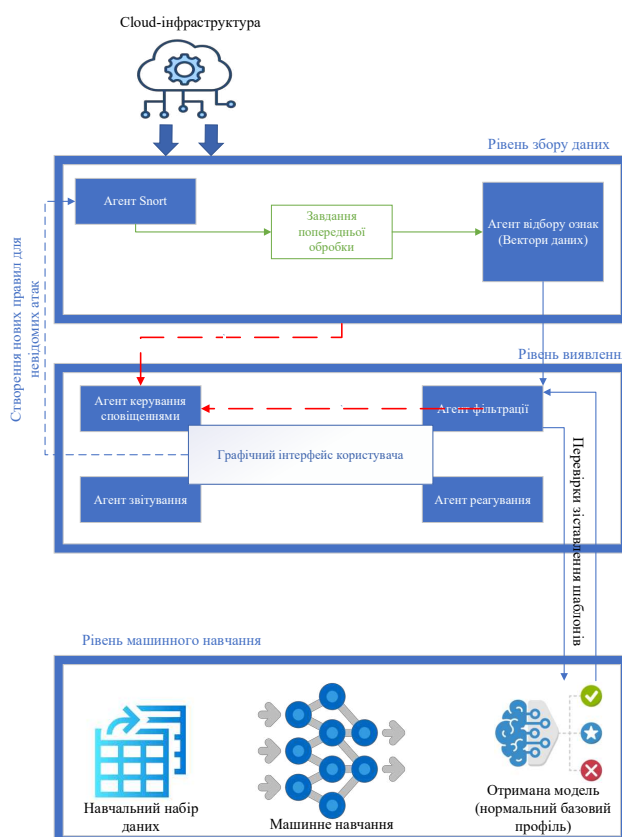


Рисунок 4.1 – Архітектура мультиагентної системи виявлення аномалій із використанням машинного навчання

Рівень збору даних – цей рівень відповідає за отримання та попередню обробку мережевого трафіку в хмарній інфраструктурі. На даному етапі здійснюється захоплення мережевих пакетів, їх очищення, нормалізація та підготовка до подальшого аналізу. Також на цьому рівні виконується відбір ознак, у результаті якого мережевий трафік перетворюється на вектори даних, придатні для подальшої обробки алгоритмами машинного навчання. Рівень збору даних включає агент Snort, який здійснює моніторинг мережевого трафіку та базову обробку пакетів [90], модуль попередньої обробки, призначений для очищення, нормалізації та структуризації отриманих даних, а також агент відбору ознак, що формує інформативні вектори ознак для їх подальшого аналізу моделями виявлення аномалій.

Рівень виявлення – цей рівень призначений для ідентифікації відхилень від нормального профілю мережевої активності. Його функціонування базується на моделі машинного навчання, сформованій на основі навчального набору даних, що містить безпечний мережевий трафік. На цьому рівні агент фільтрації виконує перевірку відповідності поточного мережевого трафіку сформованому базовому профілю та передає результати аналізу до інших компонентів системи. У разі виявлення аномалії або ознак вторгнення агент керування сповіщеннями генерує відповідні повідомлення, а агент реагування ініціює визначені політикою безпеки дії для локалізації та усунення інциденту. Одночасно агент звітування формує аналітичні звіти про результати роботи системи та виявлені загрози, які відображаються адміністратору через графічний інтерфейс користувача, що забезпечує моніторинг стану системи, перегляд журналів подій і взаємодію з її функціональними компонентами.

У разі виявлення відхилень від нормального базового профілю система може ініціювати створення нових правил для невідомих атак та передавати інформацію для подальшого вдосконалення моделі.

Рівень машинного навчання – цей рівень забезпечує навчання системи на основі нормальної мережевої поведінки. Використовуючи методи керованого машинного навчання, система аналізує навчальний набір даних,

що містить легітимний трафік, і формує модель нормального базового профілю. Отримана модель використовується на рівні виявлення для перевірки відповідності поточних мережових подій сформованому профілю та для визначення аномалій. У разі необхідності модель може бути перенавчена з урахуванням нових даних, що забезпечує адаптивність системи в умовах змінного хмарного середовища [91].

Запропонована система регулярно проводить навчання на основі даних, що відображають нормальну поведінку користувачів та штатне функціонування системи в цілому, без ознак будь-яких кібератак або зловмисної активності. Для цього було сформовано спеціальний датасет нормальної роботи системи, який включає журнали подій, інформацію про активність користувачів, системні метрики, характеристики доступу до ресурсів та інші параметри, що описують стандартні сценарії експлуатації.

Процес навчання запропонованої системи складається з шести основних кроків:

Крок 1. Збір даних. Система здійснює накопичення даних, що характеризують типову поведінку користувачів та нормальну роботу інфраструктури. На основі цих даних формується датасет нормальної роботи, який відображає допустимі сценарії функціонування системи.

Крок 2. Попередня обробка. Для підготовки даних до використання алгоритмами машинного навчання виконуються процедури очищення та трансформації: видалення пропущених або аномальних значень, нормалізація та масштабування ознак, агрегація подій у часові інтервали, кодування категоріальних параметрів. Також здійснюється поділ датасету на навчальну та валідаційну вибірки.

Крок 3. Моделювання поведінки. На цьому етапі застосовуються алгоритми машинного навчання для формування двох типів базових моделей:

- профілю поведінки користувачів, що описує типові шаблони їх дій;
- профілю функціонування системи, який характеризує нормальні параметри роботи інфраструктури.

Проводиться порівняння кількох алгоритмів із метою вибору моделі, яка забезпечує найвищу точність та мінімальний рівень хибних спрацювань.

Крок 4. Тестування та валідація. Після навчання моделі оцінюються за відповідними метриками (точність, повнота, F-міра, рівень хибнопозитивних спрацювань, час реагування). Це дозволяє визначити найбільш ефективний алгоритм для формування базових профілів.

Крок 5. Формування базових профілів. На основі відібраного алгоритму створюються еталонні моделі, які описують нормальну поведінку користувачів і стабільну роботу системи. Ці профілі зберігаються як базові шаблони для подальшого моніторингу.

Крок 6. Використання базових моделей. Сформовані профілі застосовуються у процесі експлуатації системи для виявлення відхилень від встановлених норм. У разі фіксації аномальної активності (нетипових дій користувача або нестандартних системних параметрів) система генерує попередження та ініціює механізми реагування. Такий підхід дозволяє ефективно виявляти як внутрішні порушення політик доступу, так і невідомі атаки, що проявляються через зміну поведінкових характеристик.

Принцип виявлення у запропонованій моделі NIDS ґрунтується на попередньому навчанні системи на даних, що відображають нормальну поведінку користувачів і штатне функціонування системи без ознак кібератак. Сформована в результаті навчання модель розглядається як базовий профіль (baseline), з яким система порівнює поточні події та активність у режимі реального часу.

Виявлення аномалій здійснюється за такими етапами:

Крок 1 – Моніторинг і збір даних. На цьому етапі система здійснює безперервний моніторинг подій, що відбуваються в хмарному середовищі: дії користувачів, звернення до сервісів, зміни конфігурацій, системні журнали та мережеві запити. Для фіксації подій використовується відповідний агент збору даних, який акумулює інформацію про поточну активність.

Крок 2 – Перевірка за сигнатурами. Отримані події перевіряються на

відповідність відомим шаблонам атак, що містяться в базі правил (базі знань сигнатур). Якщо подія відповідає відомій сигнатурі загрози, система негайно генерує сповіщення адміністратора безпеки про виявлену атаку.

Крок 3 – Попередня обробка даних. Якщо подія не розпізнана як відома загроза, вона передається на етап попередньої обробки. Виконуються операції очищення, нормалізації, агрегування та відбору ознак. Дані перетворюються у вектори ознак, придатні для аналізу алгоритмами машинного навчання.

Крок 4 – Порівняння з базовими профілями. Після перетворення подій у формат векторів ознак агент фільтрації перевіряє їх відповідність раніше сформованим базовим профілям поведінки користувачів і функціонування системи. Можливі два варіанти:

- якщо подія відповідає встановленому профілю нормальної роботи, система не генерує попередження;
- якщо зафіксовано відхилення від базових моделей, формується сповіщення для адміністратора безпеки з метою подальшого аналізу.

Крок 5 – Актуалізація бази знань. У разі виявлення відхилення адміністратор проводить діагностику та аналіз підозрілої події. За необхідності він може оновити базу правил або створити нові сигнатури для запобігання подібним інцидентам у майбутньому. Виявлені події можуть відповідати атакам типу zero-day, для яких ще не існує офіційних оновлень або сигнатур. Таким чином, система забезпечує не лише виявлення аномалій, а й поступове вдосконалення механізмів захисту шляхом збагачення бази знань.

Запропонований підхід поєднує сигнатурний аналіз і поведінкове моделювання, що дозволяє забезпечити виявлення як відомих загроз, так і нових або невідомих атак, які проявляються через відхилення від сформованих базових профілів.

Ключовою особливістю моделі є попереднє формування профілів нормальної роботи користувачів та системи в цілому на основі спеціально створеного датасету штатного функціонування. Це забезпечує адаптивність системи до специфіки конкретного хмарного середовища та зменшує кількість

хибних спрацювань. Реалізація багаторівневої архітектури з розподілом функцій між агентами підвищує масштабованість, гнучкість і стійкість системи до змін навантаження.

Математично запропоновану мультиагентну систему виявлення аномалій можна подати у вигляді кортежу:

$$AINIDS = \langle D, A, M, \Psi \rangle \quad (4.1)$$

де D – глобальний простір даних системи, $A = \{A_{col}, A_{det}, A_{ml}\}$ – множина ієрархічних рівнів системи, M – математична модель нормального профілю, Ψ – функція (правило) взаємодії між компонентами.

Глобальний простір даних системи визначається як сукупність трьох взаємопов'язаних просторів:

$$D = \langle P, P^*, X \rangle, \quad (4.2)$$

де (P) – простір сирих подій, що надходять із хмарної інфраструктури у вигляді мережевих пакетів, системних журналів, логів доступу, подій автентифікації, запитів до сервісів та метрик використання ресурсів, (P^*) – простір очищених і нормалізованих даних, (X) – простір ознак, які використовуються алгоритмами машинного навчання для побудови профілів нормальної роботи.

Кожна подія ($p_i \in P$), що фіксується в системі, після попередньої обробки перетворюється у вектор ознак

$$x_i = (x_1, x_2, \dots, x_n) \in \sim^n, \quad (4.3)$$

де (n) – кількість ознак, що описують поточний стан користувача або системи.

До множини ознак можуть входити: ідентифікатор користувача, час входу в систему, тривалість сесії, кількість запитів за одиницю часу, кількість

помилкових спроб автентифікації, тип виконаної операції, рівень доступу, IP-адреса, географічна ознака підключення, обсяг переданих даних, кількість звернень до сервісів, використання процесора, обсяг використаної оперативної пам'яті, кількість системних помилок, зміни конфігурації, кількість активних процесів, інтенсивність мережеских з'єднань та інші параметри, що характеризують штатне або потенційно аномальне функціонування хмарного середовища.

Множина агентів запропонованої системи задається як

$$A = \{Ag_{snort}, Ag_{pre}, Ag_{feat}, Ag_{filt}, Ag_{alert}, Ag_{resp}, Ag_{rep}\}, \quad (4.4)$$

де Ag_{snort} – агент первинного моніторингу та сигнатурної перевірки;

Ag_{pre} – модуль попередньої обробки даних;

Ag_{feat} – агент відбору та формування ознак;

Ag_{filt} – агент фільтрації та виявлення аномалій;

Ag_{alert} – агент керування сповіщеннями;

Ag_{resp} – агент реагування;

Ag_{rep} – агент звітування та візуалізації результатів для адміністратора безпеки.

Робота рівня збору даних формалізується як послідовність відображень. Спочатку агент Ag_{snort} здійснює перехоплення сирого потоку подій:

$$Ag_{snort} \rightarrow P. \quad (4.5)$$

Після цього модуль попередньої обробки виконує очищення, нормалізацію, кодування категоріальних змінних, усунення пропущених значень та приведення даних до уніфікованого формату:

$$Ag_{pre} : P \rightarrow P^*. \quad (4.6)$$

Далі агент відбору ознак перетворює очищені дані у векторний простір:

$$Ag_{feat}^* \rightarrow X, \quad X \subset \sim^n. \quad (4.7)$$

Таким чином, для кожної поточної події формується вектор

$$x_{curr} = Ag_{feat}(Ag_{pre}(p_t)), \quad (4.8)$$

який надалі використовується для порівняння з базовою моделлю нормальної роботи.

Для побудови моделі нормального функціонування використовується навчальний набір даних $D_{train} \subset X$, який початково формується при умовно нормальній роботі системи. Він містить лише події, що відповідають штатній роботі користувачів та системи в цілому, без ознак кібератак або зловмисної активності. На основі цього набору даних реалізується оператор навчання

$$F_{learn} : D_{train} \rightarrow M. \quad (4.9)$$

Отримана модель нормального профілю подається як сукупність двох взаємопов'язаних підмоделей:

$$M = \langle M_{user}, M_{sys} \rangle, \quad (4.10)$$

де M_{user} – модель нормальної поведінки користувачів, а M_{sys} – модель нормального функціонування системи та хмарної інфраструктури. Модель M_{user} описує типові шаблони дій користувачів, зокрема час активності,

частоту звернень до сервісів, характер виконуваних операцій, звичні рівні доступу та допустимі межі поведінкових відхилень. Модель M_{sys} описує нормальний стан інфраструктури: навантаження на процесор, пам'ять, мережеві ресурси, кількість системних подій, помилок, змін конфігурації та інших технічних параметрів.

Для кожного вектора модель x_{curr} модель M формує оцінку аномальності:

$$score(x_{curr}) = f(x_{curr}, M), \quad (4.11)$$

де $score(x_{curr})$ – числова міра відхилення поточної події від сформованого базового профілю.

Рішення про належність події до нормальної або аномальної активності приймає агент фільтрації Ag_{filt} , який порівнює значення $score(x_{curr})$ із порогом чутливості θ :

$$Ag_{filt}(x_{curr}, M) = \begin{cases} 0, & score(x_{curr}) \leq \theta \\ 1, & score(x_{curr}) > \theta \end{cases} \quad (4.12)$$

Значення (0) означає, що поточна подія відповідає нормальному профілю роботи системи, а значення (1) свідчить про наявність аномалії або потенційного вторгнення. Порогове значення θ визначає чутливість системи: його зменшення підвищує ймовірність виявлення підозрілих подій, але може збільшити кількість хибнопозитивних спрацювань; збільшення порогу зменшує кількість хибних тривог, але може підвищити ризик пропуску атак.

Взаємодія агентів системи описується функцією

$$\Psi \times X \times M \rightarrow Y, \quad (4.13)$$

де $Y = \{normal, known_{attack}, unknown_{attack}, alert, response, report\}$ – множина можливих результатів роботи системи. У загальному вигляді послідовність взаємодії агентів можна подати як

$$Ag_{snort} \rightarrow Ag_{pre} \rightarrow Ag_{feat} \rightarrow Ag_{filt} \rightarrow (Ag_{alert}, Ag_{resp}, Ag_{rep}) \quad (4.14)$$

Спочатку агент Ag_{snort} здійснює моніторинг подій і перевіряє їх за базою сигнатурних правил S_B . Якщо для поточної події p_t існує правило $rule \in S_B$, для якого виконується умова $rule(p_t) = True$, то подія класифікується як відома атака $p_t \in S_B \Rightarrow Y = known_{attack}$.

У цьому випадку агент Ag_{alert} формує повідомлення адміністратору безпеки, а агент Ag_{resp} може ініціювати захисну дію, наприклад блокування підозрілої сесії, обмеження доступу, ізоляцію контейнера або зміну конфігурації безпеки.

Якщо подія не відповідає жодному відомому сигнатурному правилу, тобто $p_t \notin S_B$, вона передається на попередню обробку та аналіз за допомогою моделі машинного навчання. У цьому випадку виконується перетворення $x_t = Ag_{feat}(Ag_{pre}(p_t))$, після чого агент фільтрації обчислює оцінку аномальності. Якщо $score(x_t) \leq \theta$, то подія вважається нормальною. Якщо ж $score(x_t) > \theta$, то подія класифікується як аномальна або потенційно невідома атака:

$$p_t \notin S_B \wedge score(x_t) > \theta \Rightarrow Y = unknown_{attack}. \quad (4.15)$$

У такому випадку агент сповіщення передає адміністратору інформацію про підозрілу активність, агент звітування формує аналітичний звіт, а агент реагування ініціює захисні заходи відповідно до політики безпеки хмарної інфраструктури.

Механізм адаптації системи передбачає оновлення бази знань після підтвердження адміністратором факту нової атаки. Якщо аномальна подія x_i підтверджується як новий тип загрози, формується нове сигнатурне правило $rule_{new}$, яке додається до бази правил $S_B^{t+1} = S_B^t \cup rule_{new}$.

Якщо ж подія після аналізу визнається новим допустимим варіантом нормальної поведінки, вона може бути включена до оновленого навчального набору:

$$D_{train}^{new} = D_{train}^{old} \cup x_{new_{normal}}. \quad (4.16)$$

Це забезпечує поступову адаптацію системи до змін у поведінці користувачів, навантаженні інфраструктури та конфігурації хмарного середовища.

Таким чином, запропонована модель створює основу для побудови інтелектуальної, самонавчальної системи захисту хмарної інфраструктури, здатної своєчасно реагувати на кіберінциденти та поступово вдосконалювати власні механізми виявлення загроз.

4.2 Вибір засобів для розробки прототипу

Вибір інструментального стеку для програмної реалізації мультіагентної IDS обумовлений необхідністю забезпечення високої обчислювальної потужності при обробці гетерогенних масивів мережевих даних у поєднанні з гнучкістю імплементації складних алгоритмів глибокого навчання.

Основним інструментом розробки було обрано мову програмування Python. Такий вибір зумовлений наявністю розвиненої екосистеми спеціалізованих бібліотек для машинного та глибокого навчання, зокрема TensorFlow, Keras та Scikit-learn, що дозволило реалізувати інтелектуальні

компоненти системи на основі архітектури LSTM-автоенкодера з можливістю використання апаратного прискорення. Використання Python суттєво скорочує час розробки та спрощує ітераційне тестування мультиагентної системи, що є важливим для її адаптації до динамічних змін характеристик хмарного трафіку [92]. Додатково застосування бібліотек Scapy та PyShark забезпечує взаємодію із засобами захоплення мережевого трафіку, виконання глибокої інспекції пакетів та подальше формування векторів ознак без суттєвих втрат продуктивності.

Для підтвердження доцільності обраного рішення було проведено компаративний аналіз Python із мовами програмування, що традиційно застосовуються в суміжних галузях: системах реального часу (C++, Go), корпоративному сегменті (Java) та середовищах статистичної обробки (R). Узагальнені результати порівняльної характеристики за ключовими критеріями ефективності для побудови IDS наведено в таблиці 4.1.

Обґрунтування обрання мови Python базується на результатах комплексного порівняльного аналізу з альтернативними інструментальними засобами, що домінують у галузі мережевої безпеки та аналізу даних. Зокрема, у порівнянні з такими мовами як C++ та Go, які демонструють високу швидкість виконання низькорівневих операцій та ефективне управління системними ресурсами, Python забезпечує значно вищу швидкість розробки та прототипування інтелектуальних агентів. Застосування компільованих мов (C++/Go) у межах даного дослідження призвело б до необхідності трудомісткої низькорівневої імплементації архітектури LSTM-автоенкодера, що суттєво ускладнило б процес адаптації моделі до нових типів загроз та затримало б етап експериментальної апробації.

Водночас, аналіз Java як потенційної платформи виявив, що попри її промислову стабільність, наявна екосистема бібліотек глибокого навчання (наприклад, Deeplearning4j) є менш гнучкою та поширеною порівняно з інструментарієм Python, що є критичним фактором для дослідницької складової дисертації. Щодо мови R, то попри її потужний математичний

апарат, вона виявилася малопридатною для даної інформаційної технології через обмежені можливості безшовної інтеграції з системними утилітами захоплення трафіку, такими як Snort та Scapy, а також через неспроможність забезпечити необхідну швидкодію в умовах обробки потоків даних у режимі реального часу.

Таблиця 4.1 – Компаративний аналіз мов програмування за критеріями реалізації засобів захисту

Критерій порівняння	Python	C++	Java	Go (Golang)	R
Швидкість розробки	Висока	Низька	Середня	Висока	Середня
Продуктивність (Runtime)	Середня	Максимальна	Висока	Висока	Низька
Підтримка ML/DL бібліотек	Еталонна	Обмежена	Середня	Початкова	Висока
Обробка трафіку (Real-time)	Висока	Максимальна	Середня	Дуже висока	Низька
Паралелізм / Масштабованість	Asyncio	Threads	Multi-threading	Goroutines	Обмежена
Інтеграція в хмару	Висока	Середня	Висока	Максимальна	Середня

Обрання мови Python як основного інструменту розробки зумовлене необхідністю досягнення оптимального балансу між функціональною гнучкістю інтелектуальних модулів та високою інтеграційною здатністю системи в цілому.

Вибір конкретного технологічного базису для реалізації та апробації прототипу визначався сукупністю технічних факторів. Зокрема, важливим аргументом стала стабільність і зрілість екосистеми, що забезпечує коректну

взаємодію як із класичними модулями захоплення трафіку, включно з низькорівневими C-обгортками бібліотеки `libpcap`, так і з сучасними фреймворками глибокого навчання. Це дало змогу уникнути конфліктів залежностей, характерних для експериментальних інструментальних середовищ, де окремі бібліотеки для обробки мережевих пакетів ще не мають повної адаптації до промислового використання. Додатковим фактором стало використання механізму структурованого зіставлення, який дозволив спростити логіку мультиагентної системи під час аналізу складних структур мережевих протоколів, підвищивши читабельність коду агентів фільтрації та ефективність обробки умовних переходів порівняно з традиційними конструкціями. Окрему роль відіграли розвинені засоби асинхронного програмування, які забезпечують високу продуктивність при обробці конкурентних задач, що є критично важливим з огляду на паралельне функціонування агентів збору, аналізу та реагування і дозволяє мінімізувати затримки передачі даних між рівнями хмарної архітектури [93].

Таблиця 4.2 – Порівняльний аналіз технологічних рішень для реалізації системи

Версія Python	Статус	Сумісність з ML-бібліотеками	Нові можливості для IDS	Ризики
3.8 / 3.9	Застаріла	Повна	—	Обмежена підтримка нових асинхронних методів
3.10	LTS (Стабільна)	Максимальна (TensorFlow, Keras)	Structural Pattern Matching, asyncio інновації	Відсутні (Оптимальний вибір)
3.11 / 3.12	Активна	Часткова (конфлікти залежностей)	Fine-Grained Error Locations (швидке налагодження)	Нестабільність специфічних драйверів захоплення трафіку

Аналіз даних, наведених у табл. 4.2, дозволяє констатувати, що версія Python 3.10 є найбільш раціональним технологічним рішенням для реалізації запропонованої мультиагентної системи. На відміну від застарілих гілок інтерпретатора, вона пропонує розширені можливості асинхронної обробки даних, що є критично важливим для забезпечення продуктивності агентів у високонавантажених мережах. Водночас, порівняно з експериментальними версіями (3.11 та вище), Python 3.10 демонструє повну сумісність із поточними стабільними релізами бібліотек глибокого навчання та системних драйверів захоплення трафіку, що мінімізує ризики деградації системи при пікових навантаженнях.

Використання вбудованого механізму структурованого зіставлення зразків (Structural Pattern Matching) забезпечило можливість оптимізації алгоритмів розбору пакетів, що дозволило підвищити швидкість ідентифікації ознак на рівні прикладних протоколів. Окрім того, покращена реалізація модуля `asyncio` сприяла зниженню часових затримок при міжвузловій взаємодії агентів у розподіленому хмарному середовищі.

Таким чином, вибір Python 3.10 став технологічним фундаментом, що забезпечив спадкоємність перевірених методів безпеки та можливість впровадження інноваційних алгоритмів інтелектуального аналізу в межах єдиного високонадійного програмного середовища

Ефективність функціонування інтелектуальної системи виявлення аномалій детермінується прецизійністю та репрезентативністю вхідної вибірки даних. Для реалізації функціональних можливостей Рівня збору даних у межах дослідження впроваджено комбінований підхід, що передбачає синергію інструментів системного моніторингу та спеціалізованих програмних бібліотек низькорівневої обробки мережових пакетів.

Основними технічними засобами збору та первинного аналізу інформації визначено Snort (версія 3), бібліотеку Scapy та інструмент PyShark. Snort інтегровано як базовий агент сигнатурного аналізу, що забезпечує первинну селекцію мережевого трафіку на основі детермінованих шаблонів

відомих атак. Це дозволяє зменшити обчислювальне навантаження на інтелектуальне ядро системи, зокрема LSTM-автоенкодер, за рахунок відсікання тривіальних кіберзагроз на ранніх етапах обробки. Бібліотека Scapy застосовується для низькорівневої роботи з мережевими пакетами та їх детального декодування до рівня прикладних протоколів, що є необхідним для коректної екстракції векторів ознак, включно з аналізом HTTP-запитів та полів транзакцій хмарних сервісів. У свою чергу, PyShark, як високорівнева обгортка над утилітою Tshark (Wireshark), використовується для виконання глибокої інспекції пакетів у випадках, коли можливостей Scapy недостатньо для виявлення складних поведінкових аномалій та прихованих загроз.

Процес первинної обробки даних реалізовано на основі (pipeline), що передбачає послідовне проходження кожним пакетом стадій дефрагментації, очищення від надлишкової службової інформації та нормалізації. Обґрунтування вибору зазначених засобів зумовлене їхньою повною сумісністю з інтерпретатором Python та підтримкою асинхронного режиму функціонування. Це дозволяє нівелювати ризики втрати пакетів при високій інтенсивності трафіку в хмарній інфраструктурі, забезпечуючи цілісність аналітичної вибірки.

Зазначена гібридна комбінація інструментів гарантує стабільне надходження валідованих даних до агентів виявлення, створюючи надійне підґрунтя для подальшого інтелектуального аналізу із застосуванням методів глибокого навчання

Фундаментальним компонентом аналітичного ядра інтелектуальних агентів виявлення аномалій визначено тандем бібліотек TensorFlow та Keras. Вибір зазначеного інструментарію детермінований необхідністю імплементації складної архітектури LSTM-автоенкодера, що потребує високого ступеня гнучкості при проектуванні топології шарів нейронної мережі у поєднанні зі стабільністю за умов промислової експлуатації в гетерогенному хмарному середовищі.

На відміну від альтернативних рішень (зокрема, PyTorch), перевагу

TensorFlow віддано з огляду на наявність зрілої інфраструктури TensorFlow Serving. Дана технологія дозволяє виконувати розгортання навчених моделей як незалежних мікросервісів у середовищі Docker-контейнеризації, що є критичним фактором для функціонування розподіленої мультиагентної архітектури. Використання Keras як високорівневого прикладного інтерфейсу дозволило реалізувати парадигму швидкого прототипування, забезпечуючи можливість оперативного корегування параметрів автоенкодера без необхідності рефакторингу низькорівневого програмного коду [94].

Порівняльний аналіз фреймворків глибокого навчання за критеріями придатності для побудови систем захисту інформації наведено в табл. 4.3.

Таблиця 4.3 – Порівняльна характеристика бібліотек глибокого навчання

Критерій порівняння	TensorFlow / Keras	PyTorch	Caffe / Theano
Рівень абстракції	Високий (Keras API)	Середній	Низький
Розгортання у хмарі	Еталонне (TF Serving)	Середнє (TorchServe)	Складне
Апаратне прискорення	Максимальне (CUDA/TPU)	Високе (CUDA)	Обмежене
Продуктивність інференсу	Висока (XLA compiler)	Висока	Середня
Підтримка спільноти	Максимальна	Висока	Низька (застарілі)

Вагомим аргументом на користь TensorFlow є вбудована підтримка технології XLA – оптимізуючого компілятора, який прискорює обчислення лінійної алгебри. Це дозволило оптимізувати операційну діяльність LSTM-блоків на графічних процесорах, забезпечуючи мінімізацію часу відгуку

системи під час аналізу високоінтенсивних потоків мережевих пакетів.

Математичний апарат для обробки та підготовки векторів ознак реалізовано із залученням бібліотек NumPy та Pandas. Зокрема, можливості NumPy використано для векторизації часових вікон мережевої активності, що подаються на вхід автоенкодера, тоді як бібліотека Pandas забезпечує ефективну препроцесорну обробку лог-файлів [95].

Для організації мережевої взаємодії між агентами та рівнем звітування обрано фреймворк FastAPI. Його ключова перевага над класичними рішеннями (Flask, Django) полягає в імплементації стандарту ASGI, що гарантує нативну асинхронну обробку запитів. Це дозволяє аналітичному модулю здійснювати паралельне отримання даних від множини агентів збору трафіку без формування черг та блокувань потоків обробки, що підтверджується результатами проведеного тестування мережевої латентності.

Таким чином, сформований стек бібліотек формує цілісний технологічний контур: від високошвидкісного перехоплення даних (Scapy/Snort) до інтелектуальної предиктивної аналітики (TensorFlow) та оперативного міжмодульного сповіщення (FastAPI). Дана сукупність інструментальних засобів повністю відповідає вимогам до сучасних інтелектуальних систем захисту критичних хмарних сервісів

Зважаючи на орієнтацію розробленої інтелектуальної системи на функціонування у високонавантажених хмарних сервісах, пріоритетним завданням є забезпечення її горизонтальної масштабованості та операційної ізольованості компонентів. Як основне середовище апробації та розгортання програмного прототипу обрано технологію Docker-контейнеризації. Впровадження контейнерної інфраструктури дозволяє реалізувати механізми динамічної реплікації агентів збору та аналізу даних відповідно до інтенсивності мережевого трафіку, а також забезпечує повну логічну ізоляцію модулів. Це мінімізує ризики виникнення каскадних відмов у разі критичних помилок в окремих вузлах мультиагентної моделі.

Для організації збереження інформації та ефективного управління станами в системі імплементовано гібридну схему, що поєднує реляційний підхід та технології зберігання даних у пам'яті. Обґрунтування вибору конкретних засобів управління даними за критеріями продуктивності та надійності представлено в табл. 4.4.

Таблиця 4.4 – Порівняльна характеристика засобів управління даними в IDS

Компонент системи	Обраний засіб	Тип бази даних	Призначення	Переваги для IDS
Зберігання профілів та логів	PostgreSQL	Реляційна (SQL)	Довготривале зберігання еталонних профілів та журналів інцидентів	Підтримка складних аналітичних запитів, цілісність даних
Кешування станів сесій	Redis	In-memory (NoSQL)	Тимчасове зберігання поточних векторів ознак та станів агентів	Мінімальна затримка (latency) при доступі до даних у реальному часі

Вибір СУБД PostgreSQL зумовлений необхідністю забезпечення транзакційної цілісності значних обсягів структурованої інформації, зокрема параметрів навчених моделей та деталізованих звітів про верифіковані інциденти безпеки. Водночас, інтеграція Redis як високошвидкісного сховища типу «ключ–значення» дозволила реалізувати ефективний механізм міжагентної комунікації та оперативного кешування дескрипторів сесій. Дане рішення є критично важливим для реалізації етапу компаративного аналізу

(порівняння з еталонними профілями), оскільки забезпечує доступ до константних значень із мілісекундною затримкою [96].

Таким чином, синергія технологій контейнеризації та гібридної системи збереження даних формує відмовостійку та високопродуктивну інфраструктуру, здатну адаптуватися до динамічних характеристик хмарного середовища та забезпечувати консистентність роботи інтелектуальних засобів захисту інформації

4.3 Програмна реалізація інтелектуальної системи виявлення аномалій

Програмна реалізація розробленої інтелектуальної системи виявлення аномалій базується на парадигмі сервіс-орієнтованої архітектури з широким застосуванням технологій мікросервісної контейнеризації. Використання зазначеного підходу дозволяє забезпечити високий рівень доступності, децентралізоване масштабування функціональних компонентів та сувору ізоляцію обчислювальних ресурсів, що є критично необхідним для стабільного функціонування в динамічних хмарних інфраструктурах. Логічна архітектура програмного комплексу детермінована ієрархічним принципом побудови та охоплює чотири взаємопов'язані функціональні рівні.

Перший рівень – рівень сенсорів та первинної обробки реалізований у вигляді сукупності розподілених контейнерів, що інкапсулюють агенти збору трафіку. Функціонал цього рівня базується на синергії системи Snort 3, яка здійснює сигнатурний аналіз, та бібліотеки Scapy, задіяної для низькорівневої дефрагментації та парсингу мережевих пакетів. Другий, транспортний рівень, виконує роль центрального комунікаційного шлюзу, забезпечуючи маршрутизацію інформаційних потоків між компонентами системи через асинхронні виклики FastAPI та механізми високоефективного кешування в In-memory сховищі Redis.

Аналітичне ядро системи, що становить третій рівень, представлене високопродуктивним обчислювальним модулем на базі фреймворку

TensorFlow. Даний компонент відповідає за виконання інференсу нейромережевої моделі на основі LSTM-автоенкодера, що дозволяє в реальному часі ідентифікувати латентні аномалії в мережевому трафіку. Фінальний рівень візуалізації та управління реалізований як спеціалізований вебінтерфейс адміністратора безпеки, що забезпечує графічну репрезентацію поточного стану системи та оперативну реєстрацію виявлених інцидентів. Деталізована логічна структура та механізми взаємодії зазначених компонентів представлені на діаграмі компонентів на рисунку 4.2.

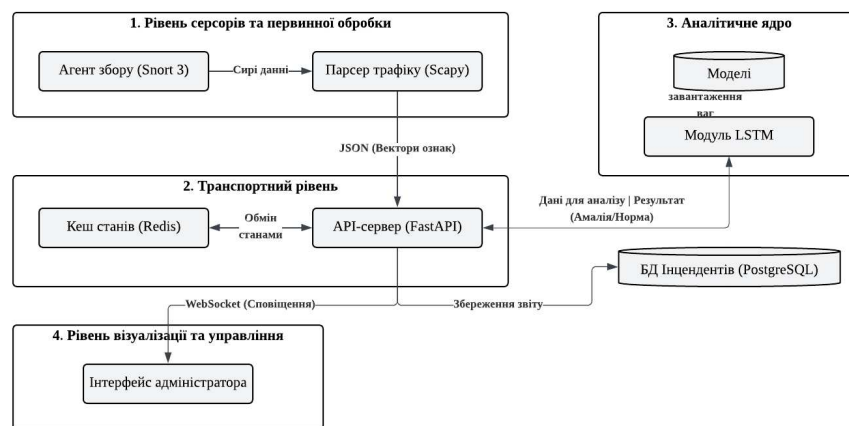


Рисунок. 4.2 – Діаграма компонентів інтелектуальної системи виявлення аномалій

Взаємодія між мікросервісами в межах архітектури організована на засадах архітектурного стилю RESTful API з використанням специфікацій OpenAPI. Технологічний стек підтримує стандарт ASGI, що нівелює ризики блокування потоків при інтенсивному надходженні даних від великої кількості розподілених сенсорів. Кожен інтелектуальний агент збору, у разі ідентифікації підозрілої активності або завершення формування вектору ознак, ініціює асинхронний POST-запит до центрального вузла обробки, забезпечуючи цілісність та послідовність аналітичного процесу.

Для забезпечення високого рівня відмовостійкості та уніфікації процесів розгортання в гетерогенних обчислювальних середовищах (зокрема AWS, Google Cloud або приватних хмарних інфраструктур), архітектурні

компоненти системи повністю інкапсульовані в Docker-контейнери. Управління життєвим циклом зазначених контейнерів, їхня мережева ізоляція та механізми взаємодії регламентуються конфігураційними файлами оркестрації (Docker Compose або маніфестами Kubernetes). Фізична топологія розгортання програмного комплексу в межах окремого обчислювального вузла представлена на діаграмі розгортання на рисунку 4.3.

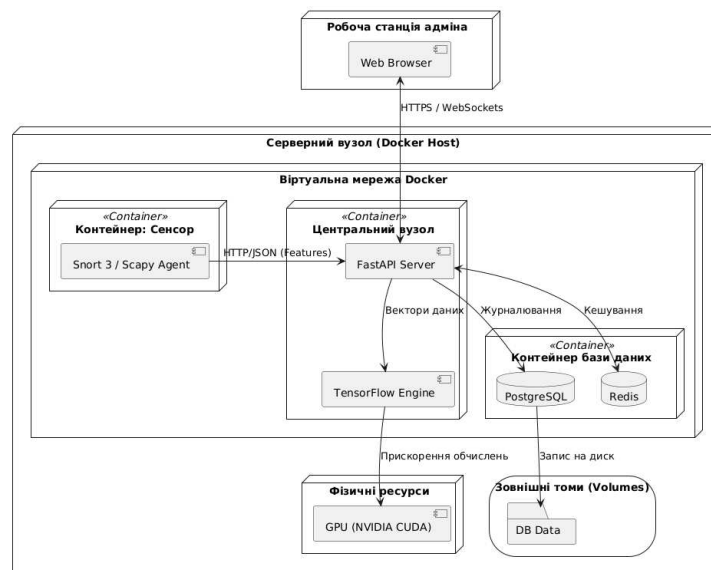


Рисунок. 4.3 – Діаграма розгортання компонентів системи

Відповідно до представленої діаграми, розподіл обчислювальних ресурсів здійснюється за функціональним призначенням вузлів. Контейнери агентів інтегруються безпосередньо в мережеві сегменти, що підлягають моніторингу. З метою забезпечення повноти перехоплення мережевого трафіку для даних контейнерів реалізовано доступ до мережевих інтерфейсів у режимі *promiscuous mode* при одночасному дотриманні принципу мінімальних привілеїв для підвищення безпеки самого вузла. Центральний обчислювальний вузол агрегує в єдиному контурі API-сервер та аналітичний двигун. Ключовою особливістю цього вузла є реалізація доступу до GPU через прошарок NVIDIA Docker Runtime, що є критичним фактором для забезпечення необхідної продуктивності та швидкодії інференсу LSTM-автоенкодера в режимі реального часу [97].

Система збереження даних реалізована через відокремлені контейнери СУБД PostgreSQL та Redis, що функціонують з використанням Persistent Volumes. Такий підхід гарантує збереженість накопиченої інформації, журналів інцидентів та еталонних профілів трафіку навіть у випадках примусового перезапуску сервісів або оновлення образів контейнерів. Представлена програмна архітектура дозволяє повноцінно реалізувати концепцію «безпека як сервіс» (SECaaS), забезпечуючи прозоре горизонтальне масштабування системи шляхом динамічного додавання нових контейнерів-сенсорів відповідно до зростання обсягів мережевого трафіку без переривання функціонування аналітичного ядра.

Програмна логіка агентів збору та фільтрації побудована на базі об'єктно-орієнтованої моделі, де ключовим елементом є клас-контролер, що координує процеси перехоплення трафіку та екстракції ознак. Функціонування агента реалізовано як безперервний асинхронний цикл моніторингу мережевого інтерфейсу, що забезпечує мінімальну затримку між надходженням пакета та його інтелектуальним аналізом.

Алгоритм роботи агента передбачає послідовне проходження даних через фільтраційні шари, що візуалізовано на рис. 4.5. Процес обробки ініціюється методом-диспетчером, який виконує первинну селекцію пакетів за протоколами та перевіряє їх на відповідність сигнатурним базам даних Snort 3. Якщо пакет не класифікується як відома загроза на рівні сигнатур, він передається до модуля екстракції ознак. На цьому етапі за допомогою інструментарію Scapy виконується розбір заголовків протоколів та формування багатовимірного вектора, що містить статистичні метрики: тривалість сесії, кількість байтів, прапорці TCP-з'єднання та інші специфічні параметри.



Рисунок 4.5 – Алгоритм функціонування агента збору та фільтрації даних

Критично важливим етапом підготовки даних для аналітичного ядра є їхня нормалізація. Оскільки ознаки мережевої активності мають різні порядки величин, в агенті реалізовано програмний модуль препроцесингу, який трансформує всі отримані значення в уніфікований діапазон. Це нівелює ризик зміщення моделі в сторону ознак з великими абсолютними значеннями та забезпечує стабільність роботи градієнтних методів оптимізації в межах LSTM-автоенкодера.

Підготовлений та нормалізований вектор ознак інтегрується у структуру часового вікна, що дозволяє системі аналізувати не лише окремі транзакції, а

й динаміку їхніх змін у часі. Транспортування готових наборів даних до аналітичного модуля реалізовано через неблокуючі HTTP-запити. Використання асинхронних клієнтів дозволяє агенту продовжувати захоплення пакетів паралельно з процесом очікування вердикту від нейромережі [98]. Така організація програмного коду мінімізує ймовірність втрати трафіку навіть в умовах різкого зростання навантаження на мережеву інфраструктуру хмарного сервісу.

Графічний інтерфейс користувача розробленої інтелектуальної системи виявлення аномалій реалізований як багатофункціональна веборієнтована консоль управління, що функціонує за моделлю «тонкого клієнта». Вибір вебтехнологій як базису для фронтенд-частини зумовлений імперативною вимогою забезпечення кросплатформного доступу адміністраторів безпеки до аналітичного інструментарію з будь-якого сегмента хмарної інфраструктури без необхідності інсталяції специфічного програмного забезпечення на кінцевих терміналах.

Основним технологічним стеком для розробки клієнтського рівня обрано фреймворк React, що дозволило впровадити компонентно-орієнтовану архітектуру та забезпечити високу швидкість рендерингу інтерфейсу за допомогою механізму Virtual DOM. Такий підхід мінімізує обчислювальне навантаження на сторону клієнта при відображенні великих масивів динамічних даних. Для візуалізації статистичних показників та результатів нейромережевого аналізу інтегровано спеціалізовану бібліотеку Chart.js, яка забезпечує апаратне прискорення при побудові інтерактивних графіків інтенсивності трафіку та розподілу помилок реконструкції.

Програмна реалізація інтерфейсу базується на принципах Single Page Application, що дозволяє реалізувати безшовну навігацію між аналітичними модулями та підтримувати стійке з'єднання з серверною частиною через протокол WebSockets. Це критично важливо для забезпечення оперативності моніторингу, оскільки дозволяє транслювати сповіщення про виявлені аномалії у режимі реального часу без

примусового оновлення сторінки браузера.

Крім того, при проектуванні GUI було враховано вимоги до ергономіки та когнітивного навантаження на оператора: впроваджено адаптивну верстку та систему кольорового кодування рівнів загрози, що корелює з результатами інференсу LSTM-автоенкодера [99]. Таким чином, розроблений інтерфейс виступає не лише як засіб візуалізації, а як інтегроване середовище прийняття рішень, що поєднує інструменти конфігурування нейромережових агентів, глибокого аналізу інцидентів та автоматизованої генерації звітності про стан захищеності мережевої інфраструктури.

Архітектурна побудова користувацького інтерфейсу детермінована модульним принципом, де кожна функціональна вкладка відповідає за детермінований етап життєвого циклу управління інформаційною безпекою. Такий підхід забезпечує логічну ізоляцію процесів моніторингу, аналізу та адміністрування, дозволяючи оператору системи фокусуватися на конкретних аналітичних завданнях. Структура інтерфейсу охоплює повний цикл обробки даних: від спостереження за первинними потоками «сирого» трафіку до проведення ретроспективного аналізу інцидентів та динамічної реконфігурації параметрів нейромережових моделей.

Центральним компонентом інтерфейсу управління розробленої системи є інтерактивна панель моніторингу, що функціонує як агрегатор ініціалізованої телеметрії в режимі реального часу. Програмна реалізація даного модуля базується на парадигмі реактивного програмування та принципах потокової візуалізації. Це дозволяє трансформувати детерміновані вектори ознак, отримані від розподілених агентів-сенсорів, у динамічні часові ряди з мінімальною латентністю.

Візуальна архітектура головної панелі (рис. 4.6) розроблена з урахуванням вимог до зручності сприйняття інформації та забезпечує комплексне відображення ключових показників стану мережевої інфраструктури. Центральним елементом інтерфейсу є модуль моніторингу інтенсивності мережевого трафіку, реалізований у вигляді інтерактивного

лінійного графіка, який відображає залежність між кількістю оброблених пакетів за одиницю часу та пропускнуою здатністю мережі. Це дає змогу своєчасно виявляти різкі зміни характеристик трафіку, що можуть свідчити про реалізацію атак типу DDoS або активне сканування мережевих ресурсів [100].

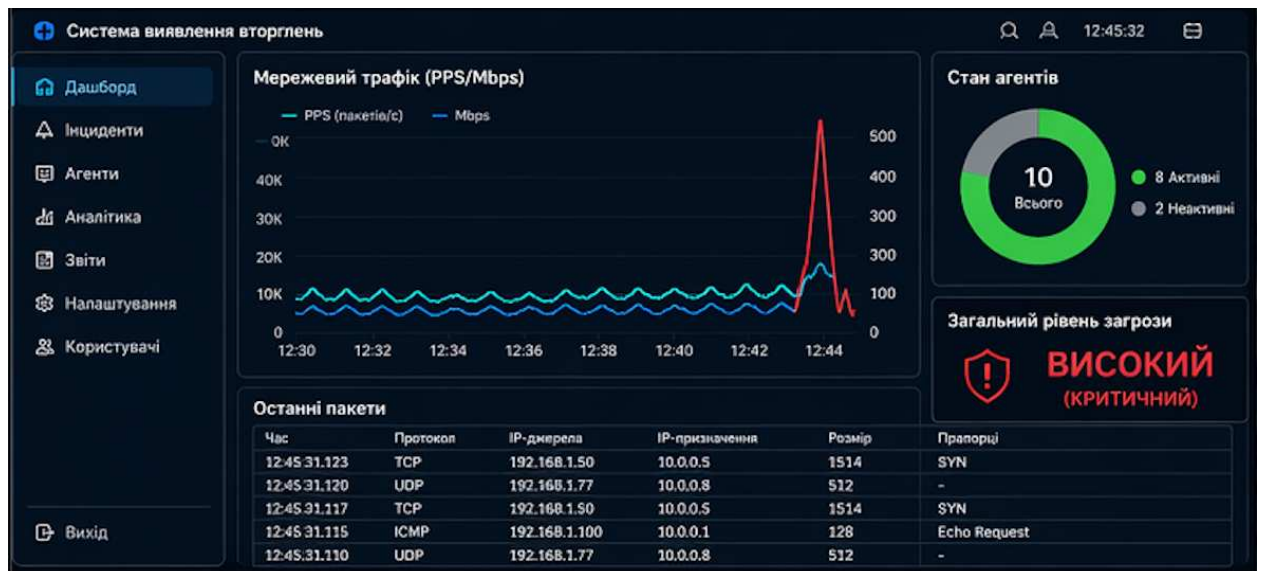


Рисунок 4.6 – Головна панель моніторингу інтелектуальної системи

Для контролю працездатності компонентів системи передбачено модуль відображення стану агентів, який наочно демонструє активність розгорнутих сенсорів у різних сегментах хмарної інфраструктури. Це дозволяє оперативно виявляти відмови окремих контейнеризованих агентів та контролювати цілісність системи моніторингу.

Важливим елементом головної панелі є індикатор рівня аномальності, який відображає середнє значення помилки реконструкції [101], отримане під час роботи нейромережевої моделі. На основі цього показника автоматично визначається інтегральний рівень загрози, який змінюється від «Низького» до «Критичного» та супроводжується відповідною кольоровою індикацією, що сприяє скороченню часу реагування адміністратора безпеки.

У нижній частині панелі розміщено модуль деталізації поточних подій, який відображає інформацію про останні оброблені мережеві пакети.

Табличне представлення містить відомості про використані протоколи, IP-адреси джерела та призначення, а також прапорці TCP-сесій, що забезпечує детальний аналіз мережевої активності та спрощує інтерпретацію виявлених аномалій.

Технологічно оновлення даних на Дашборді реалізовано через протокол WebSockets, що виключає необхідність надлишкових HTTP-запитів та забезпечує безперервну синхронізацію стану між сервером бази даних (PostgreSQL/Redis) та клієнтською частиною додатка. Такий підхід гарантує актуальність відображуваної інформації та високу чутливість системи до миттєвих змін у характері мережевої активності.

Для поглибленої верифікації виявлених загроз та проведення комплексних розслідувань у системі реалізовано модуль аналізу інцидентів. Даний компонент функціонує як високодеталізований аналітичний журнал, де кожен запис є результатом інференсу аналітичного ядра. На відміну від традиційних систем реєстрації подій, цей модуль інтегрує статистичні параметри нейромережевого аналізу з класичними мережевими атрибутами, що забезпечує багатовимірне представлення аномальної активності.

У табличній формі інтерфейсу (рис. 4.7) реалізовано засоби структурованого представлення результатів аналізу, що забезпечують швидку інтерпретацію виявлених подій. Для пріоритезації уваги адміністратора записи з високим рівнем аномальності автоматично виділяються кольоровою індикацією, що дозволяє оперативно ідентифікувати критичні інциденти та скоротити час реагування.

Фільтри		Час	IP-джерела	Тип атаки	Ймовірність (%)	MSE оцінка
Діапазон дат	СЗ	2024-05-20	-	2024-05-27		
Тип атаки	Всі					
Рівень загрози	Всі					
MSE оцінка	Від 0 До 0					
Застосувати						
Скинути						
		2024-05-27 12:44:59	192.168.1.50	DDoS	98.7	45.2
		2024-05-27 12:44:51	192.168.1.33	Port Scan	96.3	32.8
		2024-05-27 12:44:45	10.0.0.23	Brute Force	93.1	28.4
		2024-05-27 12:44:32	192.168.1.77	Suspicious Traffic	78.5	15.6
		2024-05-27 12:44:21	172.16.0.10	DDoS	97.2	41.7
		2024-05-27 12:44:10	192.168.1.88	Port Scan	88.4	22.1
		2024-05-27 12:44:03	10.0.0.15	SQL Injection	91.0	26.3
		2024-05-27 12:43:55	192.168.1.50	DDoS	98.5	44.1
		2024-05-27 12:43:41	172.16.0.25	Suspicious Traffic	70.2	12.7
		2024-05-27 12:43:30	192.168.1.99	Port Scan	88.6	19.8

Рисунок 4.7 – Інтерфейс аналізу виявлених інцидентів безпеки

Кожен запис містить набір основних параметрів, необхідних для аналізу мережевої події. Поле Timestamp фіксує точний час виявлення аномалії з мілісекундною точністю, що забезпечує можливість кореляції подій під час розслідування складних або розподілених атак. Поле мережевої ідентифікації джерела відображає IP-адресу вузла, з якого зафіксовано підозрілу активність, а також підтримує швидку фільтрацію записів для виявлення повторюваних спроб вторгнення.

Тип виявленої аномалії визначається автоматично на основі результатів аналізу мережевого трафіку та його зіставлення з відомими шаблонами атак, такими як DDoS, Port Scan або SQL Injection. Якщо характер мережевої активності не відповідає жодному з відомих шаблонів, події присвоюється статус Suspicious Traffic, що свідчить про виявлення раніше невідомої або нетипової поведінки мережевого трафіку.

Додатково для кожного інциденту відображається оцінка ймовірності, яка характеризує рівень впевненості LSTM-моделі у правильності класифікації загрози та подається у відсотках. Одним із ключових аналітичних показників є значення MSE, що відображає величину помилки реконструкції, отриманої автоенкодером. Цей показник дозволяє кількісно оцінити ступінь відхилення поточної мережевої активності від сформованого профілю

нормальної поведінки: зі збільшенням значення MSE зростає ймовірність того, що зафіксована подія є проявом аномалії або порушення політики інформаційної безпеки.

Функціонал модуля також включає систему динамічних фільтрів (ліва панель на рис. 4.7), що дозволяє звужувати область аналізу за діапазоном дат, рівнем загрози або конкретними пороговими значеннями MSE. Це забезпечує можливість проведення цілеспрямованого аудиту безпеки та виявлення прихованих довготривалих атак, які можуть маскуватися під низькорівневий шум у загальному потоці трафіку.

Для забезпечення високого ступеня адаптивності та операційної гнучкості інтелектуальної системи виявлення аномалій розроблено спеціалізований модуль децентралізованого управління сенсорною мережею. Даний компонент графічного інтерфейсу дозволяє адміністратору безпеки здійснювати прецизійний віддалений контроль над життєвим циклом кожного Docker-контейнера, що функціонує як незалежний агент збору та первинної обробки трафіку. Програмна архітектура модуля базується на принципах оркестрації мікросервісів, що забезпечує прозору взаємодію між центральною консоллю управління та розподіленими обчислювальними вузлами.

Візуальна репрезентація підсистеми керування агентами (рис. 4.8) реалізована у вигляді інтерактивної панелі. Інтерфейс надає адміністратору можливість у режимі реального часу керувати активністю вузлів, виконуючи активацію або деактивацію моніторингу в окремих сегментах мережі. Це дозволяє адаптувати роботу системи до поточної конфігурації хмарної інфраструктури та перерозподіляти обчислювальні ресурси відповідно до критичності окремих мережових сегментів [102].

Для кожного агента відображаються актуальні показники використання обчислювальних ресурсів, зокрема рівень завантаження центрального процесора та обсяг використаної оперативної пам'яті. Моніторинг цих параметрів дає змогу своєчасно виявляти потенційні вузькі місця та запобігати деградації роботи системи в умовах високого навантаження.

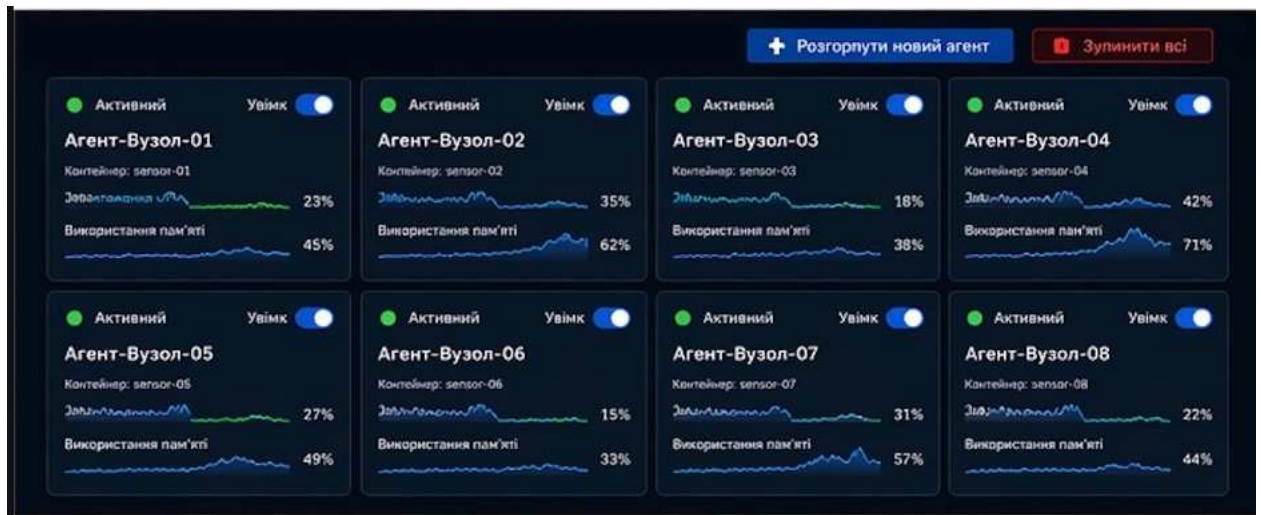


Рисунок 4.8 – Модуль управління активністю інтелектуальних агентів

Інтерфейс також підтримує централізоване керування агентами, забезпечуючи можливість швидкого розгортання нових сенсорних вузлів і групового керування вже розгорнутими агентами. Такий підхід спрощує адміністрування великомасштабної хмарної інфраструктури та скорочує час виконання типових операцій.

Для оперативного контролю працездатності системи передбачено статусну індикацію, яка відображає поточний стан кожного агента. Це дозволяє швидко перевіряти цілісність системи моніторингу, своєчасно виявляти відмови окремих агентів і реагувати на можливі порушення функціонування мікросервісної інфраструктури.

Така програмна реалізація інтерфейсу управління дозволяє повноцінно втілити стратегію оптимізації хмарних ресурсів. Шляхом вибіркового відключення моніторингу в сегментах з низькою пріоритетністю або тимчасового нарощування аналітичних потужностей на критичних напрямках досягається максимальна ефективність системи при мінімальних інфраструктурних витратах. Інтеграція модуля керування з центральним API-сервером через протокол HTTPS забезпечує захищеність команд управління та цілісність переданої телеметричної інформації.

Однією з найбільш репрезентативних особливостей розробленого графічного інтерфейсу є спеціалізована сторінка конфігурації аналітичного

ядра. Даний модуль реалізує механізм адаптивного управління чутливістю інтелектуальної системи, що дозволяє фахівцю з інформаційної безпеки здійснювати точне налаштування математичної моделі LSTM-автоенкодера відповідно до специфіки мережевого оточення.

Центральним елементом сторінки налаштування параметрів моделі (рис 4.9) є гістограма розподілу значень MSE, яка відображає розподіл нормального та аномального мережевого трафіку в логарифмічному масштабі. За допомогою інтерактивного слайдера адміністратор може змінювати порогове значення MSE, після чого вертикальна лінія на гістограмі автоматично переміщується, розділяючи області нормального та аномального трафіку. Це дозволяє візуально оцінити ступінь розділення розподілів і підібрати поріг, який забезпечує оптимальний баланс між виявленням аномалій та кількістю помилкових спрацьовувань.

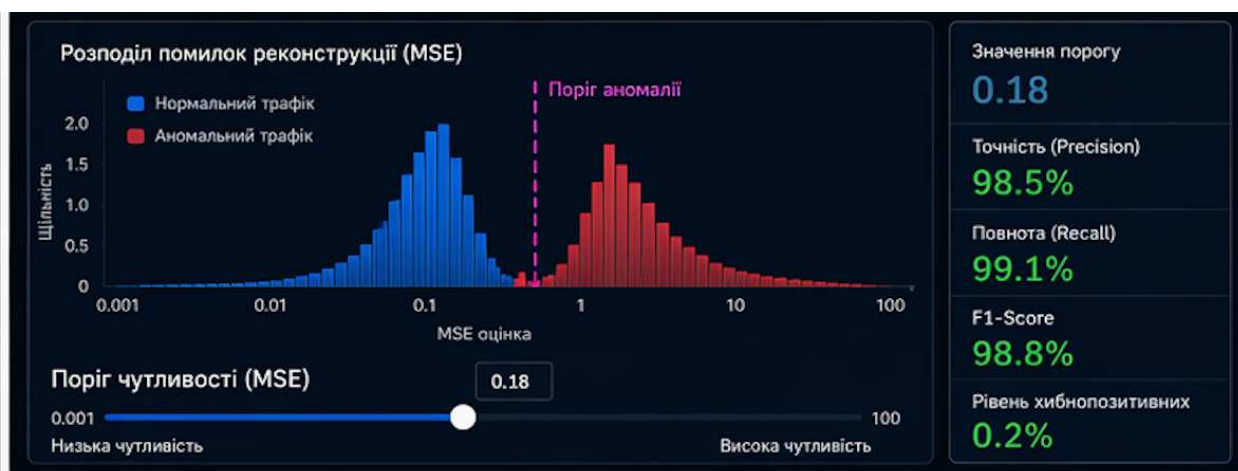


Рисунок 4.9 – Інтерфейс налаштування порогу чутливості нейромережі

У правій частині інтерфейсу реалізовано блок оцінювання ефективності моделі, який автоматично оновлюється після зміни порогового значення. Для кожного обраного порога система обчислює показники Precision, що характеризує частку правильно виявлених атак серед усіх зафіксованих аномалій, Recall, який відображає здатність моделі виявляти реальні загрози, та F1-score, що є інтегральною характеристикою якості класифікації на основі

точності й повноти. Додатково відображається рівень хибнопозитивних спрацьовувань, який показує частку легітимного мережевого трафіку, помилково класифікованого як аномальний. Це дозволяє оцінити вплив зміни порогового значення на ефективність роботи системи та підібрати параметри, що забезпечують найкращу якість виявлення аномалій.

Такий підхід до реалізації інтерфейсу дозволяє оператору вручну балансувати між жорсткістю політики безпеки та операційною стабільністю мережі. Наприклад, у періоди високої ймовірності цілеспрямованих атак адміністратор може знизити поріг для максимального охоплення підозрілих подій, а в штатному режимі – підвищити його для усунення надлишкових сповіщень. Описаний інструментарій візуалізації «матриці помилок» та динамічного аналізу порогів перетворює нейромережеву модель із «чорної скриньки» на прозорий аналітичний інструмент з керованою логікою прийняття рішень.

Висока детермінованість та швидкість реакції користувацького інтерфейсу на критичні події безпеки досягається завдяки впровадженню асинхронної моделі обміну даними. На відміну від традиційних вебдодатків, що базуються на періодичних запитах клієнта, розроблена фронтенд-частина системи використовує повнодуплексний протокол WebSockets. Це дозволяє встановити перманентне TCP-з'єднання між браузером адміністратора та бекенд-сервером, забезпечуючи миттєву трансляцію подій за ініціативою сервера.

У контексті функціонування інтелектуальної системи, як тільки аналітичне ядро на базі LSTM-автоенкодера фіксує перевищення порогу аномальності, сформований пакет даних про інцидент через брокер повідомлень передається у WebSocket-канал. Це гарантує, що критичне сповіщення з'явиться на консолі моніторингу з мілісекундною затримкою, що є вирішальним фактором для мінімізації показника Mean Time to Respond при відбитті цілеспрямованих атак.

Логіка обробки та синхронізації даних на стороні клієнта базується на централізованому управлінні станом і гібридній моделі збору даних. Для підтримки консистентності даних у масштабах усього SPA-додатка використано бібліотеку Redux, у глобальному сховищі якої зберігається вся телеметрія, що надходить від розподілених агентів. Це дозволяє миттєво оновлювати компоненти інтерфейсу, зокрема графіки, таблиці логів і статуси вузлів, без повторних запитів до бази даних, що значно знижує навантаження на мережеву інфраструктуру та обчислювальні ресурси клієнтського термінала. Поряд із потоковою передачею даних через WebSocket, у системі реалізовано резервний механізм інтелектуального опитування API, який використовується для завантаження ретроспективної інформації під час ініціалізації сесії або після відновлення з'єднання. Це забезпечує безперервність побудови графіків історичної активності та дозволяє проводити порівняльний аналіз поточного стану мережі з еталонними періодами за минулу добу або тиждень.

Програмна імплементація інтерфейсу також передбачає використання технології Web Workers для виконання важких обчислювальних операцій (наприклад, первинної фільтрації великих масивів логів перед рендерингом) в окремому потоці. Це запобігає блокуванню основного потоку виконання та забезпечує плавність роботи інтерфейсу навіть при інтенсивному надходженні даних під час масштабних мережевих інцидентів.

Ефективність обраної архітектурної моделі підтверджується здатністю системи стабільно візуалізувати потоки даних з інтенсивністю понад 1000 подій на секунду, що повністю задовольняє потреби моніторингу сучасних хмарних середовищ.

Важливою складовою інтерфейсу користувача є модуль генерації звітів, який виступає інструментом для трансформації великих масивів неструктурованих даних, накопичених у реляційній базі даних PostgreSQL, у формалізовані аналітичні документи. Програмна реалізація модуля на рисунку 4.10 побудована на принципах декларативного формування запитів,

що дозволяє користувачу гнучко налаштовувати параметри вибірки перед фінальним рендерингом документа.

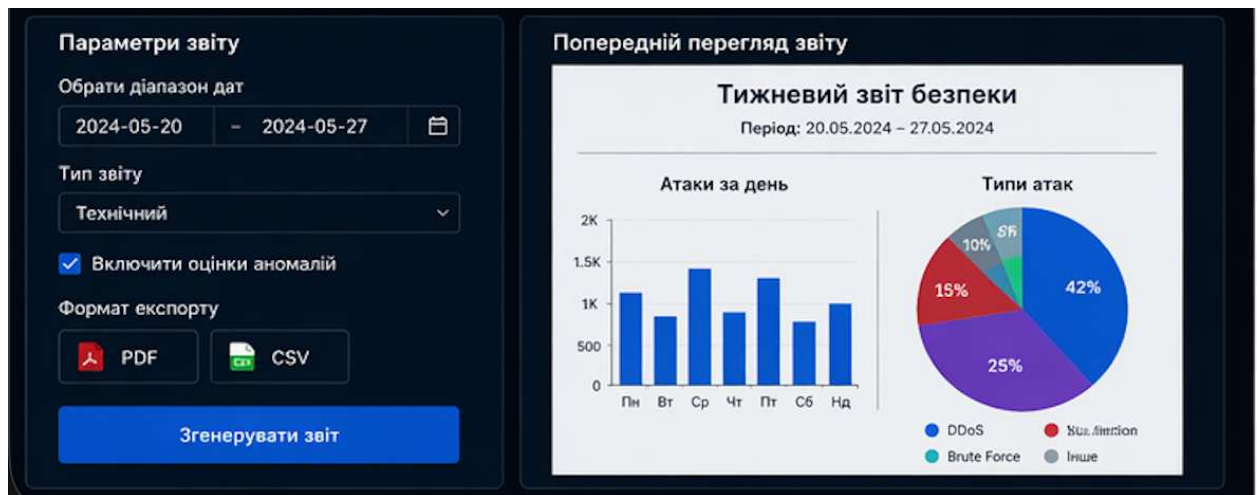


Рисунок 4.10 – Інтерфейс модуля формування аналітичних звітів

Функціональні можливості підсистеми звітування охоплюють динамічну фільтрацію даних, формування звітів різного призначення та їх експорт у декількох форматах. Алгоритми модуля дозволяють агрегувати статистику за довільні часові інтервали – від годинних зрізів до квартальних звітів, що забезпечує можливість аналізу довгострокових тенденцій мережевої активності та оцінювання ефективності впроваджених політик безпеки.

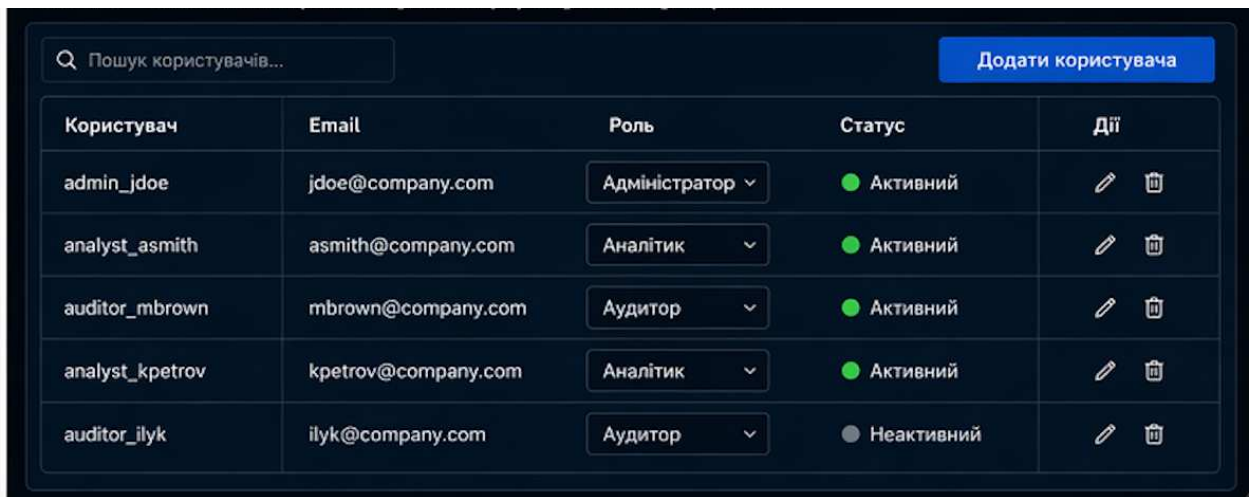
Залежно від потреб користувача система підтримує формування двох типів звітів. Технічний звіт містить детальну інформацію про кожен інцидент, включаючи дампи пакетів, значення MSE та результати класифікації, отримані LSTM-автоенкодером, і призначений для системних адміністраторів та фахівців з інформаційної безпеки. Консолідований звіт орієнтований на узагальнене представлення результатів аналізу та містить гістограми розподілу атак за днями тижня, кругові діаграми співвідношення різних типів загроз, а також загальні показники стабільності функціонування системи.

Програмний код модуля звітування інтегрований із аналітичним ядром, що дозволяє включати у звітність не лише «сухі» факти спрацювань, а й динаміку зміни точності моделі за обраний період. Це надає адміністратору

обґрунтовану базу для прийняття рішень щодо необхідності перенавчання нейромережі або корекції порогів чутливості, описаних у попередніх підрозділах. Таким чином, модуль звітування замикає цикл управління безпекою, забезпечуючи перехід від оперативного реагування до стратегічного планування захисту мережевої інфраструктури.

З огляду на високу критичність функцій управління мережевою безпекою та необхідність запобігання несанкціонованим змінам у конфігурації нейромережі, до складу графічного інтерфейсу інтегровано модуль автентифікації та багатофакторної авторизації. Програмна реалізація системи контролю доступу базується на моделі RBAC, що дозволяє диференціювати функціональні можливості інтерфейсу залежно від службових обов'язків персоналу.

Візуальний модуль керування користувачами на рисунку 4.11 реалізовано у вигляді адміністративної панелі, яка забезпечує централізоване керування обліковими записами, їх ролями та контроль активності користувачів. У системі реалізовано три ієрархічні рівні доступу, кожен з яких має визначений набір функціональних можливостей. Роль Administrator надає повний доступ до конфігурації системи, включаючи зміну параметрів нейромережевої моделі, керування життєвим циклом Docker-контейнерів, видалення журналів подій, а також створення, редагування та видалення облікових записів користувачів. Роль Security Analyst призначена для повсякденної експлуатації системи та забезпечує доступ до засобів моніторингу в реальному часі, аналізу інцидентів і формування звітів. Додатково користувачі цієї ролі можуть позначати виявлені аномалії як хибнопозитивні спрацьовування, що використовується для формування зворотного зв'язку та подальшого донавчання LSTM-автоенкодера, підвищуючи його адаптивність до особливостей мережевого трафіку конкретної інфраструктури. Роль Auditor працює в режимі «тільки читання» та призначена для перегляду поточного стану системи і сформованих звітів без можливості змінювати параметри роботи агентів.



Користувач	Email	Роль	Статус	Дії
admin_jdoe	jdoe@company.com	Адміністратор	Активний	 
analyst_asmith	asmith@company.com	Аналітик	Активний	 
auditor_mbrown	mbrown@company.com	Аудитор	Активний	 
analyst_kpetrov	kpetrov@company.com	Аналітик	Активний	 
auditor_ilyk	ilyk@company.com	Аудитор	Неактивний	 

Рисунок 4.11 – Вікно налаштування прав доступу та ролей користувачів

Програмний інтерфейс реалізований у вигляді структурованого реєстру з підтримкою інтерактивного пошуку та динамічної зміни статусів користувачів («Активний» / «Неактивний»). Використання кольорової індикації статусів та випадаючих списків для призначення ролей мінімізує ймовірність помилок при адмініструванні доступу.

З точки зору безпеки передачі даних, авторизація реалізована за допомогою механізму JSON Web Tokens (JWT). Це гарантує, що кожен запит до API від клієнтської частини супроводжується захищеним токеном, який верифікує права користувача на виконання конкретної дії. Такий підхід унеможливорює обхід інтерфейсних обмежень та забезпечує цілісність системи навіть у випадку спроб компрометації окремих сесій.

Розроблений графічний інтерфейс користувача також забезпечує практичну реалізацію моделі мультиагентної інтелектуальної системи виявлення аномалій у хмарних середовищах. Через єдиний центр керування здійснюється взаємодія агентів збору та аналізу даних, що виконують моніторинг телеметрії Kubernetes-кластерів, мережних подій, журналів API-викликів та поведінкової активності користувачів. Інтерфейс забезпечує централізовану візуалізацію результатів аналізу, керування процесами виявлення аномалій, перегляд сповіщень про інциденти безпеки та контроль стану окремих агентів системи.

Таким чином, розроблений GUI виступає не лише засобом відображення результатів, а й інтеграційним компонентом мультиагентної архітектури, що забезпечує узгоджену роботу підсистем збору, обробки та аналізу даних у межах єдиного процесу моніторингу безпеки хмарного середовища. Завершення опису даного модуля підводить підсумок розробці GUI як комплексного середовища, що об'єднує інтелектуальний аналіз, оперативне управління, багаторівневий моніторинг та суворий контроль доступу, що є обов'язковою вимогою для сучасних хмарних систем виявлення аномалій та вторгнень.

Висновки до розділу 4

У четвертому розділі дисертаційної роботи розроблено та програмно реалізовано мультиагентну модель виявлення аномалій у хмарних сервісах, орієнтовану на забезпечення своєчасного виявлення кіберінцидентів у динамічному розподіленому середовищі. Запропонований метод базується на інтеграції сигнатурного аналізу відомих загроз із поведінковим моделюванням на основі глибокого навчання, що дозволило поєднати детермінованість класичних систем захисту із гнучкістю інтелектуальних алгоритмів виявлення аномалій нульового дня. Програмна реалізація системи забезпечує фіксацію критичних відхилень від базових профілів функціонування, що ідентифікуються через аналіз помилки реконструкції нейромережевих моделей.

У ході дослідження формалізовано архітектуру взаємодії двох взаємодоповнювальних базових моделей: індивідуального профілю поведінки користувачів та інтегрального профілю функціонування хмарної інфраструктури в цілому. Для верифікації методу було сформовано репрезентативний датасет нормальної роботи системи, що відображає типові сценарії експлуатації обчислювальних ресурсів, що дозволило врахувати специфіку цільового середовища та значно знизити рівень хибнопозитивних

спрацювань. Реалізований технологічний стек забезпечує автоматизований збір даних, їх попередню обробку та формування простору ознак, необхідного для функціонування аналітичного ядра на базі LSTM-автоенкодера.

Особливістю запропонованого рішення є багаторівнева мультиагентна структура, де функції системи розподілені між інтелектуальними агентами, відповідальними за збір телеметрії, аналіз пакетів, формування сповіщень та адаптивне оновлення бази знань. Програмна імплементація даної архітектури у вигляді мікросервісів на базі Docker-контейнерів забезпечує високу масштабованість та відмовостійкість системи, дозволяючи інтегрувати її в існуючі контури моніторингу хмарної інфраструктури без деградації її продуктивності.

Сформовані результати четвертого розділу підтверджують ефективність впровадження розподілених інтелектуальних систем для захисту хмарних сервісів. Розроблений графічний інтерфейс управління та механізми адаптивного налаштування порогів чутливості створюють цілісне середовище для підтримки прийняття рішень адміністраторами безпеки. Це відкриває перспективи для подальшого вдосконалення методів автоматизованого реагування на інциденти та створення систем самовідновлюваного цифрового захисту в гетерогенних мережевих середовищах.

ВИСНОВКИ

У даній дисертаційній роботі розв'язано актуальну науково-прикладну задачу підвищення рівня безпеки хмарних сервісів в умовах динамічної мікросервісної архітектури шляхом розробки моделі та методів інтелектуальної інформаційної технології виявлення аномалій на основі глибокого навчання та мультиагентного підходу. У ході дослідження отримано такі наукові та практичні результати:

1. Удосконалено метод виявлення аномальної поведінки користувачів вебресурсів шляхом формування індивідуального поведінкового профілю на основі інтеграції часових характеристик сесій, показників активності та послідовностей переходів між вебресурсами. На відміну від існуючих підходів, метод забезпечує комплексне врахування часових, кількісних і послідовнісних ознак під час побудови профілю нормальної поведінки та оцінювання аномальності, що дозволило підвищити повноту виявлення аномалій на 11,0% та F1-міру на 11,4% порівняно з методом One-Class SVM.

2. Удосконалено метод виявлення аномалій у Kubernetes-кластерах шляхом інтеграції мережових характеристик, метрик контейнерів та показників стану кластера в єдині часові послідовності ознак для формування профілю нормального функціонування контейнеризованого середовища. На відміну від існуючих підходів, метод забезпечує комплексне врахування взаємозв'язків між різнорідними джерелами телеметричних даних та їх часової динаміки під час оцінювання аномальності, що дозволило підвищити якість виявлення аномальної активності та забезпечити виявлення раніше невідомих атак у динамічних середовищах Kubernetes.

3. Отримала подальший розвиток модель мультиагентної інтелектуальної системи виявлення аномалій у хмарних середовищах, яка відрізняється інтеграцією агентів збору мережових подій, телеметрії Kubernetes, API-журналів та поведінкових даних користувачів у межах

єдиного процесу моніторингу. На відміну від централізованого сигнатурного моніторингу на базі Snort, запропонована модель забезпечує багаторівневий аналіз подій безпеки та дозволяє підвищити повноту виявлення кіберзагроз на 18%, зменшити кількість хибнопозитивних спрацювань на 23% і скоротити час виявлення інцидентів на 29%.

4. розроблено програмну архітектуру інтелектуальної інформаційної технології на базі мікросервісної структури та Docker-контейнерів, яка включає модулі збору та обробки API-журналів, мережесих подій, телеметрії Kubernetes-кластерів і поведінкових даних користувачів, нейромережевого аналізу, візуалізації та управління доступом, що забезпечує масштабованість, відмовостійкість і можливість розгортання у хмарному контейнеризованому середовищі;

5. проведено експериментальну верифікацію запропонованих моделі та програмної архітектури на реальних потоках трафіку, API-журналах і системних метриках Kubernetes-кластерів, що підтвердила високу ефективність виявлення аномалій та здатність системи функціонувати в умовах високої волатильності навантаження без деградації продуктивності сервісів.

Результати дисертаційної роботи підтверджені експериментальними дослідженнями та узгоджуються з теоретичними положеннями сучасної кібербезпеки. Розроблені модель та методи доводять доцільність використання інтелектуальних технологій глибокого навчання для підтримки прийняття рішень у сфері захисту хмарних інфраструктур і становлять науково-практичну основу для впровадження адаптивних систем виявлення аномалій у сучасних дата-центрах та ІТ-компаніях.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Мартовицький В. В., Свиридов А. С., Авдєєв О. В., Гудзинський І. М., Коротецький О. В. Дослідження методів виявлення аномалій у API журналах для забезпечення безпеки та надійності програмних систем // Вісник Херсонського національного технічного університету. 2025. Т. 2, № 1(92). С. 142–148. DOI: <https://doi.org/10.35546/kntu2078-4481.2025.1.2.19>
2. Свиридов А. С., Гусейнов Р. Б., Винниченко С. М. Метод виявлення аномалій у поведінці користувача вебсайту // Herald of Khmelnytskyi National University. Technical Sciences. 2026. Т. 363, № 2. С. 211–219. DOI: <https://doi.org/10.31891/2307-5732-2026-363-28>.
3. Свиридов А. С., Северінов О. В. Метод виявлення аномалій у Kubernetes-кластерах з використанням LSTM Autoencoder // Herald of Khmelnytskyi National University. Technical Sciences. 2026. Т. 361, № 1. С. 501–509. DOI: <https://doi.org/10.31891/2307-5732-2026-361-70>.
4. Свиридов А. С., Гудзинський І. М. Архітектура мультиагентної системи виявлення вторгнень із використанням машинного навчання // Вісник Херсонського національного технічного університету. 2026. Т. 1, № 2(97). С. 360–366. DOI: <https://doi.org/10.35546/kntu2078-4481.2026.2.44>.
5. Kuchuk G., Kovalenko A., Elyasi Komari I., Svyrydov A., Kharchenko V. Improving Big Data Centers Energy Efficiency: Traffic Based Model and Method // Advances in Computer Science for Engineering and Education II. Cham : Springer, 2019. P. 77–88. DOI: 10.1007/978-3-030-00253-4_8.
6. Северінов О. В., Свиридов А. С. Виявлення аномалій у API журналах для підвищення безпеки програмних систем // Сучасні напрями розвитку інформаційно-комунікаційних технологій та засобів управління : тези доповідей тринадцятої міжнародної науково-технічної конференції, 27–28 листопада 2025 р. Харків, 2025. Т. 3, секц. 4. С. 35.
7. Свиридов А. С. Метод виявлення аномалій у поведінці користувачів

вебсайтів // Science and education: synergy of innovation : Proceedings of the 8th International scientific and practical conference, Berlin, Germany, 23–25 March 2026. Berlin : MDPC Publishing, 2026. P. 198. URL: <https://sci-conf.com.ua/viii-mizhnarodna-naukovo-praktichna-konferentsiya-science-and-education-synergy-of-innovation-23-25-03-2026-berlin-nimechchina-arhiv/>.

8. Свиридов А. С. Архітектура мультиагентної системи виявлення вторгнень із використанням машинного навчання // Scientific development in a changing world : Proceedings of the 3rd International scientific and practical conference, Lviv, Ukraine, 16–18 March 2026. Львів : SPC “Sci-conf.com.ua”, 2026. P. 330. URL: <https://sci-conf.com.ua/iii-mizhnarodna-naukovo-praktichna-konferentsiya-scientific-development-in-a-changing-world-16-18-03-2026-lviv-ukrayina-arhiv/>.

9. Свиридов А. С. Метод виявлення аномалій у Kubernetes-кластерах з використанням LSTM Autoencoder // European science and innovation congress : Proceedings of the 4th International scientific and practical conference, Barcelona, Spain, 9–11 March 2026. Barcelona : Barca Academy Publishing, 2026. P. 169. URL: <https://sci-conf.com.ua/iv-mizhnarodna-naukovo-praktichna-konferentsiya-european-science-and-innovation-congress-9-11-03-2026-barselona-ispaniya-arhiv/>.

10. Свиридов А. С., Сєверінов О. В. Дослідження методів виявлення аномалій у API журналах для забезпечення безпеки та надійності програмних систем // Сучасні проблеми інфокомунікацій, радіоелектроніки та наносистем : тези доповідей шістнадцятої міжнародної науково-технічної конференції, 29–30 квітня 2026 р. Харків, 2026. Т. 3, секц. 4. С. 112–113.

11. Duraisamy K. R. Cloud Computing: A Study on Architecture, Types, Benefits and Challenges. International Journal on Science and Technology (IJSAT). 2025. Vol. 16, Iss. 3. P. 1–12. DOI: <https://doi.org/10.71097/IJSAT.v16.i3.7001>

12. Xiaojie Zhang and Saptarshi Debroy. 2023. Resource Management in Mobile Edge Computing: A Comprehensive Survey. ACM Comput. Surv. 55, 13s, Article 291 (December 2023), 37 pages. <https://doi.org/10.1145/3589639>

13. Sharma A. et al. Cloud Computing 2025 and Beyond: Trends, Obstacles, and New Possibilities. Preprints.org. 2025. DOI: 10.20944/preprints202503.0724.v1
14. Cloud Native Computing Foundation. CNCF Annual Survey 2025: The State of Cloud Native and Kubernetes. CNCF Reports. 2025. 42 p. URL: <https://www.cncf.io/reports/the-cncf-annual-cloud-native-survey> (дата звернення: 23.03.2026).
15. G. A. Wellbrock et al., "Explore Benefits of Distributed Fiber Optic Sensing for Optical Network Service Providers," in *Journal of Lightwave Technology*, vol. 41, no. 12, pp. 3758-3766, 15 June 2023, doi: 10.1109/JLT.2023.3263795.
16. B. Sharma and D. Nadig, "eBPF-Enhanced Complete Observability Solution for Cloud-native Microservices," ICC 2024 – IEEE International Conference on Communications, Denver, CO, USA, 2024, pp. 1980-1985, doi: 10.1109/ICC51166.2024.10622329.
17. Wang , S., Zheng , H., Wen , X. ., & Fu , S. (2024). DISTRIBUTED HIGH-PERFORMANCE COMPUTING METHODS FOR ACCELERATING DEEP LEARNING TRAINING. *Journal of Knowledge Learning and Science Technology* ISSN: 2959-6386 (online), 3(3), 108-126. <https://doi.org/10.60087/jklst.v3.n3.p108-126>
18. Huang, Shih-Yun, Cheng-Yu Chen, Jen-Yeu Chen, and Han-Chieh Chao. 2023. "A Survey on Resource Management for Cloud Native Mobile Computing: Opportunities and Challenges" *Symmetry* 15, no. 2: 538. <https://doi.org/10.3390/sym15020538>.
19. Atalay, T. O., Famili, A., Ghafoori, A., & Stavrou, A. (2026). A Survey on Cloud-Based 6G Deployments: Current Solutions, Future Directions and Open Challenges. <https://doi.org/10.48550/arXiv.2603.09894>
20. Sabzaliyev, A., & Asgarov, A. (2025). Transforming Communication and Industry: A Deep Dive into 5G Infrastructure and Applications. *Porta Universorum*, 1(3), 135-146. <https://doi.org/10.69760/portuni.010313>
21. Shamsul Arifeen, & Md. Morshedul Islam. (2025). The Role of Cloud-

Native Infrastructures in Supporting Autonomous and Uncrewed Systems (UXS) in Operations. *Journal of Sustainable Development and Policy*, 4(03), 82-125. <https://doi.org/10.63125/vntbqq40>.

22. T. Wijesekera, L. Jayarathna, G. Pathirana, R. Banu and J. Wickramaratne, "Kube5GC: Kubernetes-Native Orchestration for 5G Core Network," 2025 IEEE 13th International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS), Gliwice, Poland, 2025, pp. 1-10, doi: 10.1109/IDAACS68557.2025.11322314.

23. Pomeliuko, Denys & Yeromina, Nataliia & Petrov, Serhii & Ishchenko, Nataliia & Voloshchuk, Olena. (2025). МЕТОДИ МАСШТАБУВАННЯ KUBERNETES КЛАСТЕРУ В ХМАРНОМУ СЕРЕДОВИЩІ. Системи управління, навігації та зв'язку. Збірник наукових праць. 2. 175-179. doi: 10.26906/SUNZ.2025.2.175.

24. Ali, D., & Sofia, R. C. (2025). Experimenting with Energy-Awareness in Edge-Cloud Containerized Application Orchestration. DOI: 10.1016/j.suscom.2025.100912

25. Mahavaishnavi, V., Saminathan, R. & Prithviraj, R. Secure container Orchestration: A framework for detecting and mitigating Orchestrator – level vulnerabilities. *Multimed Tools Appl* 84, 18351–18371 (2025). <https://doi.org/10.1007/s11042-024-19613-x>

26. Alhaidari, F., Rahman, A. & Zagrouba, R. Cloud of Things: architecture, applications and challenges. *J Ambient Intell Human Comput* 14, 5957–5975 (2023). <https://doi.org/10.1007/s12652-020-02448-3>.

27. Xiaojie Wang, Jiameng Li, Zhaolong Ning, Qingyang Song, Lei Guo, Song Guo, and Mohammad S. Obaidat. 2023. Wireless Powered Mobile Edge Computing Networks: A Survey. *ACM Comput. Surv.* 55, 13s, Article 263 (December 2023), 37 pages. <https://doi.org/10.1145/3579992>

28. A. -I. Tuns and A. Spătaru, "Cloud Service Failure Prediction on Google's Borg Cluster Traces Using Traditional Machine Learning," 2023 25th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing

(SYNASC), Nancy, France, 2023, pp. 162-169, doi: 10.1109/SYNASC61333.2023.00029.

29. Mathur, P. (2024). Cloud Computing Infrastructure, Platforms, and Software for Scientific Research. In: Ahmad, K.A., Hamid, N.A.W.A., Jawaid, M., Khan, T., Singh, B. (eds) High Performance Computing in Biomimetics. Series in BioEngineering. Springer, Singapore. https://doi.org/10.1007/978-981-97-1017-1_4

30. Y. Li, Q. Zhang, H. Yao, R. Gao, X. Xin and M. Guizani, "Next-Gen Service Function Chain Deployment: Combining Multi-Objective Optimization With AI Large Language Models," in IEEE Network, vol. 39, no. 3, pp. 20-28, May 2025, doi: 10.1109/MNET.2025.3532212.

31. Zaritto, F., Bravi, E., Sisinni, S. et al. Extending Kubernetes for Pods Integrity Verification. J Netw Syst Manage 34, 14 (2026). <https://doi.org/10.1007/s10922-025-09988-z>

32. B. Kim, J. Kim and S. Lee, "Exploring Security Enhancements in Kubernetes CNI: A Deep Dive Into Network Policies," in IEEE Access, vol. 13, pp. 35322-35338, 2025, doi: 10.1109/ACCESS.2025.3543841

33. Rui Queiroz, Tiago Cruz, Jérôme Mendes, Pedro Sousa, and Paulo Simões. 2023. Container-based Virtualization for Real-time Industrial Systems—A Systematic Review. ACM Comput. Surv. 56, 3, Article 59 (March 2024), 38 pages. <https://doi.org/10.1145/3617591>

34. Ugwueze, V. U. (2024). Cloud native application development: Best practices and challenges. International Journal of Research Publication and Reviews, 5(12), 2399-2412. <https://doi.org/10.55248/gengpi.5.1224.3533>

35. Borra, Praveen, Securing Cloud Infrastructure: An In-Depth Analysis of Microsoft Azure Security (June 13, 2024). International Journal of Advanced Research in Science, Communication and Technology (IJARSCT) Volume 4, Issue 2, June 2024, <http://dx.doi.org/10.2139/ssrn.4914157>

36. Jiang, Y., Meng, Q., Shang, F., Oo, N., Minh, L. T. H., Lim, H. W., & Sikdar, B. (2025). MITRE ATT&CK applications in cybersecurity and the way

forward. arXiv preprint arXiv:2502.10825.
<https://doi.org/10.48550/arXiv.2502.10825>.

37. Cloud-Native Security and Observability Frameworks for Modern Digital Transformation Initiatives. (2025). International Journal of Computer Technology and Electronics Communication, 8(5), 11543-11553. <https://doi.org/10.15680/IJCTECE.2025.0805032>

38. V. Moghiss and A. Shameli-Sendi, "I-RECON: An IoT-Based Search Engine for Internet-Facing Services Vulnerability Reconnaissance," in IEEE Access, vol. 12, pp. 96100-96112, 2024, doi: 10.1109/ACCESS.2024.3425062

39. Dommari, Sandeep and Khan, Shakeb, Implementing Zero Trust Architecture in Cloud-Native Environments: Challenges and Best Practices (August 10, 2023). <http://dx.doi.org/10.2139/ssrn.5259339>

40. Omar Jarkas, Ryan Ko, Naipeng Dong, and Redowan Mahmud. 2025. A Container Security Survey: Exploits, Attacks, and Defenses. ACM Comput. Surv. 57, 7, Article 170 (July 2025), 36 pages. <https://doi.org/10.1145/3715001>

41. Noah Spahn, Nils Hanke, Thorsten Holz, Christopher Kruegel, and Giovanni Vigna. 2023. Container Orchestration Honey-pot: Observing Attacks in the Wild. In Proceedings of the 26th International Symposium on Research in Attacks, Intrusions and Defenses (RAID '23). Association for Computing Machinery, New York, NY, USA, 381–396. <https://doi.org/10.1145/3607199.3607205>

42. Matteo Franzil, Valentino Armani, Luis Augusto Dias Knob, Domenico Siracusa, Sharpening Kubernetes audit logs with context awareness, Computer Networks, Volume 276, 2026, 111890, ISSN 1389-1286, <https://doi.org/10.1016/j.comnet.2025.111890>.

43. Theodoropoulos, Theodoros, Luis Rosa, Chafika Benzaid, Peter Gray, Eduard Marin, Antonios Makris, Luis Cordeiro, Ferran Diego, Pavel Sorokin, Marco Di Girolamo, and et al. 2023. "Security in Cloud-Native Services: A Survey" Journal of Cybersecurity and Privacy 3, no. 4: 758-793. <https://doi.org/10.3390/jcp3040034>

44. H. A and A. B. D, "Cluster-Based Risk Analysis For Big Data Security Framework," 2024 7th International Conference on Circuit Power and Computing

Technologies (ICCPCT), Kollam, India, 2024, pp. 1085-1090, doi: 10.1109/ICCPCT61902.2024.10673052.

45. Morić, Zlatan, Vedran Dakić, and Tomislav Čavala. 2025. "Security Hardening and Compliance Assessment of Kubernetes Control Plane and Workloads" *Journal of Cybersecurity and Privacy* 5, no. 2: 30. <https://doi.org/10.3390/jcp5020030>

46. Dora, Rengalu Sudam, Cloud-Native Transformation of Telecom Networks (January 04, 2026). <http://dx.doi.org/10.2139/ssrn.6081610>

47. Michael Sanya Oluyede, Joseph Mart and Akinbusola Olusola et al. Container Security in Cloud Environments. *ScienceOpen Preprints*. 2024. DOI: 10.14293/PR2199.000730.v1

48. Vaggu, H. (2025). Automating Infrastructure as Code (IaC): A Comparative Study of Terraform, Pulumi, and Kubernetes Operators. *International Journal of AI, BigData, Computational and Management Studies*, 6(4), 1-9. <https://doi.org/10.63282/3050-9416.IJAIBDCMS-V6I4P101>

49. Aluri, V. 2025 "Quantum-Resilient Access Control Protocols for Cloud-Native Infrastructures in Post-Quantum Security Contexts" *Preprints*. <https://doi.org/10.20944/preprints202512.1147.v1>

50. Samuel, A. J. (2026). A Cloud-Native AI Framework for Real-Time Detection of Quantum Computing Attacks on Classical Cryptography. *International Journal of Technology, Management and Humanities*, 12(01), 24-32. <https://doi.org/10.21590/ijtmh.12.01.03>

51. Sappa, A. (2025). Context-aware security policies for data-driven applications in cloud-native architectures. *Journal of Internet Services and Information Security*, 15(2), 43-64. DOI: 10.58346/JISIS.2025.I2.004

52. D. L. Marino, C. S. Wickramasinghe, C. Rieger and M. Manic, "Self-Supervised and Interpretable Anomaly Detection Using Network Transformers," in *IEEE Transactions on Industrial Informatics*, vol. 21, no. 5, pp. 4252-4261, May 2025, doi: 10.1109/TII.2025.3534443.

53. A. Sengupta and V. Ramteke, "Leveraging Next-Generation Business

Intelligence for Proactive Cybersecurity Defense," 2025 IEEE 14th International Conference on Communication Systems and Network Technologies (CSNT), Bhopal, India, 2025, pp. 414-419, doi: 10.1109/CSNT64827.2025.10968525.

54. Su, J., Jiang, C., Jin, X., Qiao, Y., Xiao, T., Ma, H., ... & Lin, J. (2024). Large language models for forecasting and anomaly detection: A systematic literature review. arXiv preprint arXiv:2402.10350. <https://doi.org/10.48550/arXiv.2402.10350>

55. G. Parker et al., "Visualizing Anti-Patterns in Microservices at Runtime: A Systematic Mapping Study," in IEEE Access, vol. 11, pp. 4434-4442, 2023, doi: 10.1109/ACCESS.2023.3236165

56. G. Solaimalai, "AI-Augmented DevSecOps for Cloud-Native Security Automation," 2025 International Conference on Recent Innovation in Science Engineering and Technology (ICRISET), CHENNAI, India, 2025, pp. 1-8, doi: 10.1109/ICRISET64803.2025.11252281.

57. Sundberg, S., Brunstrom, A., Ferlin-Reiter, S., Høiland-Jørgensen, T., Brouer, J.D. (2023). Efficient Continuous Latency Monitoring with eBPF. In: Brunstrom, A., Flores, M., Fiore, M. (eds) Passive and Active Measurement. PAM 2023. Lecture Notes in Computer Science, vol 13882. Springer, Cham. https://doi.org/10.1007/978-3-031-28486-1_9

58. Gwak, S., Doan, T. P., & Jung, S. (2023). Container Instrumentation and Enforcement System for Runtime Security of Kubernetes Platform with eBPF. Intelligent Automation & Soft Computing, 37(2). DOI: 10.32604/iasc.2023.039565

59. Țălu, M. (2025). DDoS Mitigation in Kubernetes: A Review of Extended Berkeley Packet Filtering and eXpress Data Path Technologies. JUTI: Jurnal Ilmiah Teknologi Informasi, 23(2), 60-73. <https://doi.org/10.12962/j24068535.v23i2.a1268>

60. Lin, X., Chen, Z., Yang, W. et al. eBPF-Guard: a detection method for container escape via multi-level monitoring and enhanced analysis model. Empir Software Eng 31, 51 (2026). <https://doi.org/10.1007/s10664-025-10784-1>

61. A. Gupta, P. Gupta, U. P. Pandey, P. Kushwaha, B. P. Lohani and K. Bhati, "ZTSA: Zero Trust Security Architecture a Comprehensive Survey," 2024 International Conference on Communication, Computer Sciences and Engineering (IC3SE), Gautam Buddha Nagar, India, 2024, pp. 378-383, doi: 10.1109/IC3SE62002.2024.10593067
62. R. Tandon, G. Kaur, G. Kaur and A. Kukkar, "ZTaaS: Zero Trust as a Service for Cloud-Native Environment," 2025 12th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions) (ICRITO), Noida NCR, India, 2025, pp. 1-5, doi: 10.1109/ICRITO66076.2025.11241465
63. Vollem, S. Architecting Zero Trust Security for Distributed Hybrid and Multi-Cloud Enterprise Systems 2021 DOI: 10.55859/injmlr.2021.002
64. S. Xu and Y. Qian, "Dynamic Trust Scoring for Zero Trust at the Edge: A Multi-Factor, Context-Aware Approach," 2025 IEEE Conference on Communications and Network Security (CNS), Avignon, France, 2025, pp. 1-6, doi: 10.1109/CNS66487.2025.11195044
65. Avirneni, S. T. (2025). Establishing Workload Identity for Zero Trust CI/CD: From Secrets to SPIFFE-Based Authentication. arXiv preprint arXiv:2504.14760.<https://doi.org/10.48550/arXiv.2504.14760>
66. Zaka, Engr. Rukhsar and Mushtaher Uddin, Syed Muhammad and Hayat, Muhammad Ahsan and Murtaza, Aibah and Haider, Syed Arsalan and Beejal, Chaman lal, AI-Driven Cybersecurity for IoT-Cloud Ecosystems (September 06, 2025). <http://dx.doi.org/10.2139/ssrn.5510138>
67. H. Ran, C. Hu and Y. Zhuang, "Bridging Observability and Security: A Graph-Based Arbitration and Adaptive Sensing Approach via eBPF," in IEEE Access, vol. 14, pp. 19934-19951, 2026, doi: 10.1109/ACCESS.2026.3658370
68. F. Hasni, A. S. Nefzi, A. Hakiri and A. Jemai, "Autonomous Security Framework for AI-Native 6G Networks: Architecture and Threat Taxonomy," 2026 6th Middle East and North Africa Communications Conference (MENACOMM), Hammamet, Tunisia, 2026, pp. 1-8, doi:

10.1109/MENACOMM69507.2026.11532919

69. S. Zehra, H. J. Syed, F. Samad and U. Faseeha, "DeSFAM: An Adaptive eBPF and AI-Driven Framework for Securing Cloud Containers in Real Time," in IEEE Access, vol. 13, pp. 139203-139224, 2025, doi: 10.1109/ACCESS.2025.3592192

70. Nguyen, T., Nguyen, H., Ijaz, A., Sheikhi, S., Vasilakos, A. V., & Kostakos, P. (2024). Large language models in 6G security: challenges and opportunities. arXiv preprint arXiv:2403.12239. <https://doi.org/10.48550/arXiv.2403.12239>

71. S. Kostopoulos, D. Papatsaroucha, I. Kefaloukos and E. K. Markakis, "eIDPS: A real-time eBPF-based and Machine Learning-powered Network Intrusion Detection and Prevention Solution," 2025 6th International Conference in Electronic Engineering & Information Technology (EEITE), Chania, Greece, 2025, pp. 1-8, doi: 10.1109/EEITE65381.2025.11165804

72. J. E. Joyce, A. K. Nagaraj, S. Soni, U. A. N. S. Shaik, V. Patil and A. Tripathi, "Autonomous Infrastructure Healing: A Multi-Agent Kubernetes Recovery Framework," 2025 Supercomputing India (SCI), Bangalore, India, 2025, pp. 1-8, doi: 10.1109/SCI68648.2025.11333843

73. Mesioye, A. E., Adeduro, O. O., & Oluwagbemi, J. B. (2026). Defending AI Sentinels: A Multi-Layered Runtime Security Architecture for Generative AI in AIOps. *Methods in Science and Technology Studies*, 2(1), 101–115. <https://doi.org/10.64539/msts.v2i1.2026.420>

74. D. Brodimas et al., "Towards Intent-based Network Management for the 6G System adopting Multimodal Generative AI," 2024 Joint European Conference on Networks and Communications & 6G Summit (EuCNC/6G Summit), Antwerp, Belgium, 2024, pp. 848-853, doi: 10.1109/EuCNC/6GSummit60053.2024.10597022

75. Juan Marcelo Parra-Ullauri, Hari Madhukumar, Adrian-Cristian Nicolaescu, Xunzheng Zhang, Anderson Bravalheri, Rasheed Hussain, Xenofon Vasilakos, Reza Nejabati, Dimitra Simeonidou, kubeFlower: A privacy-preserving

framework for Kubernetes-based federated learning in cloud–edge environments, *Future Generation Computer Systems*, Volume 157, 2024, Pages 558-572, ISSN 0167-739X, <https://doi.org/10.1016/j.future.2024.03.041>

76. Cherukuri, B. R. (2024). Containerization in cloud computing: comparing Docker and Kubernetes for scalable web applications. *International Journal of Scientific Research and Application*, 13(01), 3302-3315. <https://doi.org/10.30574/ijrsra.2024.13.1.2035>

77. AI-Powered Anomaly Detection for Kubernetes Security: A Systematic Approach to Identifying Threats (A. K. Bhardwaj, P. Dutta, & P. Chintale , Trans.). (2024). *Babylonian Journal of Machine Learning*, 2024, 142-148. <https://doi.org/10.58496/BJML/2024/014>

78. A. Aly, M. Fayez, M. Al-Qutt and A. M. Hamad, "Multi-Class Threat Detection Using Neural Network and Machine Learning Approaches in Kubernetes Environments," 2024 6th International Conference on Computing and Informatics (ICCI), New Cairo – Cairo, Egypt, 2024, pp. 103-108, doi: 10.1109/ICCI61671.2024.10485133

79. Simonetto, S., & Bosch, P. (2024). Quantifying Risk in the Kill-Chain: Automating Threat Prioritization for Kubernetes Clusters. In *CSNG2024: Cyber Security Next Generation Workshop 2024*

80. Kulkarni, S. V. (2025). Securing Kubernetes: AI-Powered Container Security Agents. *International Journal of AI, BigData, Computational and Management Studies*, 6(1), 124-136. <https://doi.org/10.63282/3050-9416.IJAIBDCMS-V6I1P113>

81. Pitkar, H. (2025). Enhancing Kubernetes security with AI: anomaly detection for cloud-based workloads. *Int. Sci. J. of Eng. and Management*, 4(04). DOI: 10.55041/ISJEM02746 .

82. R. Morsli, *Kubernetes Intrusion Detection Datasets (Kube-IDS0)*, Kaggle, 2025. [Online]. Available: <https://www.kaggle.com/datasets/redamorsli/kube-ids0/code>

83. R. Morsli, N. Kara, H. Ould-Slimane and L. Lahlou, "Multidimensional

Intrusion Detection System for Containerized Environments," 2025 IEEE 11th International Conference on Network Softwarization (NetSoft), Budapest, Hungary, 2025, pp. 546-554, doi: 10.1109/NetSoft64993.2025.11080585.

84. Tsui, K.L., Chen, V., Jiang, W., Yang, F., Kan, C. (2023). Data Mining Methods and Applications. In: Pham, H. (eds) Springer Handbook of Engineering Statistics. Springer Handbooks. Springer, London. https://doi.org/10.1007/978-1-4471-7503-2_38

85. Roy, D. K., & Kalita, H. K. (2025). Anomaly detection in an open set environment using reinforcement learning. IET Information Security, 19(1), 7990749. <https://doi.org/10.1049/ise2.7990749>.

86. Gudelli, V. R. (2024). Anomaly detection in cloud networks using machine learning algorithms. African Journal of Artificial Intelligence and Sustainable Development, 4(1). <https://doi.org/10.5281/zenodo.15271016>

87. Rahman, M. S., Halder, S., Uddin, M. A., et al. (2021). An efficient hybrid system for anomaly detection in social networks. Cybersecurity, 4, 10. <https://doi.org/10.1186/s42400-021-00074-w>

88. Chua, W., et al. (2024). Web traffic anomaly detection using isolation forest. Informatics, 11(4), 83. <https://doi.org/10.3390/informatics11040083>

89. Wang, L., et al. (2025). A multi-angle semantic feature fusion method for web user behavior anomaly detection. Information, 16(9), 807. <https://doi.org/10.3390/info16090807>

90. Digital 2024 deep-dive: The state of internet adoption [Электронный ресурс]. – DataReportal, 2024. – Режим доступа: <https://datareportal.com/reports/digital-2024-deep-dive-the-state-of-internet-adoption> (дата звернения: 20.02.2026).

91. Data never sleeps 12.0 [Электронный ресурс]. – Domo, 2024. – Режим доступа: <https://www.domo.com/data-never-sleeps> (дата звернения: 20.02.2026).

92. ENISA threat landscape 2024 [Электронный ресурс]. – European Union Agency for Cybersecurity (ENISA), 2024. – Режим доступа: <https://www.enisa.europa.eu/publications/enisa-threat-landscape-2024> (дата

звернення: 20.02.2026).

93. 2024 Internet crime report [Електронний ресурс] / Federal Bureau of Investigation, Internet Crime Complaint Center (IC3). – 2025. – Режим доступу: https://www.ic3.gov/Media/PDF/AnnualReport/2024_IC3Report.pdf (дата звернення: 20.02.2026).

94. Luo Y., et al. Current status, challenges, and future trends of deep learning-based intrusion detection models // Journal of Imaging. – 2024. – Vol. 10, № 10. – P. 254. – DOI: <https://doi.org/10.3390/jimaging10100254>.

95. Global cybersecurity outlook 2025 [Електронний ресурс]. – World Economic Forum, 2025. – Режим доступу: <https://www.weforum.org/publications/global-cybersecurity-outlook-2025> (дата звернення: 20.02.2026).

96. Devi, S.R., Vallem, H., Chokkarapu, S., Hazel, J., Badrapu, A. (2026). *MTH-IDS—A Multi-tiered Hybrid Intrusion Detection System for Internet of Vehicle. In: Kumar, A., Ghinea, G., Merugu, S. (eds) Proceedings of the 4th International Conference on Cognitive and Intelligent Computing—Volume 1. ICCIC 2024. Cognitive Science and Technology. Springer, Singapore. https://doi.org/10.1007/978-981-95-0140-3_38

97. Mei, J., Zhou, T., Huang, K. et al. A survey on deep learning for polyp segmentation: techniques, challenges and future trends. Vis. Intell. 3, 1 (2025). <https://doi.org/10.1007/s44267-024-00071-w>

98. Pinto Neto E. C. та ін. CICIOT2023: A Real-Time Dataset and Benchmark for Large-Scale Attacks in IoT Environment // Sensors. – 2023. – Vol. 23, № 13. – Art. 5941. – DOI: 10.3390/s23135941.

99. Ferrag M. A. та ін. Edge-IIoTset: A New Comprehensive Realistic Cyber Security Dataset of IoT and IIoT Applications for Centralized and Federated Learning // IEEE Access. – 2022. – Vol. 10. – P. 40281–40306. – DOI: 10.1109/ACCESS.2022.3165809.

100. Chapaneri R., Shah S. Enhanced Detection of Imbalanced Malicious Network Traffic with Regularized Generative Adversarial Networks // Journal of

Network and Computer Applications. – 2022. – Art. 103368. – DOI: 10.1016/j.jnca.2022.103368.

101. Khanna S. Concept Drift-Based Intrusion Detection for Evolving Data Stream Classification in IDS: Approaches and Comparative Study // The Computer Journal. – 2024. – Vol. 67, № 7. – P. 2529–2547. – DOI: 10.1093/comjnl/bxae023.

102. Elsedimy E. I., Elhadidy H., Abohashish S. M. M. A Novel Intrusion Detection System Based on a Hybrid Quantum Support Vector Machine and Improved Grey Wolf Optimizer // Cluster Computing. – 2024. – DOI: 10.1007/s10586-024-04458-8.

ДОДАТОК А

Код програмної реалізації методів представлених у роботі

A.1 Код програмної реалізації методу виявлення аномалій в інфраструктурі Kubernetes

```

import pandas as pd
import numpy as np

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.impute import SimpleImputer
from sklearn.metrics import classification_report, confusion_matrix,
f1_score

from tensorflow.keras.models import Model
from tensorflow.keras.layers import Input, LSTM, RepeatVector,
TimeDistributed, Dense

# Завантаження наборів даних
df_boa = pd.read_csv("boa_dataset_ml_ready_frontend_microservice.csv")
df_dvwa = pd.read_csv("dvwa_dataset_ml_ready.csv")

# Об'єднання даних
df = pd.concat([df_boa, df_dvwa], ignore_index=True)

# Формування ознак та міток
X = df.drop(columns=["label"])
y = df["label"]

# Вибір тільки нормальних зразків для навчання автоенкодера
X_normal = X[y == 0]

# Заповнення пропущених значень
imputer = SimpleImputer(strategy="mean")
X_normal_imp = imputer.fit_transform(X_normal)
X_imp = imputer.transform(X)

# Масштабування ознак
scaler = StandardScaler()
X_normal_scaled = scaler.fit_transform(X_normal_imp)
X_scaled = scaler.transform(X_imp)

# Формування часових послідовностей
def create_sequences(data, window_size=10):
    sequences = []
    for i in range(len(data) - window_size):
        seq = data[i:i + window_size]
        sequences.append(seq)
    return np.array(sequences)

WINDOW = 10

X_normal_seq = create_sequences(X_normal_scaled, WINDOW)
X_all_seq = create_sequences(X_scaled, WINDOW)

# Розділення нормальних послідовностей на навчальну та валідаційну вибірки
X_train, X_val = train_test_split(
    X_normal_seq,
    test_size=0.2,

```

```

        random_state=42
    )

    timesteps = X_train.shape[1]
    features = X_train.shape[2]

    # Побудова моделі LSTM Autoencoder
    inputs = Input(shape=(timesteps, features))

    encoded = LSTM(64, return_sequences=False)(inputs)
    encoded = Dense(32, activation="relu")(encoded)

    decoded = RepeatVector(timesteps)(encoded)
    decoded = LSTM(64, return_sequences=True)(decoded)
    decoded = TimeDistributed(Dense(features))(decoded)

    lstm_autoencoder = Model(inputs, decoded)
    lstm_autoencoder.compile(optimizer="adam", loss="mse")

    # Навчання моделі на нормальній поведінці системи
    history = lstm_autoencoder.fit(
        X_train,
        X_train,
        epochs=100,
        batch_size=128,
        validation_data=(X_val, X_val),
        verbose=1
    )

    # Обчислення помилки реконструкції на валідаційній вибірці
    X_val_pred = lstm_autoencoder.predict(X_val)
    mse_val = np.mean(np.power(X_val - X_val_pred, 2), axis=(1, 2))

    # Визначення порогу аномальності
    threshold = np.percentile(mse_val, 95)

    # Виявлення аномалій у всій вибірці
    X_all_pred = lstm_autoencoder.predict(X_all_seq)
    mse_all = np.mean(np.power(X_all_seq - X_all_pred, 2), axis=(1, 2))

    y_pred_anomaly = (mse_all > threshold).astype(int)

    # Узгодження міток із часовими послідовностями
    y_true = y.values[WINDOW:]
    y_true_binary = (y_true != 0).astype(int)

    # Оцінювання якості виявлення аномалій
    print("Classification report:")
    print(classification_report(y_true_binary, y_pred_anomaly))

    print("Confusion matrix:")
    print(confusion_matrix(y_true_binary, y_pred_anomaly))

    # Додатковий підбір оптимального порогу за F1-score
    thresholds = np.linspace(min(mse_val), max(mse_val), 100)

    best_thr = None
    best_f1 = 0

    for thr in thresholds:
        preds = (mse_all > thr).astype(int)

```

```

f1 = f1_score(y_true_binary, preds)

if f1 > best_f1:
    best_f1 = f1
    best_thr = thr

y_pred_opt = (mse_all > best_thr).astype(int)

print("Best threshold:", best_thr)
print("Best F1:", best_f1)

print("Optimized classification report:")
print(classification_report(y_true_binary, y_pred_opt))

print("Optimized confusion matrix:")
print(confusion_matrix(y_true_binary, y_pred_opt))

# Збереження навченої моделі
lstm_autoencoder.save("lstm_autoencoder_kubernetes.h5")

```

A.2 Код програмної реалізації методу виявлення аномалій в API хмарних сервісів

```

import pandas as pd
import numpy as np
import tensorflow as tf

from tensorflow.keras import layers, Model
from sklearn.metrics import classification_report, confusion_matrix

# Завантаження набору даних сесій користувачів
df_train = pd.read_csv("train_sessions.csv")

# Формування часових та поведінкових ознак
def time_operations(df):
    if "target" in df.columns:
        df["number_of_sites"] = ((df.notna().sum(axis=1)) / 2) - 1
    else:
        df["number_of_sites"] = ((df.notna().sum(axis=1)) / 2) - 0.5

    for col in [f"time{i}" for i in range(1, 11)]:
        df[col] = pd.to_datetime(df[col])

    df["session_start"] = df[[f"time{i}" for i in range(1,
11)]] .min(axis=1)
    df["session_end"] = df[[f"time{i}" for i in range(1, 11)]] .max(axis=1)
    df["session_len"] = (df["session_end"] -
df["session_start"]).dt.seconds

    df["weekday"] = df["session_start"].dt.dayofweek.astype("category")
    df["start_hour"] = df["session_start"].dt.hour.astype("category")
    df["end_hour"] = df["session_end"].dt.hour.astype("category")
    df["day"] = df["session_start"].dt.dayofyear
    df["minute"] = df["session_start"].dt.minute
    df["month"] = df["session_start"].dt.month

    df = df.sort_values("session_start", ignore_index=True)

```

```

df = df.reset_index().rename(columns={"index": "time_index"})
df = df.sort_values("session_id", ignore_index=True)

return df

df_train = time_operations(df_train)

# Виділення характеристик для користувача сайтів
def alice_sites(train_df):
    df_sites = []

    for j in range(1, 11):
        temp = train_df[["session_id", f"site{j}", "target"]]
        temp.columns = ["session_id", "site", "target"]
        df_sites.append(temp)

    df_sites = pd.concat(df_sites, ignore_index=True)
    sites = df_sites.groupby("site")["target"].mean()

    return sites

def add_alice_sites(df, sites, alice_size=0.06):
    mask = sites[sites > alice_size].index
    df["alice_site"] = 0

    for col in [f"site{i}" for i in range(1, 11)]:
        alice_mask = df[col].isin(mask)
        df.loc[alice_mask, "alice_site"] = 1

    return df

def add_more_alice_sites(df, sites, a_s_2=0.3, a_s_3=0.01, a_s_4=0.5):
    count = 1

    for threshold in [a_s_2, a_s_3, a_s_4]:
        count += 1
        mask = sites[sites > threshold].index
        df[f"alice_site_{count}"] = 0

        for col in [f"site{i}" for i in range(1, 11)]:
            alice_mask = df[col].isin(mask)
            df.loc[alice_mask, f"alice_site_{count}"] = 1

    return df

sites = alice_sites(df_train)
df_train = add_alice_sites(df_train, sites)
df_train = add_more_alice_sites(df_train, sites)

# Поділ даних за часом
def time_threshold_split(df, date="2014-03-01"):
    train_df = df[df.session_start < date].copy()
    test_df = df[df.session_start >= date].copy()

    return train_df, test_df

train_df, test_df = time_threshold_split(df_train)

# Кодування послідовностей сайтів
SITE_COLS = [f"site{i}" for i in range(1, 11)]

```

```

def build_site_vocab(df, site_cols=SITE_COLS, min_freq=1):
    all_sites = pd.Series(df[site_cols].values.ravel())
    all_sites = all_sites.dropna()

    value_counts = all_sites.value_counts()
    value_counts = value_counts[value_counts >= min_freq]

    vocab = {
        site: idx
        for idx, site in enumerate(value_counts.index, start=1)
    }

    return vocab

def encode_sites(df, vocab, site_cols=SITE_COLS, max_len=10):
    X = np.zeros((len(df), max_len), dtype=np.int32)

    for j, col in enumerate(site_cols[:max_len]):
        values = df[col].values
        X[:, j] = np.array(
            [vocab.get(v, 0) if pd.notna(v) else 0 for v in values],
            dtype=np.int32
        )

    return X

# Формування числових ознак сесії
NUM_COLS = [
    "session_len",
    "number_of_sites",
    "weekday",
    "start_hour",
    "end_hour"
]

def build_numeric_features(df, num_cols=NUM_COLS):
    X_num = df[num_cols].copy()

    for col in X_num.columns:
        if pd.api.types.is_categorical_dtype(X_num[col]):
            X_num[col] = X_num[col].cat.codes.replace(-1, np.nan)
        elif X_num[col].dtype == "object":
            X_num[col] = pd.to_numeric(X_num[col], errors="coerce")

    for col in X_num.columns:
        X_num[col] = X_num[col].astype("float32")
        median = np.nanmedian(X_num[col].values)
        X_num[col] = X_num[col].fillna(median)

    X_num = (X_num - X_num.mean()) / (X_num.std() + 1e-8)

    return X_num.values.astype(np.float32)

vocab = build_site_vocab(train_df)

X_sites_train = encode_sites(train_df, vocab)
X_num_train = build_numeric_features(train_df)

X_sites_test = encode_sites(test_df, vocab)
X_num_test = build_numeric_features(test_df)

```

```

normal_mask_train = train_df["target"].values == 1

X_sites_train_norm = X_sites_train[normal_mask_train]
X_num_train_norm = X_num_train[normal_mask_train]

y_test = test_df["target"].values

# Побудова LSTM Autoencoder для API-сесій
max_len = 10
vocab_size = max(vocab.values()) + 1
embed_dim = 32
enc_units = 64
latent_dim = 32

# Вхід 1: послідовність сайтів/API-ресурсів
inp_sites = layers.Input(
    shape=(max_len,),
    dtype="int32",
    name="sites_input"
)

x = layers.Embedding(
    input_dim=vocab_size + 1,
    output_dim=embed_dim,
    mask_zero=True,
    name="embedding"
)(inp_sites)

x = layers.LSTM(
    enc_units,
    name="lstm_core"
)(x)

# Вхід 2: числові ознаки сесії
inp_num = layers.Input(
    shape=(X_num_train.shape[1],),
    dtype="float32",
    name="num_input"
)

h = layers.Concatenate(name="concatenate")([x, inp_num])
h = layers.Dense(64, activation="relu", name="dense_hidden")(h)

z = layers.Dense(
    latent_dim,
    activation="relu",
    name="latent_space"
)(h)

# Декодер: реконструкція числових ознак сесії
d = layers.Dense(64, activation="relu", name="decoder_dense")(z)

out_num = layers.Dense(
    X_num_train.shape[1],
    activation="linear",
    name="recon_output"
)(d)

ae = Model(
    inputs=[inp_sites, inp_num],
    outputs=out_num,

```

```

        name="api_lstm_autoencoder"
    )

    ae.compile(
        optimizer=tf.keras.optimizers.Adam(learning_rate=1e-3),
        loss="mse"
    )

    # Навчання моделі
    history = ae.fit(
        [X_sites_train_norm, X_num_train_norm],
        X_num_train_norm,
        epochs=25,
        batch_size=256,
        validation_split=0.2,
        shuffle=True,
        verbose=1
    )

    # Обчислення помилки реконструкції
    def reconstruction_error(model, X_sites, X_num):
        reconstructed = model.predict(
            [X_sites, X_num],
            batch_size=512,
            verbose=0
        )

        error = np.mean((X_num - reconstructed) ** 2, axis=1)

        return error

    err_train_norm = reconstruction_error(
        ae,
        X_sites_train_norm,
        X_num_train_norm
    )

    threshold = np.quantile(err_train_norm, 0.95)

    err_test = reconstruction_error(
        ae,
        X_sites_test,
        X_num_test
    )

    pred_anomaly = (err_test > threshold).astype(int)

    # Перетворення міток:
    # 0 – нормальна активність, 1 – вторгнення / аномалія
    y_true = (y_test == 0).astype(int)

    print("Threshold:", threshold)

    print("Classification report:")
    print(classification_report(y_true, pred_anomaly))

    print("Confusion matrix:")
    print(confusion_matrix(y_true, pred_anomaly))

    # Збереження моделі
    ae.save("api_lstm_autoencoder.h5")

```

ДОДАТОК Б

Список публікацій здобувача

1. Мартовицький В. В., Свиридов А. С., Авдєєв О. В., Гудзинський І. М., Коротецький О. В. Дослідження методів виявлення аномалій у API журналах для забезпечення безпеки та надійності програмних систем // Вісник Херсонського національного технічного університету. 2025. Т. 2, № 1(92). С. 142–148. DOI: <https://doi.org/10.35546/kntu2078-4481.2025.1.2.19>.
2. Свиридов А. С., Гусейнов Р. Б., Винниченко С. М. Метод виявлення аномалій у поведінці користувача вебсайту // Herald of Khmelnytskyi National University. Technical Sciences. 2026. Т. 363, № 2. С. 211–219. DOI: <https://doi.org/10.31891/2307-5732-2026-363-28>.
3. Свиридов А. С., Северінов О. В. Метод виявлення аномалій у Kubernetes-кластерах з використанням LSTM Autoencoder // Herald of Khmelnytskyi National University. Technical Sciences. 2026. Т. 361, № 1. С. 501–509. DOI: <https://doi.org/10.31891/2307-5732-2026-361-70>.
4. Свиридов А. С., Гудзинський І. М. Архітектура мультиагентної системи виявлення вторгнень із використанням машинного навчання // Вісник Херсонського національного технічного університету. 2026. Т. 1, № 2(97). С. 360–366. DOI: <https://doi.org/10.35546/kntu2078-4481.2026.2.44>.
5. Kuchuk G., Kovalenko A., Elyasi Komari I., Svyrydov A., Kharchenko V. Improving Big Data Centers Energy Efficiency: Traffic Based Model and Method // Advances in Computer Science for Engineering and Education II. Cham : Springer, 2019. P. 77–88. DOI: 10.1007/978-3-030-00253-4_8.
6. Северінов О. В., Свиридов А. С. Виявлення аномалій у API журналах для підвищення безпеки програмних систем // Сучасні напрями розвитку інформаційно-комунікаційних технологій та засобів управління : тези доповідей тринадцятої міжнародної науково-технічної конференції, 27–28 листопада 2025 р. Харків, 2025. Т. 3, секц. 4. С. 35.
7. Свиридов А. С. Метод виявлення аномалій у поведінці користувачів

вебсайтів // Science and education: synergy of innovation : Proceedings of the 8th International scientific and practical conference, Berlin, Germany, 23–25 March 2026. Berlin : MDPC Publishing, 2026. P. 198. URL: <https://sci-conf.com.ua/viii-mizhnarodna-naukovo-praktichna-konferentsiya-science-and-education-synergy-of-innovation-23-25-03-2026-berlin-nimechchina-arhiv/>.

8. Свиридов А. С. Архітектура мультиагентної системи виявлення вторгнень із використанням машинного навчання // Scientific development in a changing world : Proceedings of the 3rd International scientific and practical conference, Lviv, Ukraine, 16–18 March 2026. Львів : SPC “Sci-conf.com.ua”, 2026. P. 330. URL: <https://sci-conf.com.ua/iii-mizhnarodna-naukovo-praktichna-konferentsiya-scientific-development-in-a-changing-world-16-18-03-2026-lviv-ukrayina-arhiv/>.

9. Свиридов А. С. Метод виявлення аномалій у Kubernetes-кластерах з використанням LSTM Autoencoder // European science and innovation congress : Proceedings of the 4th International scientific and practical conference, Barcelona, Spain, 9–11 March 2026. Barcelona : Barca Academy Publishing, 2026. P. 169. URL: <https://sci-conf.com.ua/iv-mizhnarodna-naukovo-praktichna-konferentsiya-european-science-and-innovation-congress-9-11-03-2026-barselona-ispaniya-arhiv/>.

10. Свиридов А. С., Сєверінов О. В. Дослідження методів виявлення аномалій у API журналах для забезпечення безпеки та надійності програмних систем // Сучасні проблеми інфокомунікацій, радіоелектроніки та наносистем : тези доповідей шістнадцятої міжнародної науково-технічної конференції, 29–30 квітня 2026 р. Харків, 2026. Т. 3, секц. 4. С. 112–113.

ДОДАТОК В

Акти впровадження

«ЗАТВЕРДЖУЮ»

Директор ТОВ «РЕІНВЕНТИНГ СОФТВЕР
ДЕВЕЛОПМЕНТ»

Карпенко І.В.

«5» лютого 2026 р.



Впровадження результатів дисертаційної роботи СВИРИДОВА Артема Сергійовича

Підтверджую, що наступні результати наукових досліджень
СВИРИДОВА Артема Сергійовича, а саме:

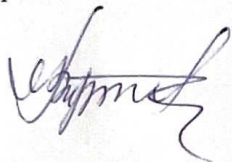
- метод виявлення аномалій у Kubernetes-кластерах за рахунок аналізу телеметрії контейнеризованого середовища та застосування моделі LSTM Autoencoder для виявлення прихованих аномальних патернів у часових послідовностях подій;
- модель інтелектуальної системи виявлення аномалій у хмарних застосунках шляхом використання методів штучного інтелекту, що дозволяє ефективно інтегрувати API-журнали, мережеві події, поведінкові дані та телеметрію Kubernetes-кластерів у межах єдиного процесу моніторингу та аналізу безпеки.

Використання запропонованого методу виявлення аномалій у Kubernetes-кластерах із застосуванням моделі LSTM Autoencoder забезпечило автоматизацію процесу аналізу часових послідовностей подій та телеметрії контейнеризованого середовища підприємства. Це дозволило з високою точністю виявляти приховані аномальні патерни, ефективніше відокремлювати звичайну поведінку користувачів від автоматизованої або потенційно небезпечної активності, зменшити кількість помилкових спрацювань систем моніторингу та скоротити час реагування на інциденти кібербезпеки.

Отримані результати стали основою для подальшого використання моделі інтелектуальної системи аналізу подій у хмарному середовищі, яка дозволила об'єднати API-журнали, мережеві події, поведінкові метрики та телеметрію Kubernetes-кластерів у межах єдиного процесу моніторингу безпеки хмарної інфраструктури. Впровадження моделі дало можливість модернізувати процеси адміністрування розподілених середовищ, підвищити стабільність функціонування хмарних застосунків та забезпечити більш гнучкий контроль стану інформаційних ресурсів підприємства.

Результати впровадження підтвердили практичну цінність та ефективність запропонованих рішень для задач моніторингу й захисту хмарної інфраструктури підприємства.

Директор
«05» лютого 2026р.



І. Карпенко.