Міністерство освіти і науки України

Харківський національний університет радіоелектроніки

Навчально-науковий центр заочної форми навчання
(повна назва)

Кафедра *Інформаційно-мережної інженерії*
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА

Пояснювальна записка

рівень вищої освіти *перший (бакалаврський)*

Система виявлення вторгнень у кластері Kubernetes

(тема)

Виконав:
здобувач *4* року навчання,
групи *ТРІМІз-22-1*
Віталій МАРТОВИЦЬКИЙ
(власне ім'я, прізвище)

Спеціальність
172 Телекомунікації та радіотехніка
(код і повна назва спеціальності)

Тип програми *освітньо-професійна*
(освітньо-професійна або освітньо-наукова)

Освітня програма
Інформаційно-мережна інженерія
(повна назва освітньої програми)

Керівник: *доц. Дарія ЧЕБОТАРЬОВА*
(посада, власне ім'я, прізвище)

Допускається до захисту

В.о. зав. кафедри ІМІ _____ *Дарія ЧЕБОТАРЬОВА*
(підпис) (власне ім'я, прізвище)

2026 р.

Не містить відомостей, заборонених до відкритого публікування

Студент _____ *Віталій Мартовицький*
(підпис) (власне ім'я, прізвище)

Керівник _____ *Дарія Чеботарьова*
(підпис) (власне ім'я, прізвище)

Харківський національний університет радіоелектроніки

Навчально-науковий центр заочної форми навчання

Кафедра *Інформаційно-мережної інженерії*
(повна назва)

Рівень вищої освіти *перший (бакалаврський)*

Спеціальність *172 Телекомунікації та радіотехніка*
(код і повна назва)

Тип програми *освітньо-професійна*

Освітня програма *Інформаційно-мережна інженерія*
(повна назва)

ЗАТВЕРДЖУЮ:

Зав. кафедри ІМІ_____
(підпис)

"*18*" *травня* 2026р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувачеві *Мартовицькому Віталію Олександровичу*
(прізвище, ім'я, по батькові)

1. Тема роботи *Система виявлення вторгнень у кластері Kubernetes*

затверджена наказом по університету від "*18*" *травня* 2026 р. № *73 Стз*

2. Термін подання здобувачем роботи до екзаменаційної комісії *14 липня 2026 р.*

3. Вхідні дані до роботи

мережевий трафік контейнеризованих застосунків у форматі PCAP,
набір даних Improved CICIDS,
критерії оцінювання ефективності системи.

4. Перелік питань, що потрібно опрацювати у роботі

Проаналізувати архітектуру Kubernetes та особливості функціонування.
Розглянути особливості потокового аналізу та формування ознак мережевих з'єднань.
Обґрунтувати вибір сигнатурного аналізу Snort і алгоритмів машинного навчання k-Nearest Neighbors та Isolation Forest.
Проаналізувати й підготувати набір даних Improved CICIDS.
Реалізувати захоплення, оброблення та класифікацію мережевого трафіку.
Провести експериментальне дослідження та оцінити ефективність системи.

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій *Слайди у форматі Power Point - 11 слайдів*

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Строк / терміни виконання етапів роботи	Примітка
1	*Ознайомлення із завданням. Уточнення ТЗ.*	*18.05.26*	*виконано*
2	*Підбір літератури за темою роботи.*	*19.05 – 01.06.26*	*виконано*
3	*Виконання розділу 1*	*02.06 – 15.06.26*	*виконано*
4	*Виконання розділу 2*	*16.06 – 22.06.26*	*виконано*
5	*Виконання розділу 3*	*23.06 – 01.07.26*	*виконано*
6	*Оформлення пояснювальної записки*	*02.07 – 10.07.26*	*виконано*
7	*Оформлення презентаційного матеріалу,*		
	підготовка до захисту у ЕК, захист.	*11.07 – 17.07.26*	*виконано*

Дата видачі завдання " *18* " *травня* 20 26 р.

Здобувач _____
(підпис)

Керівник роботи _____ *доц. Дарія Чеботарьова*
(підпис) (посада, власне ім'я, прізвище)

РЕФЕРАТ

Пояснювальна записка: 78 с., 11 рис., 7 табл., 1 дод., 11 джерел.

KUBERNETES, СИСТЕМА ВИЯВЛЕННЯ ВТОРГНЕНЬ, МЕРЕЖЕВИЙ ТРАФІК, МЕРЕЖЕВИЙ ПОТІК, МАШИННЕ НАВЧАННЯ, K-NEAREST NEIGHBORS, ISOLATION FOREST, SNORT, CICFLOWMETER, PCAP.

Метою кваліфікаційної роботи є розроблення системи виявлення вторгнень у кластері Kubernetes на основі поєднання сигнатурного аналізу та алгоритмів машинного навчання для підвищення ефективності виявлення мережевих атак.

У ході виконання кваліфікаційної роботи проаналізовано архітектуру Kubernetes, особливості мережевої взаємодії контейнеризованих застосунків і сучасні підходи до виявлення вторгнень. Розроблено архітектуру системи, яка забезпечує захоплення мережевого трафіку за допомогою Ksniff, формування файлів PCAP та їх сигнатурний аналіз засобами Snort. Для перетворення пакетного трафіку на двонаправлені мережеві потоки використано Improved CICFlowMeter. Проведено підготовку набору даних Improved CICIDS, очищення та нормалізацію ознак, а також зменшення їх розмірності методом головних компонент. Для класифікації відомих типів атак застосовано алгоритм k-Nearest Neighbors, а для виявлення аномальної та потенційно невідомої активності Isolation Forest. Компоненти інтегровано в єдину систему автоматизованого аналізу мережевого трафіку.

ABSTRACT

Explanatory note: 78 pages, 11 figures, 7 tables, 1 appendices, 11 sources.

KUBERNETES, INTRUSION DETECTION SYSTEM, NETWORK TRAFFIC, NETWORK FLOW, MACHINE LEARNING, K-NEAREST NEIGHBORS, ISOLATION FOREST, SNORT, CICFLOWMETER, PCAP.

The major goal of this thesis is to develop an intrusion detection system for a Kubernetes cluster based on a combination of signature-based analysis and machine learning algorithms in order to improve the effectiveness of network attack detection.

During the qualification work, the Kubernetes architecture, the specifics of network interaction between containerized applications, and modern intrusion detection approaches were analyzed. The architecture of a system that captures network traffic using Ksniff, generates PCAP files, and performs their signature-based analysis using Snort was developed. Improved CICFlowMeter was used to convert packet traffic into bidirectional network flows. The Improved CICIDS dataset was prepared, including data cleaning, feature normalization, and dimensionality reduction using the Principal Component Analysis method. The k-Nearest Neighbors algorithm was applied to classify known types of attacks, while Isolation Forest was used to detect anomalous and potentially unknown activity. All components were integrated into a unified automated network traffic analysis system.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ 7
ВСТУП 9
1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ 11
1.1 Загальні відомості про Kubernetes 11
1.2 Системи виявлення вторгнень 17
1.3 Мережні потоки в системах виявлення вторгнень 19
2 ОГЛЯД ЗАСТОСУВАННЯ МАШИННОГО НАВЧАННЯ ДЛЯ ВИЯВЛЕННЯ ВТОРГНЕНЬ У МЕРЕЖНОМУ ТРАФІКУ 25
2.1 Аналіз алгоритмів машинного навчання для виявлення вторгнень у мережевому трафіку 25
2.2 Особливості застосування машинного навчання в системах виявлення вторгнень 27
2.3 Використання нейронних мереж для аналізу мережевого трафіку та виявлення атак 30
2.4 Використання алгоритму k-Nearest Neighbors для класифікації мережевого трафіку 32
3 РОЗРОБКА ТА ДОСЛІДЖЕННЯ СИСТЕМИ ВИЯВЛЕННЯ ВТОРГНЕНЬ У КЛАСТЕРІ KUBERNETES 38
3.1 Методика проведення дослідження 38
3.2 Аналіз наборів даних для навчання системи виявлення вторгнень 42
3.3 Експериментальне дослідження системи виявлення вторгнень 45
ВИСНОВКИ 67
ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ 68
ДОДАТОК А Графічний матеріал кваліфікаційної роботи 70

ПЕРЕЛІК СКОРОЧЕНЬ

CSV – формат файлів для зберігання табличних даних (англ., Comma-Separated Values),

IDS – система виявлення вторгнень (англ., Intrusion Detection System),

IP – міжмережевий протокол (англ., Internet Protocol),

KNN – алгоритм k найближчих сусідів (англ., k-Nearest Neighbors),

Ksniff – плагін kubectl для перехоплення мережевого трафіку контейнерів Kubernetes,

ML – машинне навчання (англ., Machine Learning),

PCA – метод головних компонент (англ., Principal Component Analysis),

PCAP – формат файлів захоплення мережевого трафіку (англ., Packet Capture),

Pod – найменша одиниця розгортання в Kubernetes,

REST – архітектурний стиль взаємодії вебсервісів (англ., Representational State Transfer),

Snort – система сигнатурного виявлення вторгнень,

TCP – протокол керування передаванням (англ., Transmission Control Protocol),

UDP – протокол користувацьких датаграм (англ., User Datagram Protocol),

API – програмний інтерфейс застосунків (англ., Application Programming Interface),

CICFlowMeter – програмний засіб формування статистичних характеристик мережевих потоків,

CPU – центральний процесор (англ., Central Processing Unit),

DoS – атака типу «відмова в обслуговуванні» (англ., Denial of Service),

DDoS – розподілена атака типу «відмова в обслуговуванні» (англ., Distributed Denial of Service),

DNS – система доменних імен (англ., Domain Name System),

FTP – протокол передавання файлів (англ., File Transfer Protocol),

HTTP – протокол передавання гіпертексту (англ., HyperText Transfer Protocol),

HTTPS – захищений протокол передавання гіпертексту (англ., HyperText Transfer Protocol Secure),

Kubernetes – платформа оркестрації контейнеризованих застосунків,

mTLS – взаємна автентифікація за допомогою TLS (англ., Mutual Transport Layer Security),

TLS – протокол захисту транспортного рівня (англ., Transport Layer Security),

Wireshark – програмний аналізатор мережевого трафіку.

ВСТУП

Сучасний розвиток хмарних технологій, контейнеризації та мікросервісної архітектури призвів до широкого впровадження платформи Kubernetes як стандарту оркестрації контейнеризованих застосунків. Kubernetes забезпечує автоматизацію розгортання, масштабування та керування програмними сервісами, що дозволяє створювати високодоступні й масштабовані інформаційні системи. Разом із зростанням популярності Kubernetes збільшується і кількість кіберзагроз, спрямованих на компрометацію контейнерних середовищ, несанкціонований доступ до ресурсів кластера та порушення доступності сервісів.

Особливості архітектури Kubernetes, зокрема динамічне створення та видалення контейнерів, автоматичне масштабування, використання сервісних сіток (Service Mesh), а також широке застосування шифрування мережевого трафіку, значно ускладнюють використання традиційних систем виявлення вторгнень. Класичні сигнатурні системи ефективно виявляють лише відомі типи атак, тоді як нові або модифіковані загрози можуть залишатися непоміченими. Це обумовлює необхідність застосування сучасних методів аналізу мережевого трафіку, зокрема алгоритмів машинного навчання, здатних виявляти аномальну поведінку та невідомі сценарії атак.

Актуальність роботи полягає у необхідності розроблення системи виявлення вторгнень, адаптованої до особливостей функціонування кластерів Kubernetes, яка забезпечує автоматизований аналіз мережевого трафіку, поєднуючи переваги сигнатурного аналізу та методів машинного навчання. Використання потокового аналізу мережевого трафіку дозволяє здійснювати моніторинг навіть за умови застосування шифрування мережевих з'єднань, що є важливою вимогою сучасних хмарних інфраструктур.

Метою кваліфікаційної роботи є розроблення системи виявлення вторгнень у кластері Kubernetes на основі поєднання сигнатурного аналізу та

алгоритмів машинного навчання для підвищення ефективності виявлення мережевих атак.

Для досягнення поставленої мети необхідно вирішити такі завдання:

- проаналізувати архітектуру Kubernetes та особливості забезпечення інформаційної безпеки контейнеризованих середовищ;
- дослідити сучасні системи виявлення вторгнень і методи аналізу мережевого трафіку;
- обґрунтувати вибір алгоритмів машинного навчання для виявлення мережевих атак;
- розробити архітектуру системи виявлення вторгнень для Kubernetes-кластера;
- реалізувати програмне забезпечення для автоматизованого збору та аналізу мережевого трафіку;
- провести експериментальне дослідження та оцінити ефективність розробленої системи.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Загальні відомості про Kubernetes

Kubernetes – це платформа з відкритим вихідним кодом, призначена для автоматизації розгортання, масштабування та керування контейнеризованими застосунками. Спочатку вона була розроблена компанією Google, а сьогодні її розвиток і супровід здійснюються організацією Cloud Native Computing Foundation (CNCF). Kubernetes фактично стала загальновизнаним стандартом оркестрації контейнеризованих робочих навантажень у розподілених обчислювальних середовищах. Завдяки абстрагуванню від фізичної та віртуальної інфраструктури ця платформа є одним із ключових інструментів керування масштабними мікросервісними системами [1]. Розуміння архітектури Kubernetes є необхідною передумовою для визначення можливостей і складності розроблення ефективних та масштабованих засобів захисту таких середовищ.

Kubernetes використовує архітектуру «керуюча площина робочі вузли», яка забезпечує чітке розмежування функцій оркестрації кластера та виконання прикладних робочих навантажень. Такий поділ спрощує керування контейнеризованими застосунками у великих масштабах, а також сприяє забезпеченню відмовостійкості та масштабованості системи.

Керувюча площина (Control Plane) виконує функції центрального компонента керування кластером Kubernetes. Вона відповідає за підтримання бажаного стану системи та координує роботу всіх компонентів кластера, забезпечуючи відповідність фактичного стану застосунків стану, визначеному в їх конфігураціях. До складу керувальної площини входять такі основні компоненти:

- API Server – центральний інтерфейс взаємодії з Kubernetes, який надає REST API для внутрішніх компонентів системи, зовнішніх користувачів і

засобів адміністрування. API Server приймає, перевіряє та обробляє запити, виконуючи роль основного комунікаційного вузла кластера;

- Scheduler – компонент, який призначає новостворені поди відповідним робочим вузлам. Під час вибору вузла планувальник ураховує доступність обчислювальних ресурсів, правила спорідненості та антиспорідненості, обмеження розміщення й інші задані параметри;

- Controller Manager – компонент, який виконує сукупність циклів керування, кожен із яких відповідає за підтримання певного аспекту стану кластера. Наприклад, контролер реплікації забезпечує постійну роботу заданої кількості копій подів, а контролер завдань відповідає за виконання пакетних операцій і завершення задач;

- etcd – розподілене високодоступне сховище типу «ключ – значення», яке використовується для збереження стану кластера. У ньому містяться конфігураційні дані, відомості про виявлення сервісів, метадані ресурсів та інша інформація, необхідна для функціонування керувальної площини.

- Робочі вузли (Worker Nodes) призначені для безпосереднього виконання прикладних робочих навантажень. На кожному вузлі розміщується один або декілька подів, кожен із яких може містити один або кілька контейнерів, об’єднаних спільними залежностями, мережевим простором і параметрами середовища виконання. До основних компонентів робочого вузла належать:

- Kubelet – агент вузла, який забезпечує запуск і підтримання контейнерів у працездатному стані. Він взаємодіє з керувальною площиною, отримує описи подів, контролює їх фактичний стан і передає інформацію про стан вузла та контейнерів;

- Kube-proxy – мережевий компонент, який підтримує правила маршрутизації та переспрямування трафіку на вузлі. Він забезпечує мережеву взаємодію між подами та спрямовує запити сервісів до відповідних внутрішніх подів;

- середовище виконання контейнерів (Container Runtime) – програмний

компонент, наприклад containerd або CRI-O, який безпосередньо забезпечує запуск, зупинення та керування життєвим циклом контейнерів. Середовище виконання контейнерів працює у взаємодії з Kubelet і виконує команди щодо створення та обслуговування контейнеризованих робочих навантажень.

Загальну структуру кластера Kubernetes та взаємодію його основних компонентів наведено на рис. 1.1.

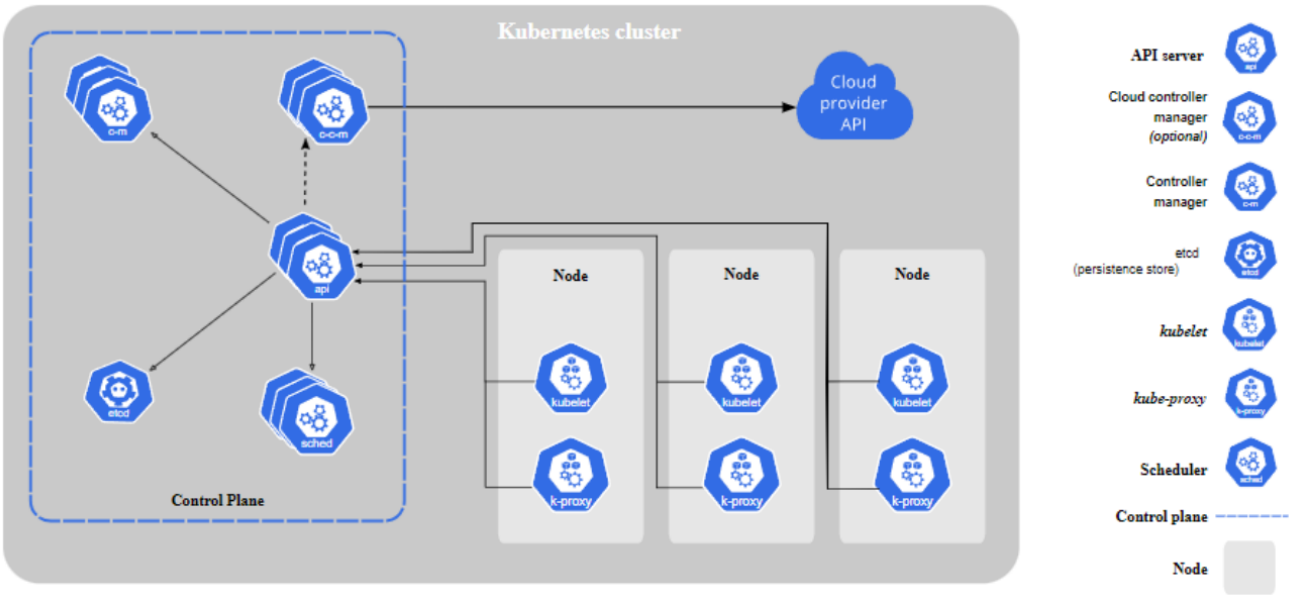


Рисунок 1.1 – Основні компоненти кластера Kubernetes

Більш детальну схему архітектури кластера із взаємозв'язками між керувальною площиною, робочими вузлами та контейнеризованими застосунками представлено на рис. 1.2.

Под (Pod) є найменшою одиницею розгортання в Kubernetes і являє собою один екземпляр виконуваного процесу, який може містити один або декілька контейнерів зі спільними мережевими, обчислювальними та файловими ресурсами. Поди мають тимчасовий характер існування (ephemeral) і, як правило, керуються за допомогою абстракцій вищого рівня, таких як Deployment або StatefulSet [2].

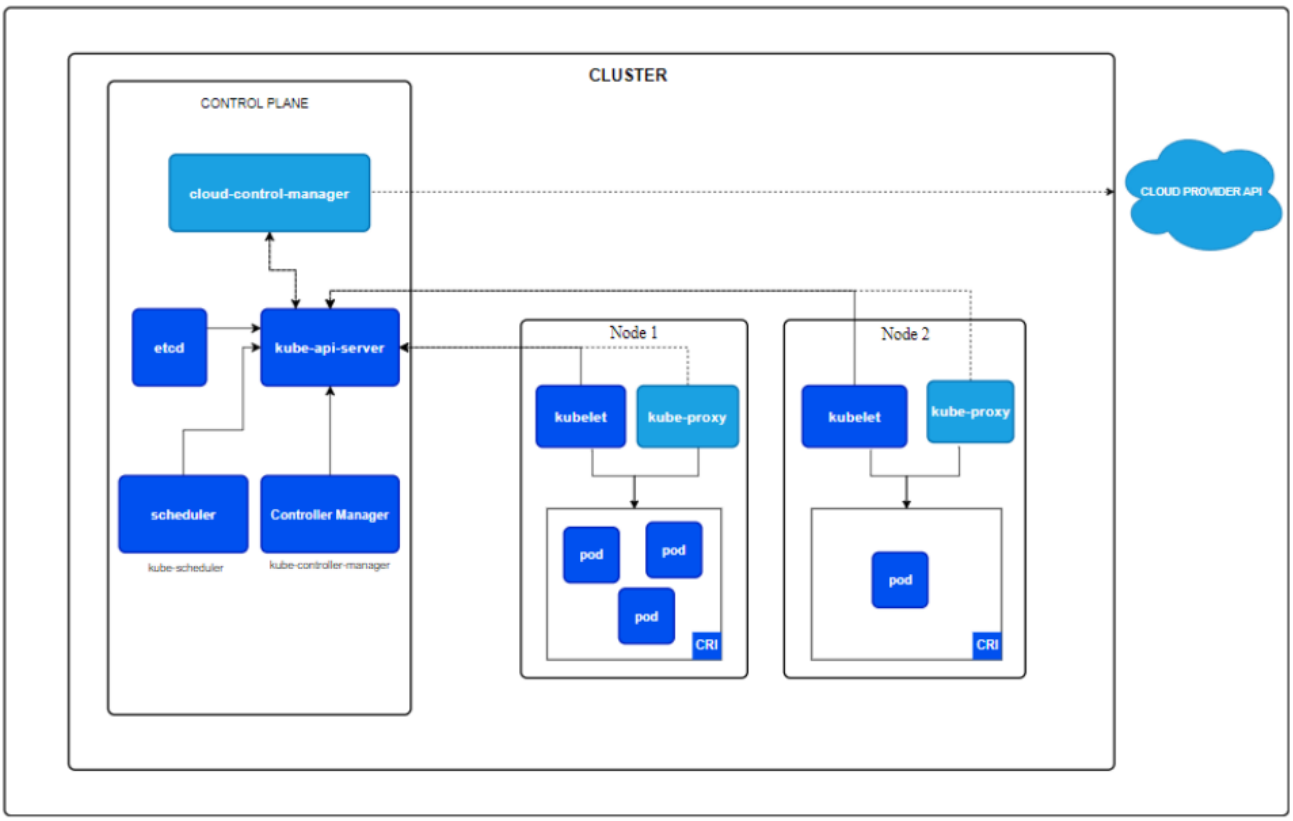


Рисунок 1.2 – Архітектура кластера Kubernetes та взаємодія його компонентів

Сервіси (Services) забезпечують мережевий доступ до групи подів, приховуючи їхню динамічну природу. Kubernetes підтримує автоматичне виявлення сервісів за допомогою DNS та виконує балансування навантаження між репліками подів, що забезпечує безперервність роботи застосунків навіть у разі зміни окремих екземплярів [3].

Мережева архітектура Kubernetes побудована на моделі єдиного плоского адресного простору, що забезпечує безперешкодну взаємодію між усіма подами кластера. Такий підхід є особливо важливим для мікросервісної архітектури, оскільки спрощує обмін даними між компонентами та підвищує загальну продуктивність системи.

Кожному поду автоматично призначається унікальна IP-адреса, що дозволяє здійснювати прямий обмін пакетами між подами без використання механізму трансляції мережевих адрес (NAT). Відсутність необхідності у NAT спрощує конфігурацію мережі, зменшує складність адміністрування та

забезпечує ефективну передачу даних із мінімальними затримками.

Для забезпечення стабільної взаємодії між компонентами Kubernetes використовує сервіси, які надають подам постійні DNS-імена або віртуальні IP-адреси. Завдяки цьому інші сервіси та клієнти можуть надійно звертатися до необхідних застосунків незалежно від зміни IP-адрес окремих подів у процесі їх створення або видалення [4].

Разом із перевагами така мережева модель створює додаткові ризики інформаційної безпеки. За замовчуванням усі поди можуть взаємодіяти між собою, тому за відсутності належним чином налаштованих Network Policies виникає можливість несанкціонованого доступу між компонентами системи. З цієї причини правила мережевої політики повинні ретельно проєктуватися та впроваджуватися для обмеження взаємодії між окремими подами й просторами імен [5].

Однією з ключових можливостей Kubernetes є автоматичне горизонтальне масштабування застосунків (Horizontal Pod Autoscaler, HPA). Цей механізм автоматично змінює кількість реплік подів відповідно до завантаження процесора або інших користувацьких метрик, що надходять від Metrics Server. Використання HPA дозволяє системі адаптуватися до змін навантаження без втручання адміністратора, підвищуючи її продуктивність, відмовостійкість та ефективність використання ресурсів [6].

Динамічне створення та видалення подів, постійний перерозподіл IP-адрес і безперервне підтримання бажаного стану кластера роблять Kubernetes високодинамічним середовищем. Незважаючи на значні переваги щодо гнучкості та масштабованості, така мінливість істотно ускладнює моніторинг інформаційної безпеки, особливо для систем виявлення вторгнень.

Однією з характерних особливостей Kubernetes є динамічне планування та перепланування компонентів застосунків. Поди, як найменші одиниці розгортання, створюються, перезапускаються, переміщуються між вузлами та видаляються залежно від поточного навантаження або змін конфігурації. Через це традиційним засобам захисту складно забезпечувати довготривале

відстеження ресурсів і поведінки компонентів кластера [7].

Системи безпеки, які використовують статичні ідентифікатори, наприклад IP-адреси або імена вузлів, не здатні ефективно працювати в умовах швидкої зміни робочих навантажень. Зловмисна активність може залишитися непоміченою, якщо под, у якому вона виникла, буде видалений або переміщений ще до завершення процесу аналізу системою виявлення вторгнень.

За замовчуванням Kubernetes не забезпечує жорсткої ізоляції між контейнерами або просторами імен. Хоча платформа підтримує механізми захисту, зокрема Pod Security Admission, а також засоби безпеки ядра Linux, такі як AppArmor і Seccomp, їх необхідно окремо активувати та правильно налаштовувати [8].

Особливо небезпечними є контейнери, які виконуються з розширеними привілеями або використовують спільні простори імен із хостовою операційною системою. За таких умов зловмисник може виконати підвищення привілеїв, вийти за межі контейнера та отримати доступ до ресурсів вузла, на якому він виконується.

Поширення технологій Service Mesh, зокрема Istio, а також використання взаємної автентифікації із шифруванням (mutual TLS, mTLS) суттєво підвищило конфіденційність внутрішнього мережевого трафіку Kubernetes-кластерів. Водночас це зменшує ефективність традиційних систем виявлення вторгнень, що виконують глибокий аналіз мережевих пакетів (Deep Packet Inspection, DPI) .

Оскільки корисне навантаження мережевих пакетів шифрується, традиційні методи аналізу пакетів не можуть безпосередньо виявляти шкідливі команди або факти несанкціонованого виведення даних. У зв'язку з цим сучасні системи виявлення вторгнень у середовищі Kubernetes дедалі частіше використовують альтернативні джерела інформації, зокрема метадані мережевих потоків, журнали аудиту Kubernetes, телеметрію sidecar-проксі та інші засоби спостереження за поведінкою компонентів кластера.

1.2 Системи виявлення вторгнень

Система виявлення вторгнень (Intrusion Detection System, IDS) – це засіб інформаційної безпеки, призначений для моніторингу мережевого трафіку або стану інформаційної системи з метою виявлення несанкціонованого доступу, шкідливих дій або порушень політик безпеки [9]. На відміну від міжмережевих екранів, які блокують небажаний трафік, IDS виконує функції спостереження, аналізу та формування повідомлень про потенційні загрози. У хмарних середовищах, зокрема в кластерах Kubernetes, системи виявлення вторгнень є важливим елементом захисту, оскільки дозволяють виявляти внутрішні атаки, які можуть обходити традиційні засоби периметрової безпеки.

Системи виявлення вторгнень класифікують за місцем розгортання та принципом виявлення загроз.

Хостові системи виявлення вторгнень (Host-based IDS, HIDS) здійснюють моніторинг окремого вузла або обчислювального пристрою з метою виявлення ознак модифікації системи, порушення політик безпеки чи несанкціонованих адміністративних дій. Для цього на вузлі встановлюється програмний агент, який безперервно аналізує системні виклики, поведінку застосунків, журнали подій та цілісність файлів. Робота HIDS ґрунтується на зіставленні отриманих даних із набором правил, евристик або моделей нормальної поведінки.

Мережеві системи виявлення вторгнень (Network-based IDS, NIDS) аналізують мережевий трафік між вузлами або компонентами інформаційної системи. Вони дозволяють виявляти такі загрози, як сканування портів, розподілені атаки типу «відмова в обслуговуванні» (DDoS), експлуатацію відомих вразливостей та інші мережеві атаки. Робота NIDS полягає у перехопленні мережевого трафіку та його порівнянні з базою відомих сигнатур або статистичними моделями аномальної поведінки. У середовищі Kubernetes мережеві системи виявлення вторгнень можуть бути розгорнуті у привілейованих подах, які мають можливість аналізувати мережеву взаємодію між іншими подами та вузлами кластера. Саме такий підхід використовується в

даній роботі, оскільки він забезпечує безпосередній контроль мережевої взаємодії між контейнеризованими застосунками.

За принципом аналізу подій системи IDS поділяються на сигнатурні та аномалійні.

Сигнатурні системи виявлення вторгнень (Signature-based IDS) працюють шляхом пошуку відомих шаблонів атак у мережевому трафіку або поведінці системи. Такі шаблони (сигнатури) являють собою набір правил, що описують характерні ознаки вже відомих загроз. Сигнатурний підхід забезпечує високу точність виявлення відомих атак завдяки використанню бази сигнатур. Принцип його роботи полягає у порівнянні поточного мережевого трафіку або подій із визначеними правилами. Якщо виявлено збіг із відомою сигнатурою, система формує повідомлення про потенційне вторгнення. Основним недоліком цього підходу є неможливість ефективного виявлення нових або раніше невідомих атак, для яких відповідні сигнатури ще не створені.

Системи виявлення аномалій (Anomaly-based IDS) функціонують шляхом формування моделі нормальної поведінки інформаційної системи. Подальший аналіз здійснюється через пошук відхилень від сформованої моделі, які можуть свідчити про потенційні загрози. Для реалізації такого підходу широко використовуються методи машинного навчання, що дозволяють виявляти не лише відомі атаки, а й їх модифікації або принципово нові способи компрометації системи . Водночас характерною особливістю аномалійного підходу є підвищена ймовірність хибних спрацьовувань, оскільки окремі легітимні зміни поведінки можуть бути інтерпретовані як ознаки атаки.

Особливості архітектури Kubernetes створюють низку додаткових труднощів для розгортання та ефективного функціонування систем виявлення вторгнень.

Однією з основних проблем є динамічність робочих навантажень. Поди мають тимчасовий характер існування, постійно створюються, видаляються та отримують нові IP-адреси. Унаслідок цього система виявлення вторгнень повинна в режимі реального часу встановлювати відповідність між мережевим

трафіком і конкретними подами.

Другою проблемою є шифрування мережевого трафіку. Використання взаємної автентифікації з шифруванням (mTLS) у поєднанні із сервісними мережами (Service Mesh) значно ускладнює аналіз вмісту мережевих пакетів. У таких умовах більш ефективним підходом є аналіз метаданих мережевих потоків, що дозволяє виявляти аномальну поведінку навіть без доступу до корисного навантаження пакетів.

Ще однією особливістю є необхідність отримання розширених привілеїв для перехоплення мережевого трафіку. Для аналізу мережевої взаємодії між контейнерами система IDS повинна мати доступ до мережевого простору вузла. У межах даної роботи ця проблема вирішується шляхом використання привілейованого пода, який здійснює перехоплення мережевих пакетів на рівні вузла кластера, не порушуючи при цьому ізоляцію прикладних контейнерів.

З метою подолання зазначених обмежень у даній роботі запропоновано гібридний підхід до побудови системи виявлення вторгнень. Він поєднує сигнатурний аналіз із використанням системи Snort та методи машинного навчання для реалізації як контрольованої класифікації, так і виявлення аномалій. Таке поєднання дозволяє забезпечити ефективне виявлення як відомих типів атак, так і нових або модифікованих загроз, що є особливо важливим для динамічного середовища Kubernetes

1.3 Мережні потоки в системах виявлення вторгнень

Під час обміну даними між застосунками мережею формується значний обсяг пакетів. Детальний аналіз кожного пакета (Deep Packet Inspection, DPI) забезпечує повну інформацію про мережеву взаємодію, однак потребує значних обчислювальних ресурсів і може негативно впливати на продуктивність системи. Альтернативним підходом є використання мережевих потоків (network flows), які узагальнюють інформацію про сеанс зв'язку між мережевими вузлами та забезпечують ефективний аналіз мережевої активності.

Мережевий потік являє собою односпрямований або двонапрямлений опис мережевого сеансу між двома вузлами. Як правило, до його складу входять такі характеристики:

- IP-адреси джерела та призначення;
- номери портів джерела та призначення;
- використовуваний транспортний протокол (TCP, UDP тощо);
- тривалість з'єднання;
- кількість переданих пакетів;
- загальний обсяг переданих даних.

На відміну від аналізу окремих пакетів, мережевий потік не містить корисного навантаження пакетів, а лише описує характеристики комунікації між вузлами. За аналогією з телефонним рахунком, мережевий потік відображає, хто, коли, як довго та в якому обсязі обмінювався інформацією, але не містить самого змісту переданих повідомлень [15].

Використання потокового аналізу набуває особливої актуальності в середовищах Kubernetes, де традиційні системи аналізу пакетів стикаються з низкою обмежень.

По-перше, робочі навантаження Kubernetes є динамічними: поди постійно створюються, видаляються та переміщуються між вузлами кластера. Це призводить до безперервної зміни мережевих інтерфейсів і появи великої кількості короткочасних мережевих сеансів.

По-друге, сучасні Kubernetes-кластери широко використовують механізми шифрування мережевого трафіку, зокрема mutual TLS (mTLS), які практично унеможливлюють аналіз корисного навантаження пакетів традиційними засобами DPI.

У таких умовах системи виявлення вторгнень на основі мережевих потоків (Flow-Based IDS) аналізують не вміст пакетів, а їхні метадані. Це дозволяє зберігати повну видимість мережевої взаємодії незалежно від використання шифрування.

Потоковий аналіз добре підходить для мікросервісної архітектури

Kubernetes, оскільки дозволяє відстежувати поведінку окремих сервісів і виявляти аномальні шаблони взаємодії між ними протягом часу. Крім того, агрегування потокових даних значно зменшує обсяг інформації, що аналізується, забезпечуючи високу масштабованість системи та мінімальне навантаження на обчислювальні ресурси.

Ще однією перевагою потокового підходу є його сумісність із сучасними методами машинного навчання. Метадані потоків можуть використовуватися як ознаки для алгоритмів класифікації та виявлення аномалій, що дозволяє автоматично виявляти як відомі, так і нові типи атак.

На відміну від пакетних систем, потокові IDS залишаються ефективними навіть при використанні шифрованого трафіку, оскільки для аналізу використовуються службові характеристики мережевих потоків (IP-адреси, порти, тривалість з'єднання, кількість пакетів, обсяг переданих даних тощо), а не вміст переданої інформації.

Порівняльну характеристику систем виявлення вторгнень на основі пакетного та потокового аналізу наведено в табл. 1.1.

Таблиця 1.1 – Порівняння пакетних та потокових систем виявлення вторгнень

Характеристика	Пакетна IDS	Потокова IDS
Обсяг даних для аналізу	Високий	Низький
Аналіз корисного навантаження пакетів	Так	Ні
Видимість шифрованого TLS-трафіку	Ні	Так (за метаданими)
Навантаження на систему	Високе	Низьке
Конфіденційність	Нижча	Вища
Сумісність із Kubernetes	Середня	Висока

Як показано в табл. 1.1, потокові системи не аналізують безпосередньо вміст пакетів, проте забезпечують оптимальний баланс між продуктивністю, конфіденційністю та масштабованістю, що робить їх особливо придатними для

побудови систем виявлення вторгнень у Kubernetes-кластерах.

Для автоматичного аналізу мережевих потоків у сучасних IDS широко застосовуються методи машинного навчання, які можна поділити на контрольоване (Supervised Learning) та неконтрольоване (Unsupervised Learning).

Контрольоване навчання передбачає навчання моделі на розміченому наборі даних, у якому кожний мережевий потік заздалегідь позначений як легітимний або шкідливий. На основі цих прикладів модель формує правила класифікації, які використовуються для аналізу нових даних.

Основними перевагами контрольованого навчання є:

- висока точність виявлення відомих типів атак;
- можливість налаштування моделей на конкретні категорії загроз;
- простота кількісного оцінювання якості моделей за допомогою показників Precision, Recall та F1-score.

Водночас цей підхід має і певні недоліки:

- потребує великих обсягів якісно розмічених даних;
- ефективність суттєво знижується під час появи нових або невідомих атак, відсутніх у навчальній вибірці.

На відміну від цього, неконтрольоване навчання працює без використання розмічених даних. Його основною метою є побудова моделі нормальної поведінки мережі та виявлення статистичних відхилень від неї. Для цього використовуються методи кластеризації, зменшення розмірності простору ознак та інші алгоритми аналізу даних.

Основними перевагами неконтрольованого навчання є:

- можливість виявлення атак нульового дня та раніше невідомих загроз;
- відсутність потреби у трудомісткому процесі розмічування навчальних даних;
- придатність для аналізу великих потоків даних у реальному часі.

До недоліків цього підходу належать:

- більша кількість хибнопозитивних спрацювань;

- складність інтерпретації результатів роботи моделей.

Таким чином, у системах виявлення вторгнень для Kubernetes доцільно поєднувати потоковий аналіз мережевого трафіку з методами машинного навчання. Такий підхід дозволяє забезпечити високу масштабованість системи, ефективний аналіз шифрованого мережевого трафіку та можливість виявлення як відомих, так і нових типів кібератак.

Порівняльну характеристику методів контрольованого та неконтрольованого машинного навчання для задач виявлення вторгнень наведено в табл. 1.2.

Таблиця 1.2 – Порівняння методів контрольованого та неконтрольованого навчання в системах виявлення вторгнень

Характеристика	Контрольоване навчання	Неконтрольоване навчання
Необхідність розмічених даних	Потрібні	Не потрібні
Виявлення атак нульового дня	Низька ефективність	Висока ефективність
Точність виявлення відомих атак	Висока	Середня
Кількість хибнопозитивних спрацювань	Низька	Вища
Інтерпретованість результатів	Вища	Нижча
Складність навчання моделі	Висока	Середня

Як видно з табл. 1.2, контрольоване навчання забезпечує високу точність класифікації відомих типів атак завдяки використанню попередньо розмічених навчальних вибірок. Проте його ефективність істотно знижується під час появи нових або невідомих атак, оскільки модель не має відповідних прикладів для навчання.

Неконтрольоване навчання, навпаки, не потребує розмічених даних і здатне виявляти нові типи атак шляхом аналізу відхилень від нормальної

поведінки системи. Разом із тим використання такого підходу зазвичай супроводжується більшою кількістю хибнопозитивних спрацювань та ускладнює інтерпретацію отриманих результатів.

Порівняння, наведене в табл. 1.2, свідчить, що жоден із розглянутих підходів не є універсальним. Контрольоване навчання демонструє високу ефективність під час виявлення відомих загроз, тоді як неконтрольоване навчання краще пристосоване до пошуку нових, раніше невідомих атак. Саме тому останніми роками все більшого поширення набувають гібридні системи виявлення вторгнень, які поєднують переваги обох методів.

Поєднання моделей контрольованого та неконтрольованого машинного навчання дозволяє компенсувати недоліки кожного з підходів окремо. Контрольована модель забезпечує точне розпізнавання відомих атак, а модель виявлення аномалій доповнює її можливістю виявлення нових або модифікованих загроз, які ще відсутні у навчальних даних. Такий комплексний підхід забезпечує більш повне охоплення потенційних загроз і є особливо доцільним для кластерів Kubernetes, де динамічна зміна робочих навантажень, короткочасність існування подів та широкий спектр можливих векторів атак суттєво ускладнюють використання лише одного методу аналізу.

2 ОГЛЯД ЗАСТОСУВАННЯ МАШИННОГО НАВЧАННЯ ДЛЯ ВИЯВЛЕННЯ ВТОРГНЕНЬ У МЕРЕЖНОМУ ТРАФІКУ

2.1 Аналіз алгоритмів машинного навчання для виявлення вторгнень у мережевому трафіку

Стрімкий розвиток інформаційних технологій, збільшення обсягів мережевого трафіку та ускладнення сучасних кібератак зумовили широке впровадження методів машинного навчання у системах інформаційної безпеки. На сьогодні такі методи є одним із найефективніших інструментів автоматизованого виявлення вторгнень, оскільки дозволяють аналізувати закономірності мережевої активності, виявляти приховані взаємозв'язки між параметрами трафіку та своєчасно реагувати на появу нових загроз. На відміну від класичних сигнатурних механізмів, алгоритми машинного навчання здатні працювати з багатовимірними наборами ознак і виявляти аномальні шаблони поведінки, які можуть свідчити про реалізацію невідомих раніше атак. Саме тому сучасні системи виявлення вторгнень дедалі частіше використовують модулі поведінкового аналізу, побудовані на основі методів машинного навчання.

Важливе місце серед таких методів займають алгоритми класифікації, основним призначенням яких є визначення належності мережевого трафіку до класу легітимного або шкідливого. Для розв'язання цієї задачі застосовують різноманітні алгоритми, серед яких дерева рішень, метод опорних векторів (SVM), логістична регресія, наївний байєсівський класифікатор, алгоритм k-Nearest Neighbors (KNN), ансамблеві методи та нейронні мережі. Кожен із наведених підходів має власні особливості використання, які визначаються характером навчальної вибірки, структурою ознак, складністю розділення класів, а також вимогами до швидкодії та обчислювальних ресурсів.

Одними з найбільш зрозумілих і простих для інтерпретації є дерева

рішень. Їх застосування дозволяє будувати послідовність правил, за допомогою яких здійснюється класифікація мережевих потоків. Перевагами такого підходу є висока швидкість прийняття рішень та можливість пояснення результатів класифікації. Разом із тим дерева рішень можуть демонструвати схильність до перенавчання, особливо під час роботи зі складними наборами даних. Для підвищення стійкості моделей використовуються ансамблеві алгоритми, зокрема Random Forest та Gradient Boosting. Перший формує множину незалежних дерев рішень, після чого остаточний результат визначається за принципом голосування, що забезпечує високу точність і стійкість до шуму. Другий підхід послідовно навчає дерева, кожне з яких компенсує помилки попереднього, дозволяючи досягати високої ефективності навіть під час розв'язання складних задач багатокласової класифікації.

Метод опорних векторів також широко використовується при побудові систем виявлення вторгнень. Його головною перевагою є здатність знаходити оптимальну межу між класами навіть у просторах великої розмірності, що робить алгоритм ефективним для аналізу складних мережевих даних. Проте процес навчання моделей SVM характеризується значними обчислювальними витратами, через що їх застосування для аналізу великих масивів мережевого трафіку може бути обмеженим.

Важливим напрямом досліджень є також використання алгоритмів кластеризації, які не потребують попередньо розмічених даних. До найбільш відомих методів цього класу належать k-means, DBSCAN та інші алгоритми, засновані на аналізі щільності розподілу даних. Такі підходи дозволяють сформувати модель нормальної поведінки мережевого середовища та виявляти суттєві відхилення від неї, що можуть бути ознаками кібератаки. Їх використання є особливо доцільним під час пошуку нових або модифікованих загроз, відсутніх у навчальних наборах даних.

Не менш важливим етапом підготовки даних є зменшення розмірності простору ознак. Для цього широко застосовуються методи Principal Component Analysis (PCA) та t-distributed Stochastic Neighbor Embedding (t-SNE).

Використання цих алгоритмів дає змогу виділити найбільш інформативні характеристики мережевого трафіку, усунути надлишкові або шумові ознаки та зменшити складність подальшого навчання моделей. Це особливо актуально для сучасних наборів мережевих даних, які можуть містити сотні параметрів і потребують попереднього відбору інформативних характеристик.

Ефективність алгоритмів машинного навчання значною мірою визначається якістю сформованого набору ознак. Для аналізу мережевого трафіку найчастіше використовуються кількість пакетів, обсяг переданих даних, тривалість мережевих з'єднань, число активних сеансів, напрямок передавання трафіку, часові характеристики міжпакетних інтервалів, статистичні параметри потоків, а також інформація про використовувані мережеві протоколи. Чим повніше такі характеристики описують поведінку мережевого потоку, тим вищою є точність класифікації та надійність функціонування системи виявлення вторгнень.

Отже, сучасні алгоритми машинного навчання створюють потужне підґрунтя для побудови інтелектуальних систем виявлення вторгнень. Їх здатність аналізувати великі обсяги мережевих даних, адаптуватися до змін у характері трафіку та виявляти складні закономірності дозволяє суттєво підвищити ефективність сучасних IDS порівняно з традиційними сигнатурними методами захисту.

2.2 Особливості застосування машинного навчання в системах виявлення вторгнень

Сучасний розвиток інформаційних технологій супроводжується постійним ускладненням кіберзагроз та зростанням кількості атак на інформаційні системи. Використання автоматизованих засобів, складних сценаріїв проникнення та методів приховування шкідливої активності суттєво ускладнює роботу традиційних механізмів захисту. Засоби безпеки, що базуються виключно на сигнатурному аналізі або фіксованих правилах, не

завжди здатні своєчасно реагувати на появу нових видів атак. У зв'язку з цим методи машинного навчання стали одним із ключових напрямів розвитку сучасних систем кіберзахисту, оскільки забезпечують можливість адаптації до змін поведінки користувачів, мережевих сервісів і потенційних зловмисників.

Одним із найважливіших напрямів використання машинного навчання є побудова систем виявлення аномальної активності. На відміну від сигнатурних підходів, такі методи не обмежуються пошуком заздалегідь відомих шаблонів атак, а формують модель нормального функціонування мережі та визначають будь-які статистично значущі відхилення від неї. Це дозволяє виявляти атаки нульового дня, несанкціоновані дії внутрішніх користувачів, а також складні багатоступеневі сценарії проникнення, які практично неможливо ідентифікувати традиційними засобами захисту. Крім цього, моделі машинного навчання використовуються для формування поведінкових профілів користувачів, вузлів і сервісів, що дає змогу своєчасно виявляти нетипові зміни у функціонуванні інформаційної системи.

Ще одним важливим напрямом є аналіз різнорідних джерел інформації, серед яких журнали подій, мережеві потоки, записи NetFlow, телеметрія системних процесів та журнали аудиту. Використання алгоритмів машинного навчання дозволяє автоматично визначати рівень небезпеки подій, виявляти приховані взаємозв'язки між окремими інцидентами та формувати цілісне уявлення про поточний стан інформаційної безпеки мережі. Особливе значення це має для сучасних центрів моніторингу безпеки (SOC), де обсяг подій значно перевищує можливості їх ручного аналізу.

Методи машинного навчання також широко використовуються для виявлення шкідливого програмного забезпечення. Навчені моделі здатні аналізувати характеристики виконуваних файлів і розпізнавати шкідливі об'єкти навіть після їх модифікації або застосування методів обфускації. Особливо ефективними у таких задачах є глибокі нейронні мережі, які можуть аналізувати внутрішню структуру програмних файлів та знаходити приховані закономірності, недоступні традиційним сигнатурним антивірусним рішенням.

Використання подібних моделей дозволяє скоротити час реагування на нові загрози та підвищити загальний рівень захищеності інформаційних систем.

Важливе місце машинне навчання займає й у сфері аналізу мережевого трафіку. Сучасні алгоритми використовуються для класифікації мережевих потоків, розпізнавання підозрілих шаблонів взаємодії між вузлами, оцінювання рівня ризику інцидентів та автоматичного пріоритезування подій безпеки. Особливого значення такі підходи набувають у сучасних мережах, де переважна частина трафіку передається із застосуванням криптографічного захисту. Поширення протоколів TLS 1.3 та інших механізмів шифрування значно обмежує можливості класичного аналізу вмісту пакетів. За таких умов моделі машинного навчання можуть ефективно працювати з метаданими мережевих потоків, використовуючи часові характеристики, статистику передачі пакетів, тривалість сеансів, обсяг переданих даних та інші непрямі ознаки для виявлення потенційно небезпечної активності.

Перспективним напрямом розвитку є застосування методів машинного навчання для прогнозування кібератак. На основі історичних даних сучасні статистичні моделі, а також рекурентні нейронні мережі здатні оцінювати ймовірність виникнення майбутніх інцидентів інформаційної безпеки. Це створює можливість переходу від реактивної моделі захисту до проактивного підходу, коли потенційні загрози можуть бути виявлені ще до моменту реалізації атаки. Подібні технології сьогодні розглядаються як один із ключових елементів розвитку інтелектуальних систем кібербезпеки нового покоління.

Таким чином, машинне навчання стало одним із фундаментальних компонентів сучасних систем захисту інформації. Використання інтелектуальних алгоритмів дає змогу автоматизувати процеси аналізу мережевого трафіку, підвищити точність виявлення атак, скоротити кількість хибнопозитивних спрацьовувань та забезпечити ефективне функціонування систем виявлення вторгнень навіть в умовах швидкої зміни характеру сучасних кіберзагроз. Застосування таких методів є особливо актуальним для динамічних

середовищ, зокрема кластерів Kubernetes, де традиційні сигнатурні механізми не завжди здатні забезпечити необхідний рівень захисту.

2.3 Використання нейронних мереж для аналізу мережевого трафіку та виявлення атак

Стрімкий розвиток технологій глибокого навчання сприяв широкому впровадженню нейронних мереж у системи виявлення вторгнень. Завдяки здатності працювати з великими обсягами даних і моделювати складні нелінійні залежності такі моделі стали одним із найбільш перспективних інструментів аналізу мережевого трафіку. На відміну від традиційних алгоритмів машинного навчання, нейронні мережі можуть автоматично знаходити приховані закономірності між великою кількістю ознак, що є особливо важливим під час аналізу сучасних мереж, де характер трафіку постійно змінюється, а сценарії кібератак стають дедалі складнішими .

Одним із найпоширеніших різновидів нейронних мереж є багатошаровий перцептрон (Multilayer Perceptron, MLP). Такі моделі широко використовуються для класифікації мережевих потоків і дозволяють розпізнавати різні категорії атак на основі значної кількості статистичних характеристик трафіку. Простота реалізації, достатньо висока точність і можливість навчання на розмічених вибірках роблять перцептрони ефективним інструментом для побудови систем виявлення вторгнень. Разом із тим такі мережі аналізують окремі приклади незалежно один від одного і практично не враховують часові залежності між подіями, що обмежує їх ефективність під час аналізу потокового мережевого трафіку.

Для оброблення часових послідовностей значно кращі результати демонструють рекурентні нейронні мережі (Recurrent Neural Networks, RNN), насамперед архітектури LSTM та GRU. Вони здатні зберігати інформацію про попередні стани системи, що дозволяє аналізувати розвиток мережевих подій у часі та виявляти довготривалі закономірності. Подібні моделі особливо

ефективні при виявленні атак, які формуються поступово, наприклад повільного сканування портів, низькоінтенсивних DDoS-атак або інших сценаріїв, де окремі пакети не є підозрілими, проте їх послідовність свідчить про шкідливу активність.

Важливе місце серед сучасних методів аналізу мережевого трафіку займають згорткові нейронні мережі (Convolutional Neural Networks, CNN). Хоча спочатку вони були розроблені для задач комп'ютерного зору, сьогодні їх успішно використовують і для оброблення структурованих мережевих даних. Застосування згорткових шарів дозволяє автоматично знаходити локальні закономірності у потоках даних, серед яких характерні послідовності TCP-прапорців, нетипові розподіли розмірів пакетів, зміни інтенсивності передавання інформації та інші особливості мережевої активності. Проведені дослідження підтверджують, що CNN забезпечують високу точність класифікації складних кібератак, особливо у складі комбінованих моделей глибокого навчання.

Ще одним перспективним напрямом є використання автоенкодерів (Autoencoder) для виявлення мережевих аномалій. На відміну від класичних класифікаторів, автоенкодери навчаються лише на прикладах нормального мережевого трафіку, формуючи його компактне приховане представлення. Після завершення навчання модель виконує реконструкцію вхідних даних. Якщо характеристики нового мережевого потоку істотно відрізняються від звичного профілю, помилка реконструкції різко збільшується, що використовується як критерій виявлення аномалії. Подібний підхід дозволяє ефективно знаходити нові або невідомі раніше атаки без необхідності формування великих розмічених навчальних вибірок.

Для підвищення якості навчання моделей машинного навчання дедалі частіше використовуються генеративно-змагальні мережі (Generative Adversarial Networks, GAN). Їх основним призначенням є синтез нових прикладів даних, які за своїми характеристиками максимально наближені до реальних мережевих потоків. Це особливо актуально для задач інформаційної

безпеки, оскільки більшість навчальних наборів містить значний дисбаланс між нормальним і атакувальним трафіком. Використання GAN дозволяє збільшити кількість прикладів рідкісних атак, покращити збалансованість навчальної вибірки та підвищити якість подальшої класифікації.

Окрім безпосереднього аналізу мережевого трафіку, нейронні мережі широко застосовуються для кореляції подій інформаційної безпеки та побудови моделей поведінки користувачів і інформаційних систем. Глибокі моделі здатні об'єднувати інформацію з різних джерел, серед яких журнали подій IDS, системні журнали, мережеві потоки, телеметрія контейнерів та інші елементи інформаційної інфраструктури. Комплексний аналіз таких даних дозволяє формувати цілісну картину розвитку атаки, визначати взаємозв'язки між окремими інцидентами та автоматизувати процес прийняття рішень щодо реагування на кіберзагрози.

Отже, нейронні мережі є одним із найбільш ефективних напрямів розвитку сучасних систем виявлення вторгнень. Їхня здатність аналізувати великі обсяги різнорідних даних, виявляти приховані нелінійні залежності, враховувати часові закономірності та автоматично формувати інформативні ознаки забезпечує високий рівень точності виявлення сучасних кібератак. Саме тому технології глибокого навчання сьогодні розглядаються як один із ключових компонентів інтелектуальних IDS нового покоління, у тому числі призначених для захисту динамічних контейнерних середовищ і кластерів Kubernetes.

2.4 Використання алгоритму k-Nearest Neighbors для класифікації мережевого трафіку

Алгоритм k-Nearest Neighbors (KNN) є одним із найвідоміших методів контрольованого машинного навчання, який широко використовується для розв'язання задач класифікації у сфері аналізу мережевого трафіку та побудови систем виявлення вторгнень. Популярність цього алгоритму пояснюється

простотою реалізації, відсутністю складного процесу навчання та здатністю ефективно класифікувати дані зі складною нелінійною структурою. На відміну від більшості сучасних алгоритмів машинного навчання, KNN не будує узагальнену математичну модель під час навчання, а використовує безпосередньо навчальну вибірку для прийняття рішення щодо кожного нового об'єкта. Така особливість робить алгоритм достатньо гнучким і дозволяє швидко адаптувати його до появи нових прикладів мережевого трафіку без необхідності повторного навчання моделі.

Принцип функціонування алгоритму k-Nearest Neighbors наведено на рис. 2.1. Як показано на схемі, після надходження нового мережевого потоку формується його вектор ознак, який порівнюється з усіма прикладами навчальної вибірки. Для кожного навчального прикладу обчислюється відстань до нового об'єкта, після чого визначаються k найближчих сусідів. На завершальному етапі здійснюється голосування між знайденими сусідами, а остаточне рішення щодо належності мережевого потоку до певного класу приймається за принципом більшості голосів.

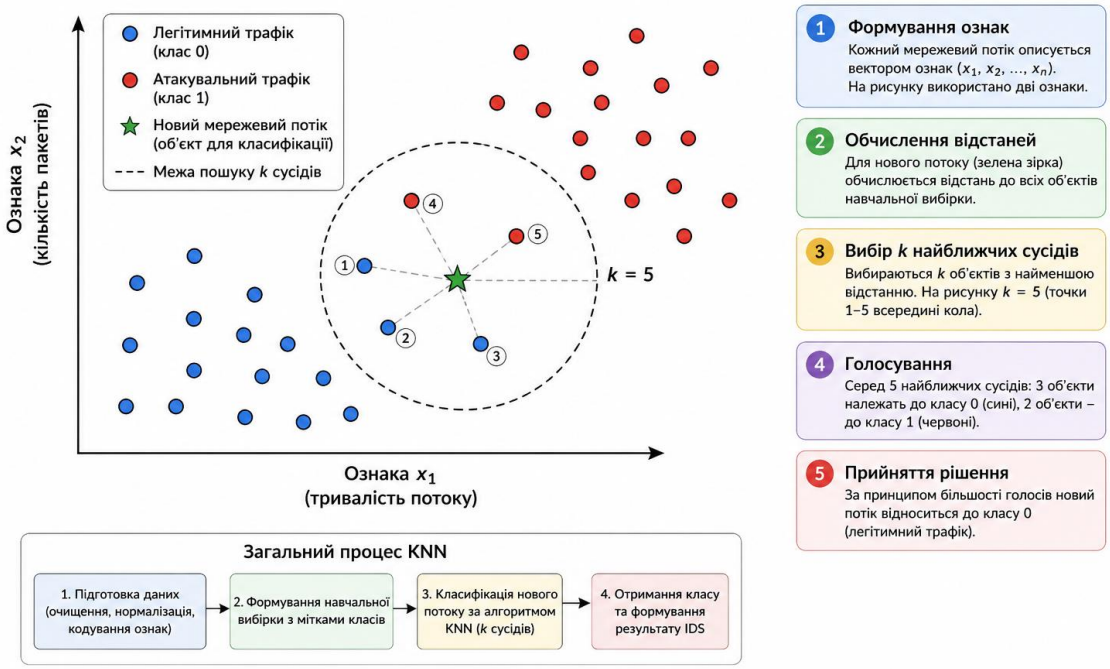


Рисунок 2.1 – Схема класифікації мережевого потоку алгоритмом k-Nearest Neighbors

На відміну від ансамблевих методів, зокрема Random Forest або Gradient Boosting, алгоритм KNN не виконує побудову дерев рішень чи інших складних моделей. Етап навчання фактично полягає лише у збереженні розміченої навчальної вибірки, тоді як основний обсяг обчислень виконується безпосередньо під час класифікації нового мережевого потоку.

Як показано на рис. 2.1, для кожного нового об'єкта здійснюється пошук найближчих прикладів у просторі ознак, після чого на основі результатів голосування формується кінцеве рішення. Такий підхід відносять до алгоритмів «ледачого навчання» (lazy learning), оскільки модель не виконує узагальнення даних до моменту надходження нового об'єкта.

Основою роботи алгоритму є визначення міри подібності між мережевими потоками. Для цього використовуються різні метрики відстані, серед яких найбільшого поширення набули евклідова, мангеттенська та косинусна відстані. У більшості задач класифікації мережевого трафіку застосовується евклідова метрика

$$d(x,y)=\sqrt{\sum_{i=1}^{n}(x_i-y_i)^2}, \qquad (2.1)$$

де x та y – вектори ознак двох мережевих потоків, а n – кількість характеристик, що використовуються під час класифікації.

Після обчислення відстані між новим об'єктом і всіма прикладами навчальної вибірки вони впорядковуються за зростанням значення цієї відстані. Алгоритм вибирає k найближчих сусідів, після чого виконується голосування між їхніми класами. Якщо більшість найближчих об'єктів належить до класу атакувального трафіку, новий мережевий потік також класифікується як потенційне вторгнення. Аналогічно, переважання легітимних прикладів свідчить про нормальний характер досліджуваного мережевого з'єднання.

Одним із найважливіших параметрів алгоритму є кількість найближчих сусідів k, яка безпосередньо впливає на точність класифікації. Використання

малого значення цього параметра підвищує чутливість моделі до випадкових шумів та аномальних спостережень, що може збільшити кількість хибнопозитивних спрацьовувань. Водночас занадто велике значення k призводить до надмірного узагальнення інформації та погіршує здатність алгоритму розділяти близькі між собою класи. Тому оптимальне значення параметра визначають експериментально за допомогою процедур крос-валідації або аналізу показників точності, повноти та F1-міри.

Для ефективної роботи алгоритму KNN особливе значення має попередня підготовка даних. Оскільки класифікація базується на обчисленні відстаней між об'єктами, усі числові ознаки повинні бути приведені до однакового масштабу. З цією метою застосовуються методи нормалізації Min-Max або стандартизації Z-score, що дозволяють уникнути домінування окремих характеристик під час обчислення відстаней.

Під час аналізу мережевого трафіку кожний потік описується сукупністю статистичних характеристик. До найбільш інформативних ознак належать IP-адреси джерела та призначення, номери транспортних портів, використовуваний мережевий протокол, тривалість з'єднання, кількість переданих пакетів і байтів, середній розмір пакета, інтенсивність передавання даних, часові інтервали між пакетами, кількість TCP-прапорців, напрямок потоку, кількість повторних передач та інші параметри, що характеризують поведінку мережевого сеансу. Перед використанням алгоритму категоріальні ознаки перетворюються у числовий формат за допомогою методів кодування, а пропущені значення усуваються або замінюються відповідними статистичними характеристиками.

Побудова навчальної вибірки передбачає агрегацію окремих мережевих пакетів у потоки та формування інтегральних характеристик за визначені часові інтервали. Джерелами інформації можуть бути журнали Suricata, Zeek, записи NetFlow, IPFIX, а також телеметрія Kubernetes-кластера. Після оброблення формується вектор ознак, який містить статистичні характеристики кожного мережевого потоку. До таких характеристик можуть входити сумарна кількість

пакетів, загальний обсяг переданих даних, середня швидкість передачі, кількість активних з'єднань, число унікальних IP-адрес, частота звернень до окремих сервісів, часові характеристики мережевої взаємодії та інші параметри, що дозволяють ефективно описати поведінку трафіку.

Практичне використання алгоритму KNN у системах виявлення вторгнень пов'язане з проблемою дисбалансу класів. У більшості реальних мереж переважна частина трафіку є легітимною, тоді як частка атак становить лише незначний відсоток від загальної кількості подій. За таких умов алгоритм може демонструвати зміщення у бік більшості. Для мінімізації цього ефекту застосовуються методи балансування даних, серед яких випадкове збільшення кількості прикладів міноритарного класу, випадкове зменшення кількості прикладів більшості, а також алгоритм SMOTE, який формує синтетичні приклади атакувального трафіку на основі існуючих даних. Додатково використовуються методи відбору найбільш інформативних ознак, що дозволяють зменшити розмірність простору ознак і скоротити час класифікації без істотної втрати точності.

Перевагами алгоритму k-Nearest Neighbors є простота реалізації, відсутність необхідності складного навчання моделі, можливість ефективної роботи зі складними нелінійними розподілами даних та висока точність класифікації за умови правильно підібраного набору ознак. Крім того, алгоритм легко адаптується до змін характеру мережевого трафіку, оскільки оновлення моделі здійснюється шляхом додавання нових прикладів до навчальної вибірки без повторного навчання.

Разом із тим використання KNN має певні обмеження. Основним недоліком є значна обчислювальна складність під час прогнозування, оскільки для кожного нового мережевого потоку необхідно обчислювати відстані до всіх навчальних прикладів. Також алгоритм є чутливим до масштабу ознак, наявності шуму та надлишкових характеристик, що потребує ретельної попередньої обробки даних і застосування методів відбору ознак.

Отже, алгоритм k-Nearest Neighbors є ефективним інструментом

класифікації мережевого трафіку та може успішно використовуватися у сучасних системах виявлення вторгнень. Його простота, можливість роботи без побудови складної математичної моделі та достатньо висока точність класифікації роблять цей алгоритм перспективним для застосування у системах захисту інформації. У контексті кластерів Kubernetes алгоритм KNN може використовуватися для аналізу мережевих потоків між подами та сервісами, забезпечуючи своєчасне виявлення аномальної активності та підвищуючи ефективність функціонування системи виявлення вторгнень.

3 РОЗРОБКА ТА ДОСЛІДЖЕННЯ СИСТЕМИ ВИЯВЛЕННЯ ВТОРГНЕНЬ У КЛАСТЕРІ KUBERNETES

3.1 Методика проведення дослідження

Дослідження, виконані в межах роботи, спрямовані на розроблення та експериментальну оцінку системи виявлення вторгнень у кластері Kubernetes із використанням сигнатурного аналізу та методів машинного навчання. Методика дослідження передбачає послідовне виконання етапів аналізу предметної області, побудови архітектури системи, формування набору даних, навчання моделей машинного навчання та оцінювання ефективності запропонованого рішення.

На початковому етапі було проведено аналіз сучасних підходів до побудови систем виявлення вторгнень, особливостей функціонування кластерів Kubernetes, а також методів аналізу мережевого трафіку. Значну увагу приділено дослідженню можливостей сигнатурних систем виявлення вторгнень, алгоритмів машинного навчання та сучасних наборів даних, які використовуються для навчання моделей класифікації мережевого трафіку.

Експериментальна частина роботи базується на перевірці працездатності запропонованої системи в умовах функціонування Kubernetes-кластера. Основною гіпотезою дослідження є припущення, що використання комбінованого підходу, який поєднує сигнатурний аналіз із методами машинного навчання, дозволить забезпечити ефективне виявлення мережевих атак у контейнеризованому середовищі.

Нульова гіпотеза полягає у тому, що запропонована система забезпечує ефективне виявлення вторгнень у кластері Kubernetes із прийнятними показниками точності класифікації. Альтернативна гіпотеза передбачає, що запропонований підхід не забезпечує необхідного рівня ефективності та не дозволяє надійно виявляти мережеві атаки.

Незалежними змінними експериментального дослідження є параметри функціонування Kubernetes-кластера, які можуть впливати на ефективність роботи системи виявлення вторгнень. До них належать спосіб захоплення мережевого трафіку, рівень доступу до вузлів кластера, використання привілейованих контейнерів, інтенсивність мережевого навантаження, тип мережевого трафіку (зашифрований або відкритий), а також особливості мережевої архітектури та сервісної взаємодії між контейнерами.

Залежними змінними виступають показники ефективності системи виявлення вторгнень, серед яких точність (Accuracy), повнота (Recall), точність позитивних прогнозів (Precision) та F1-міра. Крім цього, оцінюється ефективність сигнатурного аналізу за допомогою правил Snort та результативність алгоритму k-Nearest Neighbors при класифікації мережевого трафіку.

Під час проведення експериментів враховувалися фактори, які можуть впливати на достовірність отриманих результатів. До внутрішніх загроз належать вибір алгоритмів машинного навчання, параметрів їх налаштування, характеристик навчальної вибірки, а також особливості реалізації програмного забезпечення. Для мінімізації впливу цих факторів було проведено аналіз сучасних досліджень, обрано алгоритми, що демонструють високі показники ефективності у задачах аналізу мережевого трафіку, а також використано перевірені програмні засоби збору та оброблення даних.

До зовнішніх загроз належить обмеженість експериментального середовища. Запропонована система розроблялася для використання у Kubernetes-кластері, тому отримані результати безпосередньо характеризують саме контейнеризовані середовища і можуть потребувати додаткової перевірки під час використання в інших типах мережевих інфраструктур.

Загрозою коректності висновків може бути наявність інших алгоритмів або архітектур систем виявлення вторгнень, які забезпечують кращі показники ефективності. Для зменшення цього ризику було виконано аналіз сучасних наукових публікацій, присвячених системам виявлення вторгнень у Kubernetes,

а також проведено порівняння характеристик запропонованого підходу з існуючими рішеннями.

Загальна послідовність виконання дослідження наведена на рис. 3.1. На схемі відображено основні етапи функціонування системи – від захоплення мережевого трафіку до прийняття рішення щодо наявності вторгнення. Спочатку мережеві пакети перехоплюються безпосередньо у контейнерах Kubernetes за допомогою інструмента Ksniff, після чого передаються до Wireshark для формування файлів формату PCAP. Далі за допомогою програмного комплексу CICFlowMeter виконується перетворення пакетного трафіку у двонаправлені мережеві потоки (Flow), що містять набір статистичних характеристик, придатних для машинного навчання.

Для сигнатурного аналізу мережевого трафіку використовується система Snort, яка забезпечує виявлення відомих типів атак на основі набору правил. Паралельно сформовані мережеві потоки передаються до модуля машинного навчання, де здійснюється їх попередня обробка, нормалізація та, за необхідності, зменшення розмірності простору ознак методом головних компонент (PCA). Після цього підготовлені дані надходять до алгоритму k-Nearest Neighbors, який виконує класифікацію мережевого трафіку на нормальний або атакувальний.

Для навчання та оцінювання моделі використовується вдосконалений набір даних CICIDS2017, який містить приклади легітимного мережевого трафіку та різноманітних типів мережевих атак. Використання покращеної версії набору даних дозволяє усунути недоліки початкового датасета, пов'язані з помилками розмітки та нерівномірним розподілом класів, зберігаючи при цьому різноманітність сценаріїв мережевої взаємодії та високий рівень складності задачі класифікації.

Для оцінювання якості роботи алгоритму використовуються стандартні метрики класифікації – Accuracy, Precision, Recall та F1-score, що дозволяють комплексно оцінити ефективність запропонованої системи виявлення вторгнень та визначити її придатність для використання у Kubernetes-кластерах.

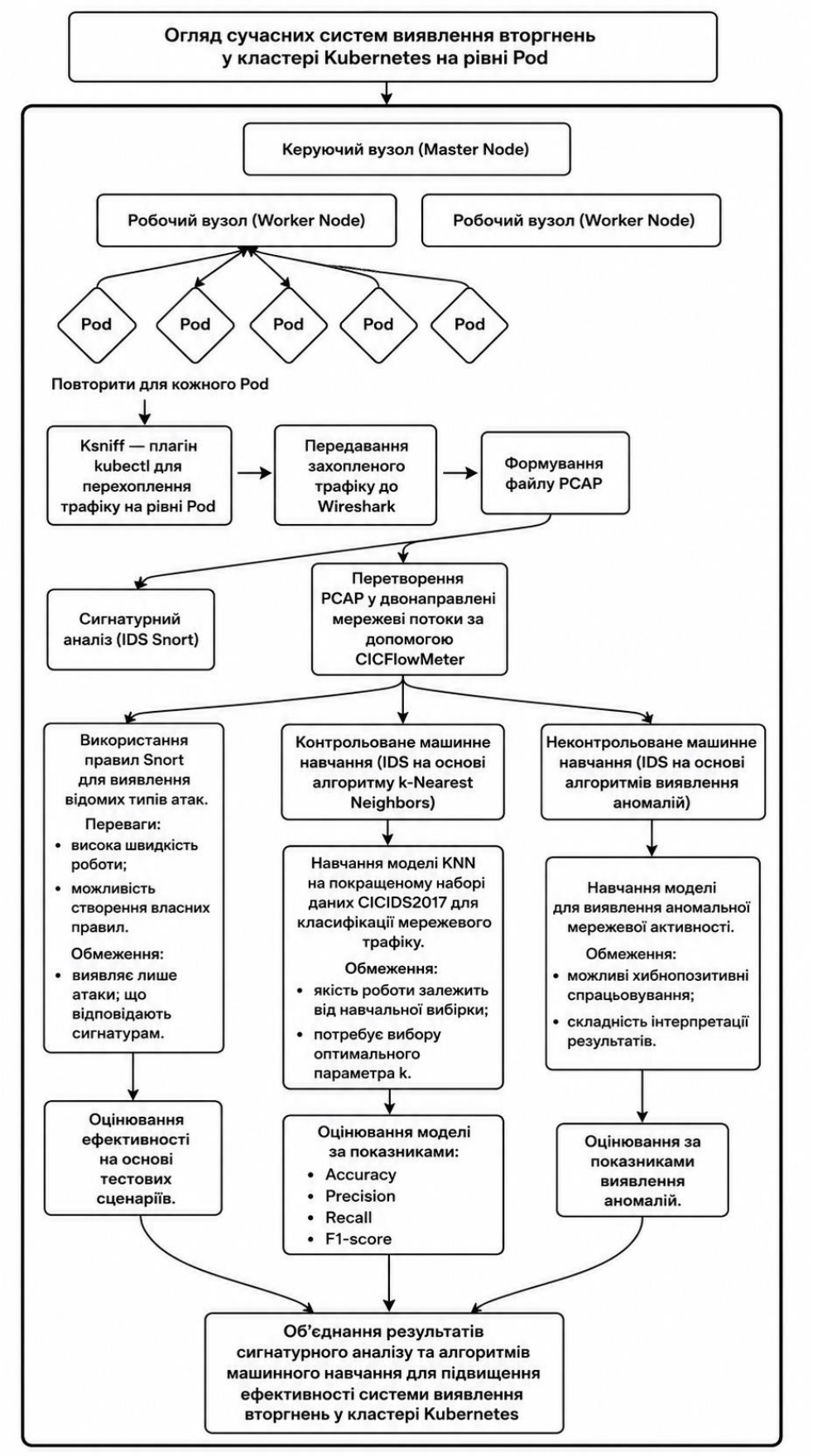


Рисунок 3.1 – Методика проведення дослідження та функціонування системи виявлення вторгнень у кластері Kubernetes

3.2 Аналіз наборів даних для навчання системи виявлення вторгнень

Одним із ключових етапів побудови системи виявлення вторгнень на основі машинного навчання є вибір навчального набору даних. Саме від якості, структури та репрезентативності даних залежить здатність моделі формувати коректні закономірності, ефективно класифікувати мережевий трафік та виявляти аномальні події. Для середовища Kubernetes проблема вибору набору даних є особливо актуальною, оскільки сучасні контейнерні платформи характеризуються високою динамічністю, використанням зашифрованого трафіку, автоматичним масштабуванням сервісів та постійною зміною топології мережі. Більшість існуючих відкритих наборів даних були сформовані для традиційних корпоративних мереж і не відображають специфіку функціонування контейнеризованих застосунків.

З метою вибору найбільш придатного набору даних було проведено аналіз найбільш поширених відкритих датасетів, що використовуються у дослідженнях систем виявлення вторгнень. Порівняльну характеристику цих наборів наведено у табл. 3.1.

Тривалий час еталонними наборами даних для оцінювання алгоритмів машинного навчання залишалися KDD99 та його вдосконалена версія NSL-KDD. Незважаючи на широке використання у наукових дослідженнях, ці набори формувалися для мережевих середовищ кінця 1990-х років та не враховують сучасні особливості функціонування комп'ютерних мереж. У них відсутній зашифрований трафік, не моделюються контейнерні платформи, а мережеві сценарії побудовані на статичній топології. Через це використання KDD99 та NSL-KDD для дослідження Kubernetes є малоефективним, оскільки вони не відображають сучасних моделей взаємодії між сервісами.

Таблиця 3.1 – Порівняльна характеристика наборів даних для навчання IDS у середовищі Kubernetes

Набір даних	Переваги	Обмеження щодо використання у Kubernetes
KDD99 / NSL-KDD	Широко використовуються як еталонні набори даних; доступні для порівняння результатів різних моделей	Застарілі сценарії мережевого трафіку; відсутність контейнерних технологій, шифрування та сучасних протоколів; не враховують особливості Kubernetes
CICIDS2017 / CICIDS2018	Містять сучасні типи атак; велика кількість інформативних ознак; реалістичний мережевий трафік	Статична мережева топологія; відсутність моделювання автоматичного масштабування Pod та Service Mesh; виявлені окремі помилки розмітки
UNSW-NB15	Різноманітні сучасні атаки; більш реалістичний трафік порівняно з KDD99	Орієнтований на корпоративні мережі; не враховує особливості контейнерних середовищ та сервісної взаємодії Kubernetes
BoT-IoT / TON-IoT	Призначені для дослідження безпеки IoT; містять потоки мережевого трафіку та різноманітні IoT-атаки	Орієнтовані на пристрої Інтернету речей; не відображають взаємодію між мікросервісами Kubernetes
CSE-CIC-IDS2018	Багатоденний запис мережевого трафіку; комбіновані сценарії атак; висока різноманітність потоків	Не моделює характерні особливості Kubernetes: автоматичне масштабування Pod, Service Mesh та внутрішній сервісний трафік
Improved CICIDS	Виправлена розмітка; покращена якість мережевих потоків; акцент на Flow-метаданих; сучасні типи атак	Відсутнє моделювання реального Kubernetes-кластера; середовище залишається синтетичним

Значно ширше поширення сьогодні отримали набори CICIDS2017 та CICIDS2018, які містять різноманітні типи сучасних мережевих атак, включаючи DDoS, Brute Force, Port Scanning, Botnet, Web Attack та інші сценарії. Дані набори характеризуються великою кількістю статистичних характеристик мережевих потоків, що робить їх придатними для навчання моделей машинного навчання. Водночас вони також мають низку обмежень. Насамперед мережеве середовище, у якому формувався трафік, є статичним і не враховує особливостей автоматичного масштабування контейнерів, використання Service Mesh та взаємного TLS-шифрування між сервісами. Крім того, дослідниками було виявлено певні помилки у розмітці окремих записів та неточності під час формування ознак.

Окремий клас навчальних наборів представлений BoT-IoT та TON-IoT, які орієнтовані на середовище Інтернету речей. Основний акцент у цих наборах зроблено на взаємодії між периферійними пристроями, сенсорами та IoT-шлюзами. Незважаючи на достатньо високу різноманітність сценаріїв атак, структура мережевого трафіку суттєво відрізняється від взаємодії мікросервісів у Kubernetes. Через це зазначені набори даних доцільніше використовувати під час дослідження безпеки IoT-систем, ніж контейнерних платформ.

Більш сучасним набором є CSE-CIC-IDS2018, який містить багатоденні записи мережевого трафіку та комбіновані сценарії розвитку атак. Порівняно з CICIDS2017 цей набір даних характеризується більшою різноманітністю сценаріїв та покращеною якістю сформованих потоків. Разом із тим він також не моделює специфічні особливості Kubernetes, такі як автоматичне масштабування Pod, короткотривалі життєві цикли контейнерів, внутрішній сервісний трафік та використання Service Mesh.

Останнім часом значну увагу дослідників привертає Improved CICIDS удосконалена версія набору CICIDS, створена для усунення недоліків попередніх редакцій. Основною перевагою цього набору є виправлення помилок розмітки, уточнення часових меж атак та усунення неточностей при побудові мережевих потоків. Додатково було виправлено низку помилок формування ознак, зокрема пов'язаних із підрахунком TCP-прапорців, дублюванням полів та некоректними статистичними характеристиками потоків.

Важливою особливістю удосконаленого набору є також більш реалістичне формування мережевих потоків. На відміну від початкової версії, алгоритми агрегації враховують часові інтервали завершення потоків та мінімізують появу штучних закономірностей, які могли негативно впливати на процес навчання моделей машинного навчання. Це дозволяє уникнути ситуацій, коли алгоритм класифікації фактично запам'ятовує особливості формування датасета, а не характерні ознаки атак.

Ще однією перевагою Improved CICIDS є акцент на використанні метаданих мережевих потоків замість аналізу корисного навантаження пакетів.

Такий підхід повністю відповідає сучасним вимогам контейнеризованих середовищ, де дедалі частіше використовується взаємне TLS-шифрування між сервісами. За таких умов аналіз вмісту пакетів стає неможливим, тоді як статистичні характеристики мережевих потоків залишаються доступними для побудови моделей машинного навчання.

Незважаючи на наявність певних обмежень, повністю адаптованих відкритих наборів даних саме для середовища Kubernetes наразі не існує. Жоден із розглянутих датасетів не моделює повною мірою динамічне створення та видалення контейнерів, автоматичне масштабування Pod, зміну мережевої топології або використання сучасних сервісних сіток. Проте серед доступних відкритих наборів даних саме Improved CICIDS забезпечує найкраще поєднання актуальності мережевого трафіку, різноманітності сучасних атак, високої якості розмітки та придатності для аналізу мережевих потоків.

Саме тому у даній роботі для навчання та оцінювання алгоритму k-Nearest Neighbors було обрано Improved CICIDS. Використання цього набору дозволяє формувати більш надійну модель класифікації мережевого трафіку, зменшити вплив помилок розмітки та підвищити достовірність результатів експериментального дослідження. Обраний набір даних забезпечує оптимальний компроміс між сучасністю сценаріїв атак, якістю сформованих мережевих потоків та можливістю застосування у задачах побудови систем виявлення вторгнень для кластерів Kubernetes.

3.3 Експериментальне дослідження системи виявлення вторгнень

Першим етапом функціонування розробленої системи виявлення вторгнень є захоплення мережевого трафіку контейнерів (Pod) у кластері Kubernetes та формування файлів мережевого трасування у форматі PCAP. Файли формату PCAP (Packet Capture) є стандартним форматом збереження мережевого трафіку та містять заголовок файлу і записи окремих мережевих пакетів. Структура цих файлів визначається відповідною специфікацією

формату [10].

Для перехоплення мережевого трафіку використовується програмний інструмент Ksniff (Kubesniff), який реалізований у вигляді розширення (plugin) до утиліти kubectl. Інструмент забезпечує можливість виконання мережевого моніторингу безпосередньо на рівні Pod у кластері Kubernetes. Для запуску аналізу достатньо вказати ім'я контейнера та простір імен Kubernetes. Якщо аналіз виконується для привілейованого контейнера, використовується додатковий параметр -p. Після запуску Ksniff автоматично передає захоплений трафік до програми Wireshark, з якої формується файл мережевого трасування у форматі PCAP.

Приклад запуску Ksniff: kubectl sniff <PodName> -n <Namespace> або для привілейованого контейнера kubectl sniff -p <PodName>

Отримані файли PCAP надалі перетворюються у мережеві потоки (Network Flows) за допомогою удосконаленої версії програмного комплексу Improved CICFlowMeter [11]. На відміну від традиційного аналізу окремих пакетів, потоковий аналіз дозволяє формувати статистичні характеристики мережевих з'єднань, що використовуються як вхідні дані для алгоритмів машинного навчання.

Результатом роботи CICFlowMeter є файл формату CSV, у якому кожен рядок відповідає одному мережевому потоку та містить 83 статистичні характеристики. Під мережевим потоком розуміється двонаправлений обмін пакетами між двома вузлами мережі, які мають однакову п'ятірку параметрів (Source IP, Destination IP, Source Port, Destination Port, Transport Protocol) протягом визначеного інтервалу часу.

У роботі використовується саме удосконалена версія CICFlowMeter, оскільки дослідження [11] показали наявність низки недоліків у початковій реалізації цього програмного забезпечення, пов'язаних із формуванням мережевих потоків та розрахунком окремих статистичних характеристик. Використання потокового аналізу також дозволяє вирішити проблему аналізу зашифрованого мережевого трафіку, оскільки при використанні Service Mesh

класичний аналіз вмісту пакетів за допомогою Wireshark стає практично неможливим.

Важливою складовою розробленої системи є сигнатурний аналізатор Snort. Це відкрита система виявлення вторгнень, яка широко застосовується як у практичних рішеннях, так і в наукових дослідженнях інформаційної безпеки. Основною перевагою Snort є можливість створення власних правил (Rules), що дозволяє адаптувати систему до особливостей конкретного Kubernetes-кластера.

Файли PCAP, сформовані за допомогою Ksniff, передаються до Snort ще до етапу перетворення у мережеві потоки, оскільки Snort працює безпосередньо з пакетними даними. Архітектура системи включає модуль декодування пакетів (Packet Decoder), систему попередньої обробки (Preprocessor), модуль виявлення вторгнень (Detection Engine) та систему формування повідомлень (Alert System).

Принцип роботи Snort базується на порівнянні мережевих пакетів із набором правил. Кожне правило складається із двох основних частин: заголовка (Rule Header) та параметрів (Rule Options). Заголовок містить тип дії, транспортний протокол, IP-адреси джерела та призначення, порти та напрямок передачі даних. Друга частина визначає додаткові умови формування повідомлення про виявлену атаку, включаючи ключові слова, сигнатури та текст повідомлення.

Для власного Kubernetes-кластера можуть бути сформовані спеціалізовані правила, що реагують на певні типи небажаної мережевої активності. Разом із тим ефективність сигнатурного аналізу безпосередньо залежить від кількості та якості сформованих правил. Snort ефективно виявляє лише відомі типи атак, тоді як нові або модифіковані сценарії вторгнень залишаються поза увагою системи. Крім того, використання Service Mesh із взаємним TLS-шифруванням значно обмежує можливості Snort, оскільки його SSL-препроцесор аналізує лише процес встановлення TLS-з'єднання і не має доступу до зашифрованого вмісту пакетів.

Приклад правила Snort має представлено в лістингу 3.1

Лістинг 3.1 – Приклад правила Snort

```
alert tcp any 80 -> any any (content:"HTTP";
msg:"HTTP packet found";
sid:10000100;
rev:005;)
```

Для навчання моделі машинного навчання використовується удосконалений набір даних Improved CICIDS. Вибір саме цього набору даних обумовлений тим, що початкова версія CICIDS2017 містила низку недоліків, серед яких помилки розмітки, некоректне формування окремих ознак та закритий алгоритм генерації міток класів. Покращена версія використовує відкритий механізм формування розмітки, виправлені статистичні характеристики мережевих потоків та додаткові ознаки, що можуть бути сформовані за допомогою удосконаленого CICFlowMeter. Формування цього набору даних виконувалося шляхом детального аналізу понад 630 ГБ сирого мережевого трафіку.

Набір даних містить 83 ознаки, які характеризують кожний мережевий потік. Для зручності аналізу вони можуть бути згруповані за функціональним призначенням (табл. 3.2).

Перед навчанням алгоритму k-Nearest Neighbors усі дані проходять попередню обробку, що включає очищення набору даних, нормалізацію ознак, кодування категоріальних змінних та, за необхідності, зменшення розмірності простору ознак методом головних компонент (PCA). Після цього формується навчальна та тестова вибірки, які використовуються для побудови та оцінювання моделі машинного навчання. Це забезпечує можливість отримання достовірних результатів класифікації мережевого трафіку та оцінювання ефективності розробленої системи виявлення вторгнень у кластері Kubernetes.

Таблиця 3.2 – Основні групи ознак набору даних Improved CICIDS

Група ознак	Приклади ознак
Ідентифікація потоку	Flow ID, Source IP, Destination IP, Source Port, Destination Port, Protocol, Timestamp
Часові характеристики	Flow Duration, Flow IAT Mean, Flow IAT Std, Fwd IAT Mean, Bwd IAT Mean, Active Mean, Idle Mean
Статистика пакетів	Total Forward Packets, Total Backward Packets, Packet Length Min, Packet Length Max, Packet Length Mean, Packet Length Std
Обсяг переданих даних	Total Length of Forward Packets, Total Length of Backward Packets, Flow Bytes/s, Flow Packets/s
Характеристики TCP	FIN Flag Count, SYN Flag Count, ACK Flag Count, PSH Flag Count, RST Flag Count, URG Flag Count, CWR Flag Count, ECE Flag Count
Параметри сегментів	Average Packet Size, Forward Segment Size Avg, Backward Segment Size Avg
Підпотоки (Subflow)	Subflow Forward Packets, Subflow Forward Bytes, Subflow Backward Packets, Subflow Backward Bytes
Вікна TCP	FWD Init Win Bytes, Bwd Init Win Bytes
Інтенсивність передачі	Fwd Packets/s, Bwd Packets/s, Down/Up Ratio
Мітка класу	Label

Після формування набору ознак було виконано аналіз їх інформативності із застосуванням методу головних компонент (Principal Component Analysis, PCA). Використання PCA дозволяє визначити кількість компонент, достатню для збереження більшої частини інформації, що міститься у вихідному наборі даних, а також зменшити розмірність простору ознак перед навчанням алгоритму k-Nearest Neighbors.

На рис. 3.2 наведено графік накопиченої поясненої дисперсії (Scree Plot), який демонструє залежність між кількістю головних компонент та часткою поясненої дисперсії.

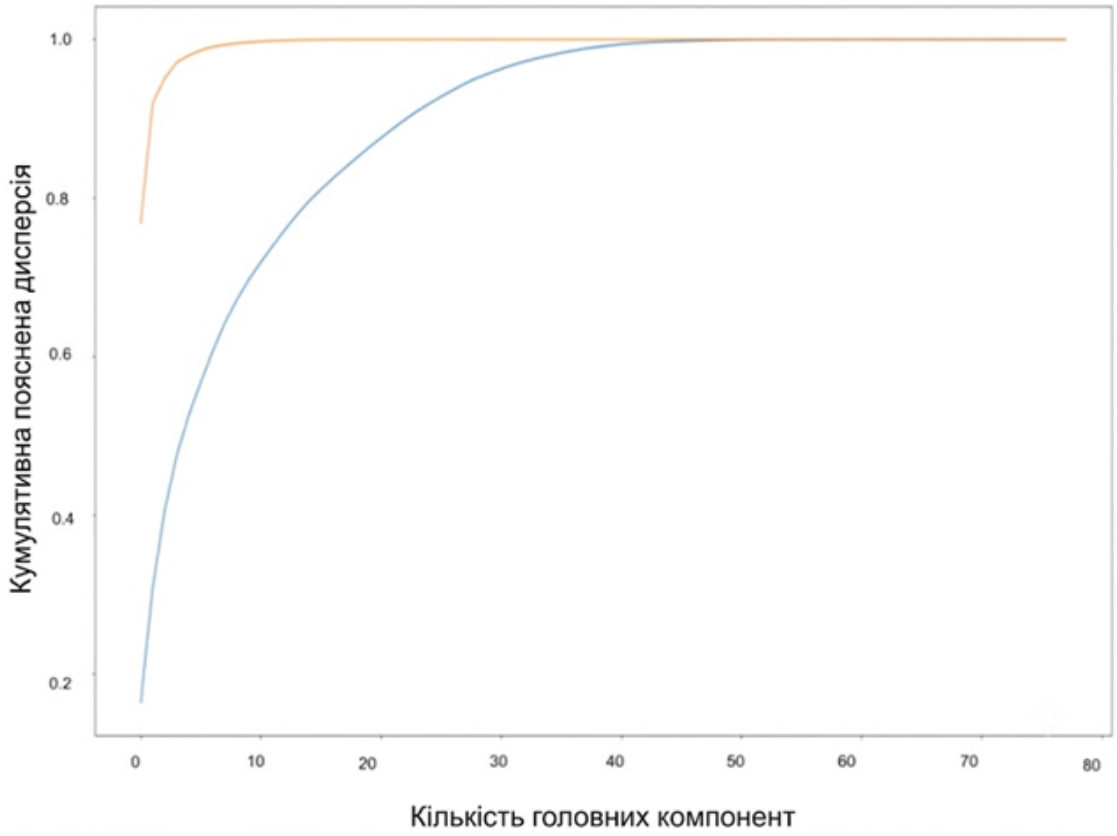


Рисунок 3.2 – Залежність накопиченої поясненої дисперсії від кількості головних компонент

Як видно з рис. 3.2, більша частина інформативності вихідних даних зосереджена у відносно невеликій кількості головних компонент. Найбільший приріст накопиченої поясненої дисперсії спостерігається для перших компонент, після чого швидкість її зростання поступово зменшується. При використанні приблизно 30–35 головних компонент пояснена дисперсія перевищує 95 %, що свідчить про наявність значної кореляції між окремими ознаками та підтверджує доцільність застосування методу головних компонент для зменшення розмірності простору ознак.

Використання PCA є особливо важливим при застосуванні алгоритму k-Nearest Neighbors, оскільки обчислювальна складність цього алгоритму залежить від кількості ознак, що використовуються під час обчислення відстані між об'єктами. Зменшення розмірності дозволяє скоротити час навчання і класифікації, знизити вплив надлишкових та шумових ознак, а також підвищити узагальнювальну здатність моделі.

Для проведення експериментального дослідження використано набір даних Improved CICIDS, записи якого розподілено на п'ять окремих файлів відповідно до дня формування мережевого трафіку. Кожен файл містить різні сценарії легітимної мережевої активності та кібератак, що забезпечує можливість комплексного оцінювання ефективності алгоритму класифікації. Структуру використаного набору даних наведено в табл. 3.3.

Таблиця 3.3 – Структура набору даних Improved CICIDS

Файл	Типи мережевого трафіку
Monday	Лише легітимний (Benign) мережевий трафік
Tuesday	Легітимний трафік, атаки FTP-Patator та SSH-Patator
Wednesday	Легітимний трафік, DoS/DDoS, DoS Slowloris, DoS SlowHTTPTest, DoS Hulk, DoS GoldenEye, Heartbleed
Thursday	Легітимний трафік, Web Attack – Brute Force, XSS, SQL Injection, Infiltration
Friday	Легітимний трафік, атаки Port Scan

Поділ набору даних за окремими днями дозволяє охопити широкий спектр сучасних мережевих атак та забезпечує формування навчальної вибірки з різнорідними сценаріями функціонування мережі. Це сприяє підвищенню узагальнювальної здатності алгоритму k-Nearest Neighbors та забезпечує більш об'єктивне оцінювання ефективності розробленої системи виявлення вторгнень у середовищі Kubernetes.

Після попередньої обробки даних усі записи, що містили пропущені

значення (NULL), були вилучені з набору даних. Це дозволило забезпечити коректність подальшого навчання моделі машинного навчання та виключити вплив неповних записів на результати класифікації.

Оскільки набір даних Improved CICIDS містить велику кількість статистичних характеристик мережевих потоків, для зменшення розмірності простору ознак було застосовано метод головних компонент (Principal Component Analysis, PCA). Перед виконанням PCA всі числові ознаки були стандартизовані за допомогою алгоритму StandardScaler, який виконує масштабування даних до нульового середнього значення та одиничного стандартного відхилення. Така попередня нормалізація є необхідною умовою коректної роботи PCA, оскільки метод головних компонент чутливий до різних масштабів вимірювання ознак.

Результати аналізу поясненої дисперсії наведено на рис. 3.2. Аналіз графіка показує, що вже сім головних компонент пояснюють понад 99 % загальної дисперсії початкового набору даних. Це свідчить про наявність значної кореляції між окремими ознаками та підтверджує доцільність застосування методу PCA для скорочення розмірності даних без істотної втрати інформації.

З метою об'єктивного оцінювання якості роботи алгоритму k-Nearest Neighbors застосовано метод стратифікованої 10-кратної перехресної перевірки (Stratified 10-Fold Cross-Validation). Використання цього підходу дозволяє отримати більш достовірну оцінку узагальнювальної здатності моделі та мінімізувати вплив випадкового розподілу вибірки на результати експерименту.

Під час стратифікованої перехресної перевірки набір даних було розділено на десять приблизно однакових підвибірок (fold), при цьому у кожній із них зберігалося співвідношення між класами, аналогічне вихідному набору даних. На кожній ітерації дев'ять підвибірок використовувалися для навчання моделі, тоді як одна підвибірка застосовувалася для її тестування. Процедура повторювалася десять разів таким чином, щоб кожна підвибірка один раз виступала тестовою. Для кожної ітерації обчислювалися показники Accuracy,

Precision, Recall та F1-score, після чого визначалися їх середні значення.

Використання стратифікації є особливо важливим для набору даних Improved CICIDS, оскільки кількість записів, що відповідають нормальному мережевому трафіку, суттєво перевищує кількість записів із мережевими атаками. Збереження однакового співвідношення класів у кожній підвибірці дозволяє уникнути зміщення результатів оцінювання та забезпечує більш об'єктивне порівняння ефективності моделі. Перед виконанням розподілу всі записи були випадковим чином перемішані, що додатково зменшує ризик виникнення систематичних похибок.

Застосування стратифікованої перехресної перевірки також знижує ймовірність перенавчання моделі, запобігає витоку інформації між навчальною та тестовою вибірками та підвищує достовірність отриманих результатів.

Після завершення етапу попередньої обробки та зменшення розмірності даних було виконано навчання моделі k-Nearest Neighbors із використанням семи головних компонент, отриманих методом PCA.

Під час навчання використовувалася бібліотека Scikit-learn, яка реалізує алгоритм k-Nearest Neighbors та забезпечує широкий набір засобів для налаштування параметрів моделі.

Основні параметри навчання моделі наведено нижче:

- програмна реалізація алгоритму здійснювалася із використанням бібліотеки Scikit-learn;
- кількість найближчих сусідів встановлено рівною k = 5;
- усі сусідні об'єкти мали однакову вагу під час голосування (weights = uniform);
- пошук найближчих сусідів здійснювався із використанням параметра algorithm = auto, який автоматично обирає найбільш ефективний алгоритм пошуку залежно від структури даних. За необхідності бібліотека може використовувати алгоритми Ball Tree, KD Tree або повний перебір (Brute Force);
- як функцію обчислення відстані використано метрику Мінковського

(Minkowski distance);

- оцінювання моделі виконувалося методом стратифікованої 10-кратної перехресної перевірки, що забезпечує однаковий розподіл кожного класу атак у всіх підвибірках та дозволяє отримати більш надійну оцінку ефективності класифікації.

Фрагменти програмного коду, що реалізують процедуру навчання моделі та формування звіту класифікації, наведено на рис. 3.3 та рис. 3.4

```
def train_model_predict(model, model_name, X, y, skf):
    for train_index, test_index in skf.split(X, y): # 1)
        X_train, X_test = X.iloc[train_index], X.iloc[test_index] # 2)
        y_train, y_test = y.iloc[train_index], y.iloc[test_index]

        reshaped_y_train = np.asarray(y_train).reshape(-1, 1) # 3)
        reshaped_y_test = np.asarray(y_test).reshape(-1, 1)

    model.fit(X_train, reshaped_y_train.ravel()) # 4)
    pred_y = model.predict(X_test) # 5)
    score = classification_report(reshaped_y_test, pred_y, digits=4) # 6)
    cm = confusion_matrix(reshaped_y_test, pred_y, labels=model.classes_)
    print('Classification report: \n', score, '\n')
    disp=ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=model.classes_)
    disp.plot()
    plt.show()
    return accuracy_score(reshaped_y_test, pred_y), X_train, X_test, y_train, pred_y
```

Рисунок 3.3 – Фрагмент програмного коду для формування звіту про результати класифікаці

```
#Training
X = df_final.drop(['Label'], axis = 1)
y = df_final['Label']
skf = StratifiedKFold(n_splits=10, shuffle=False)

for train_index, test_index in skf.split(X, y):
    X_train, X_test = X.iloc[train_index], X.iloc[test_index]
    y_train, y_test = y.iloc[train_index], y.iloc[test_index]

    reshaped_y_train = np.asarray(y_train).reshape(-1, 1)
    reshaped_y_test = np.asarray(y_test).reshape(-1, 1)

#Using the ml models
knn_model = KNeighborsClassifier(n_neighbors=5, weights='uniform',algorithm='auto', leaf_size=30,p=2, metric='minkowski',metric_params=None, n_jobs=None)
accuracy, X_train, X_test, y_train, pred_y = train_model_predict(knn_model, "K-Nearest Neighbor", X, y, skf)
```

Рисунок 3.4 – Фрагмент програмного коду навчання моделі k-Nearest Neighbors

Для побудови моделі аномального аналізу всі типи атак були об'єднані в один клас Attack, тоді як легітимний мережевий трафік зберіг мітку Benign. Такий підхід дозволяє перейти від багатокласової класифікації до двокласової задачі, основною метою якої є розмежування нормального та аномального мережевого трафіку.

Використання двокласової класифікації пояснюється тим, що модель аномального аналізу не орієнтована на визначення конкретного типу атаки. Її призначення полягає у виявленні будь-яких відхилень від нормальної поведінки мережевого трафіку, що потенційно можуть відповідати як відомим, так і новим сценаріям вторгнень. Водночас класифікація конкретних типів атак виконується алгоритмом k-Nearest Neighbors, який навчений на повному наборі класів Improved CICIDS.

Для побудови моделі аномального аналізу використано алгоритм Isolation Forest, реалізований засобами бібліотеки Scikit-learn. Під час навчання всі записи, що належать до різних типів атак, розглядалися як аномальні, тоді як записи легітимного мережевого трафіку формували клас нормальної поведінки. Такий підхід забезпечує можливість виявлення нетипових мережевих потоків незалежно від їх належності до конкретного класу атак.

Після навчання моделей було розроблено керуючий Bash-скрипт, який автоматизує всі етапи роботи системи виявлення вторгнень. Скрипт поєднує захоплення мережевого трафіку, сигнатурний аналіз та класифікацію мережевих потоків із використанням алгоритмів машинного навчання, що дозволяє реалізувати повністю автоматизований цикл аналізу мережевої активності у кластері Kubernetes.

Під час виконання скрипта послідовно реалізуються такі операції:

- захоплення мережевого трафіку вибраного контейнера (Pod) за допомогою Ksniff;
- формування файлу мережевого трасування у форматі PCAP;
- аналіз отриманого файлу системою Snort із використанням визначеного набору правил;
- перетворення файлу PCAP у файл мережевих потоків (CSV) засобами Improved CICFlowMeter;
- передавання сформованого набору потоків до моделі Isolation Forest для виявлення аномалій;
- передавання тих самих потоків до моделі k-Nearest Neighbors для

класифікації типів атак;

- збереження результатів роботи моделей машинного навчання у окремих файлах;

- аналіз результатів усіх трьох компонентів системи (Snort, Isolation Forest та k-Nearest Neighbors) і формування повідомлення про можливе вторгнення.

Для налаштування роботи скрипта використовуються параметри командного рядка:

- -n – ім'я контейнера (Pod), мережевий трафік якого необхідно аналізувати;
- -o – ім'я вихідного файлу PCAP;
- -r – файл локальних правил Snort;
- -f – ім'я файлу мережевих потоків (CSV);
- -s – запуск сигнатурного аналізу Snort;
- -u – ім'я файлу результатів моделі Isolation Forest;
- -x – ім'я файлу результатів моделі k-Nearest Neighbors.

Приклад запуску скрипта наведено в лістингу 3.2.

Лістинг 3.2 – Приклад запуску скрипта

```
./IDS.bash -n NameOfThePod \
-o OutputPCAP.pcap \
-r LocalSnort.rules \
-f OutputFlows.csv \
-s \
-u IsolationForestOutput.csv \
-x KNNOutput.csv
```

Обов'язковим параметром є лише ім'я контейнера (-n). Інші параметри визначають назви файлів, що автоматично створюються під час роботи системи.

Після запуску скрипт безперервно здійснює моніторинг мережевого трафіку вибраного контейнера до моменту його примусового завершення користувачем. Захоплений трафік зберігається у форматі PCAP, після чого

послідовно аналізується системою Snort, перетворюється у мережеві потоки та передається до моделей Isolation Forest і k-Nearest Neighbors. На завершальному етапі результати всіх трьох підсистем об'єднуються. Якщо хоча б одна з них виявляє аномальну або потенційно шкідливу активність, система автоматично формує повідомлення про можливе вторгнення, що дозволяє адміністратору оперативно перейти до детального аналізу інциденту.

3.4 Результати тестування системи виявлення вторгнень

Оцінювання ефективності розробленої системи виявлення вторгнень проводилося у два етапи. Перший етап передбачав аналіз якості роботи моделей машинного навчання на наборі даних Improved CICIDS, що дозволило визначити їхню здатність до класифікації мережевого трафіку та виявлення атак. Другий етап полягав у комплексному тестуванні всієї розробленої системи шляхом моделювання мережевого вторгнення в середовищі Kubernetes. На цьому етапі оцінювалася взаємодія всіх компонентів системи, зокрема Snort, k-Nearest Neighbors та Isolation Forest.

Загальна схема проведення експериментального дослідження наведена на рисунку 3.5.

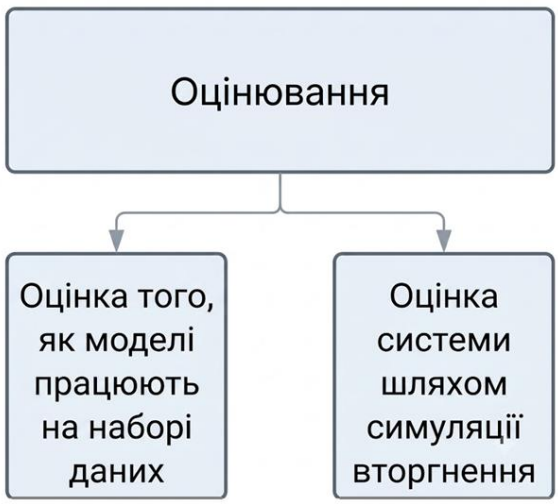


Рисунок 3.5 – Схема двоетапного оцінювання системи виявлення вторгнень

На першому етапі виконувалося оцінювання моделей машинного навчання на тестовій вибірці набору даних Improved CICIDS. Для моделі k-Nearest Neighbors аналізувалися результати багатокласової класифікації мережевого трафіку, тоді як для моделі Isolation Forest оцінювалася здатність виявляти аномальні мережеві потоки незалежно від конкретного типу атаки.

Другий етап експериментального дослідження був спрямований на перевірку працездатності всієї системи виявлення вторгнень. Для цього здійснювалося моделювання мережевої атаки у кластері Kubernetes із подальшим захопленням мережевого трафіку, його аналізом за допомогою сигнатурного механізму Snort, перетворенням пакетів у мережеві потоки засобами Improved CICFlowMeter та їхньою обробкою алгоритмами k-Nearest Neighbors і Isolation Forest. Після завершення аналізу результати всіх компонентів інтегрувалися для формування остаточного рішення щодо наявності або відсутності вторгнення.

Оцінювання якості роботи моделей здійснювалося за загальноприйнятими метриками класифікації: Accuracy, Precision, Recall та F1-score. Дані показники дозволяють комплексно оцінити ефективність роботи системи виявлення вторгнень, враховуючи як загальну точність класифікації, так і здатність моделі виявляти атаки без значного збільшення кількості хибних спрацювань.

Для оцінювання результатів використовувалися такі метрики:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{3.1}$$

$$Precision = \frac{TP}{TP + FP} \tag{3.2}$$

$$Recall = \frac{TP}{TP + FN} \tag{3.3}$$

$$F1 = \frac{2 \cdot Precision \cdot Recall}{Precision + Recall} \quad (3.4)$$

де: TP (True Positive) – кількість атак, правильно класифікованих як атаки, TN (True Negative) – кількість легітимних з'єднань, правильно визначених як нормальний трафік, FP (False Positive) – кількість нормальних з'єднань, помилково класифікованих як атаки, FN (False Negative) – кількість атак, які система не змогла виявити.

Матрицю помилок, а також основні показники оцінювання якості моделі, зокрема точність (Precision), повноту (Recall), F1-міру (F1-score) та підтримку (Support), наведено на рисунках 3.6 та 3.7.

Рядки матриці помилок відповідають таким класам мережевого трафіку та типам атак:

1. Benign (легітимний трафік)
2. Bot – Attempted (спроба бот-атаки)
3. Bot
4. Port Scan
5. DDoS
6. FTP-Patator
7. SSH-Patator
8. FTP-Patator – Attempted
9. SSH-Patator – Attempted
10. DoS Slowloris
11. DoS Slowloris – Attempted
12. DoS Slowhttptest
13. DoS Slowhttptest – Attempted
14. DoS Hulk
15. DoS Hulk – Attempted
16. DoS GoldenEye
17. Heartbleed

18. DoS GoldenEye – Attempted
19. Web Attack – Brute Force
20. Web Attack – Brute Force – Attempted
21. Infiltration – Attempted
22. Infiltration
23. Web Attack – XSS – Attempted
24. Web Attack – XSS
25. Web Attack – SQL Injection.

Як видно з результатів, наведених на рисунку 3.6, модель демонструє високий рівень загальної точності класифікації. Водночас необхідно враховувати, що набір даних характеризується значним дисбалансом класів, оскільки більшість мережевих потоків належать до легітимного трафіку. Саме тому значення загальної точності (Accuracy) не може повною мірою характеризувати ефективність моделі й повинно аналізуватися разом із показниками Precision, Recall та F1-score.

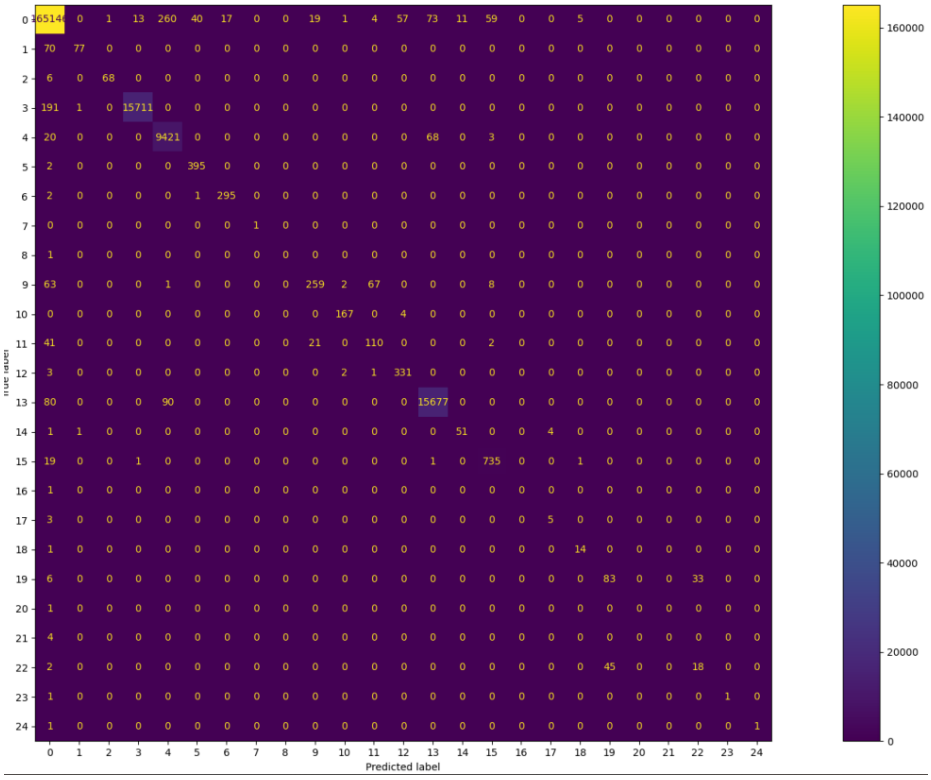


Рисунок 3.6 – Матриця помилок моделі k-Nearest Neighbors

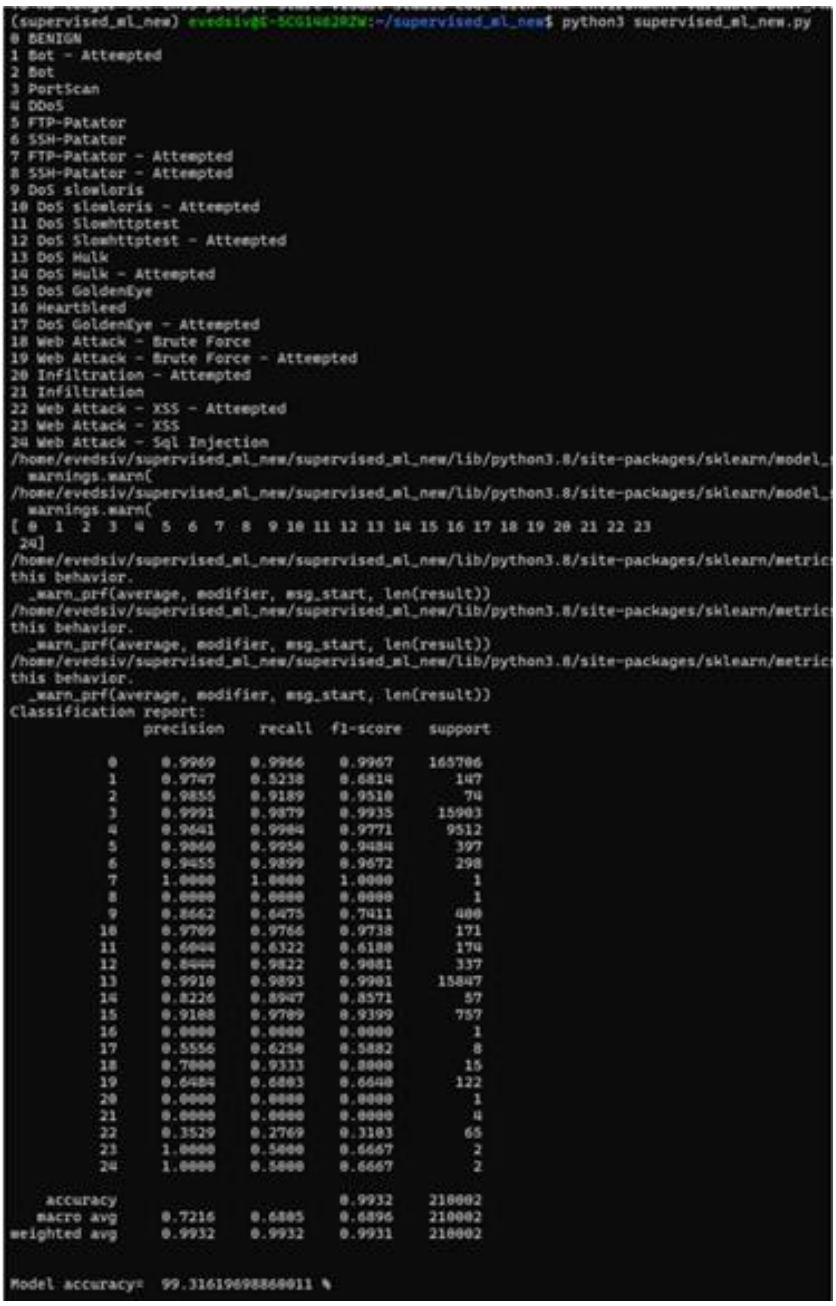


Рисунок 3.7 – Результати оцінювання моделі k-Nearest Neighbors на наборі даних Improved CICIDS

На рисунку 3.8 також наведено результати класифікації для кожного окремого типу атак. Аналіз показує, що для класів, представлених невеликою кількістю прикладів (низьке значення показника Support), якість класифікації істотно знижується. Це пояснюється недостатньою кількістю навчальних прикладів, що не дозволяє алгоритму k-Nearest Neighbors сформувати достатньо репрезентативну модель простору ознак для таких атак.

Для класів, кількість прикладів яких перевищує 5000, зокрема PortScan, DDoS та DoS Hulk, значення показників Precision, Recall та F1-score перевищують 90 %. Отримані результати свідчать, що використання характеристик мережевих потоків (Flow Features) є достатньо ефективним для виявлення поширених типів мережевих атак за умови наявності достатнього обсягу навчальних даних. Крім того, результати дозволяють зробити висновок про високу ефективність запропонованої моделі під час виявлення атак типу відмова в обслуговуванні (DoS) та розподілена відмова в обслуговуванні (DDoS).

Матрицю помилок, отриману під час тестування моделі Isolation Forest, наведено на рисунку 3.8.

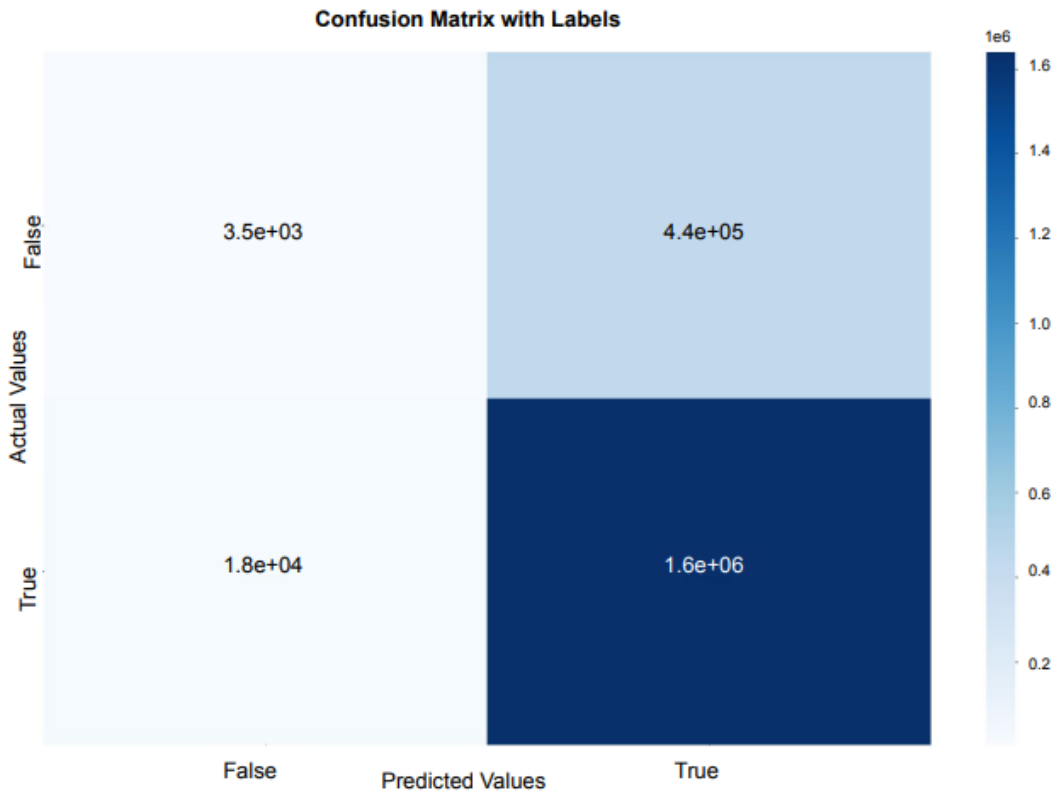


Рисунок 3.8 – Матриця помилок моделі Isolation Forest

Оскільки перед навчанням моделі всі типи атак були об'єднані в один клас «Аномалія», кількість аномальних мережевих потоків у вибірці значно

перевищує кількість легітимного трафіку. Це зумовлено попередньою реконструкцією набору даних, у якій усі класи атак були позначені як аномальні, а нормальний мережевий трафік залишився окремим класом.

Аналіз матриці помилок (рис. 3.8) показує, що найбільшу кількість становлять істинно позитивні спрацювання (True Positive), тобто випадки, коли модель правильно класифікувала аномальний мережевий потік як атаку. Кількість правильно виявлених атак майже на порядок перевищує кількість хибнопозитивних спрацювань (False Positive), коли легітимний трафік був помилково віднесений до категорії аномалій.

Разом із тим результати свідчать про наявність певної кількості хибнопозитивних спрацювань, що є характерною особливістю алгоритмів аномального аналізу. Найбільш критичним показником залишаються хибнонегативні результати (False Negative), коли мережеві атаки помилково класифікуються як нормальний трафік. У проведеному експерименті модель не виявила близько 18 тис. атакувальних потоків, тоді як приблизно 3,5 тис. легітимних мережевих потоків були правильно класифіковані як нормальні.

Для кількісного оцінювання ефективності моделі використовувалися стандартні метрики класифікації.

Показник загальної точності (Accuracy) визначається за формулою 3.1.

За результатами експерименту значення Accuracy становило 77,7 %. Незважаючи на відносно високий показник, його не можна розглядати як основний критерій оцінювання системи виявлення вторгнень, оскільки через дисбаланс класів він не відображає реальної здатності моделі виявляти мережеві атаки. Тому аналіз ефективності доцільно проводити також за показниками Precision, Recall та F1-score.

Показник точності позитивних прогнозів (Precision) обчислюється за формулою 3.2

У проведеному дослідженні його значення становило 78,4 %. Відносно невисоке значення пояснюється тим, що модель досить часто класифікує легітимний мережевий трафік як аномальний, що призводить до збільшення

кількості хибнопозитивних спрацювань.

Повнота виявлення атак (Recall) визначається формулою 3.3.

За результатами тестування значення Recall склало 98,8 %, що свідчить про високу здатність моделі виявляти майже всі мережеві атаки. Для систем виявлення вторгнень саме цей показник є одним із найважливіших, оскільки пропуск атаки може призвести до значно серйозніших наслідків, ніж виникнення додаткових хибнопозитивних спрацювань.

Для комплексного оцінювання роботи моделі використовується F1-міра, яка визначається як гармонійне середнє між показниками Precision та Recall.

У ході експериментального дослідження значення F1-score становило 0,87. Отриманий результат свідчить про достатньо хорошу збалансованість моделі між здатністю виявляти атаки та кількістю помилкових спрацювань. Зниження значення F1-міри порівняно з Recall пояснюється відносно невисоким показником Precision.

Підвищення ефективності моделі можливе шляхом удосконалення процедури підготовки навчального набору даних. Зокрема, перспективним напрямом є формування окремих моделей для різних типів атак або використання багатокласової класифікації замість об'єднання всіх атак у єдиний клас аномалій. Це дозволить зменшити кількість хибнопозитивних спрацювань і покращити загальний баланс між Precision та Recall.

Узагальнені результати оцінювання моделі Isolation Forest наведено у таблиці 3.4.

Для оцінювання працездатності розробленої системи було проведено експеримент із моделюванням мережевої атаки типу DoS у кластері Kubernetes. Під час експерименту мережевий трафік захоплювався за допомогою Ksniff, після чого проходив послідовну обробку системою Snort, програмним комплексом Improved CICFlowMeter та моделями машинного навчання k-Nearest Neighbors і Isolation Forest.

Таблиця 3.4 – Результати оцінювання моделі Isolation Forest на наборі даних Improved CICIDS

Показник	Значення
Істинно негативні (True Negative, TN)	3 500
Хибнопозитивні (False Positive, FP)	440 000
Хибнонегативні (False Negative, FN)	18 000
Істинно позитивні (True Positive, TP)	1 600 000
Accuracy (точність класифікації)	77%
Precision (точність позитивних прогнозів)	78%
Recall (повнота виявлення)	98%

Результати експерименту показали, що обидві моделі успішно виявили мережеве вторгнення. Алгоритм k-Nearest Neighbors правильно класифікував 96 % атакувальних потоків. При цьому переважна більшість атак була віднесена до класу DoS Hulk, тоді як незначна частина була класифікована як PortScan, що пояснюється схожістю окремих статистичних характеристик мережевих потоків.

Алгоритм Isolation Forest продемонстрував високу здатність до виявлення аномальної мережевої активності та правильно визначив 98 % атакуючиих потоків як аномальні. Завдяки використанню неконтрольованого навчання модель змогла виявити також нетипові відхилення від нормальної поведінки мережевого трафіку.

Об'єднання результатів Snort, k-Nearest Neighbors та Isolation Forest дозволило підвищити загальну ефективність системи виявлення вторгнень. Сигнатурний аналіз забезпечував виявлення відомих атак, алгоритм k-Nearest Neighbors здійснював їх точну класифікацію, а Isolation Forest дозволяв виявляти потенційно нові або невідомі сценарії атак. Такий підхід забезпечив більш надійне прийняття рішення щодо наявності мережевого вторгнення та зменшив кількість пропущених атак.

Результати тестування запропонованої системи наведено в табл. 3.5.

Таблиця 3.5 – Результати оцінювання запропонованої системи виявлення вторгнень

Показник	Значення
Істинно негативні (True Negative)	3 612
Хибнопозитивні (False Positive)	87
Хибнонегативні (False Negative)	15
Істинно позитивні (True Positive)	434
Accuracy	96,40%
Precision	83,30%
Recall	96,70%

Як видно з таблиці 3.5, запропонована система забезпечує високу повноту виявлення мережевих атак (Recall = 96,7 %) при одночасному підтриманні достатньо високої точності класифікації (Precision = 83,3 %). Значення F1-score = 0,90 свідчить про збалансованість роботи системи та підтверджує ефективність використання гібридної архітектури.

Отримані результати демонструють, що поєднання сигнатурного аналізу (Snort) із алгоритмами k-Nearest Neighbors та Isolation Forest дозволяє компенсувати недоліки окремих підходів. Запропонована система ефективно виявляє як відомі типи атак, так і аномальну мережеву активність, що робить її придатною для використання в системах моніторингу безпеки кластерів Kubernetes.

ВИСНОВКИ

У кваліфікаційній роботі розроблено систему виявлення вторгнень у кластері Kubernetes, що поєднує сигнатурний аналіз мережевого трафіку із методами машинного навчання. Проведено аналіз архітектури Kubernetes, сучасних систем виявлення вторгнень, особливостей потокового аналізу мережевого трафіку та алгоритмів машинного навчання, що застосовуються для забезпечення інформаційної безпеки контейнеризованих середовищ.

Розроблено архітектуру системи, яка забезпечує автоматизоване захоплення мережевого трафіку, його аналіз за допомогою Snort, формування мережевих потоків засобами Improved CICFlowMeter та їх класифікацію алгоритмами k-Nearest Neighbors і Isolation Forest. Для автоматизації роботи системи створено програмний сценарій, який інтегрує всі етапи обробки мережевого трафіку.

За результатами експериментального дослідження підтверджено працездатність запропонованого рішення. Алгоритм k-Nearest Neighbors забезпечив високу точність класифікації відомих типів атак, тоді як Isolation Forest продемонстрував ефективність під час виявлення аномальної мережевої активності. Поєднання сигнатурного аналізу та алгоритмів машинного навчання дозволило підвищити ефективність виявлення вторгнень у порівнянні з використанням окремих методів.

Практична цінність роботи полягає у можливості використання розробленої системи для безперервного моніторингу мережевого трафіку та підвищення рівня інформаційної безпеки кластерів Kubernetes. Перспективою подальших досліджень є використання ансамблевих і глибоких моделей машинного навчання, а також розширення навчальної вибірки реальними сценаріями мережевих атак.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Gopalan, Vinodkrishna. "Adaptive Query Optimization in Kubernetes Orchestrated Open Source Relational Databases Using Telemetry Driven Feedback Loops." International Journal of Technology, Management and Humanities 12.01 (2026): 77-91.

2. Pods. Kubernetes Documentation. 2024. URL: https://kubernetes.io/docs/concepts/workloads/pods/ (accessed: 30.06.2026).

3. Service // Kubernetes Documentation. 2024. URL: https://kubernetes.io/docs/concepts/services-networking/service/ (accessed: 30.06.2026).

4. L. H. Phuc, L. Phan, and T. Kim, “Traffic-aware horizontal pod autoscaler in kubernetes-based edge computing infrastructure,” IEEE Access, vol. 10, pp. 18 966–18 977, 2022. [Online]. Available: https://ieeexplore.ieee.org/document/9709810

5. Services, Load Balancing, and Networking // Kubernetes Documentation. 2024. URL: https://kubernetes.io/docs/concepts/services-networking/ (accessed: 30.06.2026).

6. Autoscaling Workloads // Kubernetes Documentation. 2024. URL: https://kubernetes.io/docs/concepts/workloads/autoscaling/ (дата звернення: 30.06.2026).

7. O. Jarkas, R. Ko, N. Dong, and R. Mahmud, “A container security survey: Exploits, attacks, and defenses,” ACM Comput. Surv., vol. 57, no. 7, Feb. 2025. [Online]. Available: https://doi.org/10.1145/3715001

8. D. Inc., “Docker security best practices,” 2018, accessed: 2025-04-30. [Online]. Available: https://docs.docker.com/engine/security/security/

9. K. Scarfone and P. Mell, “Guide to intrusion detection and prevention systems (idps),” NIST Special Publication 800-94, 2007, accessed: 2025- 04-30. [Online]. Available: https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-94.pdf.

10. Harris G. Pcap Formats. Internet-Draft. Internet Engineering Task Force (IETF). URL: https://www.ietf.org/archive/id/draft-gharris-opsawg-pcap-01.html (дата звернення: 30.04.2025).

11. M. Müller, D. Behnke, P.-B. Bök, S. Schneider, M. Peuster, and H. Karl, “Cloud-native threat detection and containment for smart manufacturing,” in 2020 6th IEEE Conference on Network Softwarization (NetSoft), 2020, pp. 347–349. [Online]. Available: https://ieeexplore.ieee.org/document/9165321.