ЦИФРОВА ЕКОНОМІКА ТА
ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ
Digital Economy and Information Technologies

УДК 004.75:004.8

СИСТЕМА МОНІТОРИНГУ МЕРЕЖЕВОЇ АКТИВНОСТІ З ВИКОРИСТАННЯМ ЕЛЕМЕНТІВ ШТУЧНОГО ІНТЕЛЕКТУ

Курашвілі Дімітрі
Сергій Сабанов,
Кафедра інформаційних технологій
Запорізький інститут економіки та інформаційних технологій
Запоріжжя, Україна

Анотація – У роботі розглядаються питання, пов’язані з вирішенням комплексної науково-практичної задачі розробки інтелектуальної системи моніторингу мережевого трафіку з метою виявлення вторгнень із використанням методів машинного навчання. Робота охоплює аналіз сучасних підходів до мережевої безпеки, проєктування та програмну реалізацію ключових модулів системи, а також експериментальне дослідження ефективності запропонованого рішення. Аналіз стану систем мережевого моніторингу та виявлення вторгнень, зокрема сигнатурних систем (Snort, Suricata) та систем аналізу потоків (NetFlow, sFlow) показав, що класичні підходи характеризуються обмеженою здатністю до виявлення нових та модифікованих атак, тоді як інтелектуальні методи забезпечують вищу адаптивність, але потребують коректного вибору даних і моделей. В зв’язку з цим було запропоновано систему, що використовує сучасні датасети з реалістичними сценаріями атак. Система створена на основі технологічного стеку, реалізованого мовою Python з використанням бібліотек Scapy, Pandas, NumPy та Scikit-learn. Реалізовано механізм перехоплення трафіку, інженерію ознак, нормалізацію даних та класифікацію потоків із використанням алгоритму Random Forest. На відміну від сигнатурних систем, запропоноване рішення не потребує постійного ручного оновлення правил і здатне узагальнювати поведінкові ознаки атак.

Ключові слова - *IDS/IPS, NMS, SIEM, машинне навчання, мережева атака, моніторинг мережевого трафіку, сигнатурний аналіз, штучний інтелект*

I.ВСТУП

Стрімка цифрова трансформація та зростання залежності від мережевих технологій логічно призвели до пропорційного зростання кількості та складності кіберзагроз. Ландшафт кіберзагроз не є статичним: він еволюціонує, адаптуючись до нових технологій та методів захисту. Зловмисники постійно вдосконалюють свої інструменти та тактики, переходячи від простих масових атак до складних, цілеспрямованих кампаній, які часто фінансуються на державному рівні або діють як організовані злочинні синдикати [1, 2].

Традиційні системи виявлення вторгнень (IDS/IPS), що переважно покладаються на сигнатурний аналіз, нездатні ефективно протидіяти новітнім загрозам, демонструючи ефективність лише проти відомих атак, для яких вже існують сигнатури, але виявляються безсилими, наприклад, перед атаками "нульового дня" (zero-day) та

складними, багатоетапними вторгненнями.

Класична модель безпеки, відома як "замок і рів" (castle-and-moat), де чіткий, захищений периметр (наприклад, корпоративний брандмауер) відділяв "довірену" внутрішню мережу від "недовіреного" зовнішнього світу, практично перестала існувати [3]. Сучасні виклики зумовлені трьома ключовими факторами:

1) Хмарна та гібридна інфраструктури. Організації масово переносять свої дані та додатки у публічні, приватні та гібридні хмари (AWS, Azure, Google Cloud). Це створює складну, розподілену інфраструктуру, де дані постійно переміщуються між локальними центрами обробки даних та хмарними сервісами, що ускладнює моніторинг та контроль [4].

2) Інтернет речей (IoT) та OT-мережі. Мільярди пристроїв IoT – від смарт-сенсорів та камер до медичного обладнання та промислових контролерів (Operational Technology, OT) підключаються до мереж. Ці пристрої часто мають обмежені ресурси для реалізації належних функцій безпеки, поставляються зі слабкими паролями за замовчуванням та нерегулярно отримують оновлення. Це перетворює їх на легкі цілі для створення бот-нетів або на точки входу у корпоративну мережу [5].

3) Віддалена робота. Гібридні моделі роботи означають, що співробітники підключаються до корпоративних ресурсів з особистих пристроїв та незахищених домашніх мереж, фактично розширюючи периметр безпеки до кожного окремого домогосподарства [3].

Ця нова реальність вимагає переходу від застарілої концепції захисту периметра до нових парадигм, таких як "Нульова довіра" (Zero Trust) та "Безперервне управління вразливостями" (Continuous Threat Exposure Management, CTEM), які розглядають кожне з'єднання та кожен пристрій як потенційно небезпечні [4, 6].

Таким чином, сучасне мережеве середовище, в якому традиційні, статичні методи захисту на кшалт сигнатурного аналізу стають неефективними, потребує розробку інтелектуальних систем моніторингу, здатних аналізувати поведінку мережі та виявляти аномалії мережевого трафіку. Іншими словами, найбільш перспективним можна вважати

застосування елементів штучного інтелекту (ШІ) та машинного навчання (МН). Моделі ШІ здатні аналізувати складні нелінійні патерни поведінки, виявляти аномалії в трафіку в реальному часі та вчитися на нових даних. Це дозволяє перейти від реактивного підходу до проактивного захисту, ідентифікуючи навіть раніше невідомі загрози до того, як вони завдадуть значної шкоди.

І. МОДЕЛІ ТА МЕТОДИ ДОСЛІДЖЕННЯ

Комплексний аналіз існуючих підходів, інструментів та технологій, що застосовуються для побудови сучасних систем виявлення вторгнень і моніторингу мережевої безпеки з використанням методів штучного інтелекту, зокрема комерційних та відкритих рішень, на кшалт сигнатурних систем IDS/IPS (Snort, Suricata), систем аналізу потоків (NetFlow, sFlow) та сучасних рішень з елементами штучного інтелекту (Darktrace, Vectra AI) показав, що сигнатурні системи забезпечують високу точність виявлення відомих загроз, проте є малоефективними проти нових або модифікованих атак. Системи на основі аналізу потоків демонструють кращу масштабованість та здатність працювати із зашифрованим трафіком, однак мають обмежені можливості глибокого аналізу. Рішення з використанням ШІ здатні виявляти аномалії та складні атаки, але часто є комерційними, закритими та складними у налаштуванні. Це обґрунтовує доцільність розробки власної гібридної системи з використанням відкритих інструментів і алгоритмів машинного навчання.

Оскільки вибір набору даних для навчання та тестування системи виявлення вторгнень є критично важливим етапом, в ході роботи було проаналізовано декілька найбільш відомих наборів даних для дослідження систем виявлення вторгнень: KDD Cup 1999 (KDD-99), NSL-KDD, CIC-IDS2017/2018 та UNSW-NB15 [7, 8, 9, 10]. Для коректного вибору dataset було враховано низку факторів, зокрема типи представлених атак, відповідність сучасним мережевим протоколам, наявність інформативних ознак, баланс класів та складність попередньої обробки даних. Результати порівняльного аналізу наведено в таблиці 1.

Таблиця 1 – Порівняльна характеристика наборів даних для систем виявлення вторгнень

Критерій порівняння	KDD-99	NSL-KDD	CIC-IDS2017 / 2018	UNSW-NB15
Рік створення	1999	2009	2017 / 2018	2015
Тип трафіку	Симульований	Симульований	Реалістичний	Змішаний
Актуальність атак	Низька	Низька	Висока	Висока
Кількість типів атак	Обмежена	Обмежена	Широка	Широка
Робота із зашифрованим трафіком	Ні	Ні	Частково	Частково
Кількість ознак	41	41	70+	49
Баланс класів	Низький	Покращений	Середній	Покращений
Наявність дублікатів	Висока	Низька	Низька	Низька
Складність обробки	Низька	Низька	Висока	Середня
Придатність для ML/ШІ	Обмежена	Обмежена	Висока	Висока
Практична застосовність	Низька	Низька	Висока	Висока

Очевидно, що класичні набори даних KDD-99 та NSL-KDD, незважаючи на їхню простоту та історичну значущість, не відповідають сучасним вимогам до систем виявлення вторгнень. Вони містять застарілі сценарії атак, не враховують використання шифрування та сучасних мережевих протоколів, а також мають обмежену здатність відображати реальні умови експлуатації корпоративних мереж.

Натомість сучасні набори даних CIC-IDS2017/2018 та UNSW-NB15 характеризуються значно вищим рівнем релевантності. Вони включають актуальні типи атак, реалістичний мережевий трафік та широкий набір ознак, що дозволяє ефективно застосовувати методи машинного навчання та інтелектуального аналізу даних. Особливо важливою є можливість використання цих наборів для дослідження поведінкових моделей і виявлення аномалій у зашифрованих або динамічних мережевих середовищах.

З огляду на мету та завдання даної роботи, у межах практичної реалізації доцільним є використання набору даних типу CIC-IDS2017 (або UNSW-NB15). Такий вибір дозволяє отримати більш об’єктивні результати та створює надійну основу для розробки й оцінювання ефективності системи виявлення вторгнень.

Реалізація системи виявлення вторгнень із використанням методів штучного інтелекту потребує застосування спеціалізованих програмних інструментів для збору та обробки мережевого трафіку, аналізу даних, навчання моделей машинного та глибокого

навчання. Для збору та первинного аналізу мережевого трафіку були використані бібліотеки Scapy та tshark, які забезпечують гнучкий і точний контроль над мережевими даними. Високорівнева бібліотека Scapy для мови програмування Python надає розширені можливості для створення, надсилання, перехоплення та аналізу мережевих пакетів у програмному режимі, а також засоби програмної побудови власних сценаріїв аналізу трафіку, що дозволяє адаптувати процес збору даних до конкретних вимог дослідження [11]. tshark забезпечує можливість пакетного збору трафіку, експорту даних у різних форматах (CSV, JSON, PCAP) та подальшої обробки отриманої інформації за допомогою алгоритмів машинного навчання. Завдяки цьому tshark часто застосовується у складі експериментальних платформ для навчання та тестування інтелектуальних систем виявлення вторгнень [12].

Для попередньої обробки, аналізу та підготовки даних в роботі використано бібліотеки Pandas і NumPy, а також алгоритми машинного навчання з пакету Scikit-learn. Бібліотека NumPy є фундаментальним компонентом екосистеми наукових обчислень у Python і забезпечує ефективну роботу з багатовимірними масивами та матрицями. Вона надає оптимізовані реалізації числових операцій, що використовуються при обчисленні статистичних характеристик мережевого трафіку, нормалізації даних та підготовці ознак для навчання моделей [13]. Pandas є бібліотекою для аналізу та обробки табличних даних, яка широко використовується для роботи з мережевими наборами даних. Вона надає структури даних типу DataFrame, що дозволяють зручно зчитувати, очищувати, фільтрувати та агрегувати інформацію. За допомогою Pandas реалізуються операції попередньої обробки даних, зокрема усунення пропущених значень, кодування категоріальних ознак, масштабування числових параметрів та формування навчальних і тестових вибірок. Бібліотека Scikit-learn є ключовим інструментом для реалізації алгоритмів машинного навчання. Містить широкий набір методів для задач класифікації, кластеризації, регресії та зменшення розмірності. У контексті систем виявлення вторгнень Scikit-learn використовується для побудови моделей на основі таких алгоритмів, як Support Vector Machines, Random Forest, k-Nearest Neighbors, Naive Bayes та методи кластеризації типу k-means [14]. Бібліотека добре інтегрується з NumPy та Pandas, що спрощує побудову повного конвеєра обробки даних. Водночас можливості Scikit-learn є обмеженими при роботі з великими обсягами неструктурованих даних або складними нелінійними залежностями, що характерно для сучасних мережевих середовищ [15].

Таким чином, бібліотеки NumPy, Pandas та Scikit-learn забезпечують надійний інструментарій для реалізації класичних методів машинного навчання в системах виявлення вторгнень. Вони доцільні для попереднього аналізу даних, побудови базових моделей і порівняльних експериментів, а також можуть виступати основою для подальшого застосування методів глибокого навчання.

Для реалізації складних моделей аналізу трафіку та виявлення аномалій обрано фреймворки глибокого навчання TensorFlow, Keras та PyTorch, які дозволяють будувати та навчати нейронні мережі з високою точністю й масштабованістю і відіграють ключову роль у побудові сучасних інтелектуальних систем виявлення вторгнень. TensorFlow є потужним фреймворком глибокого навчання від компанії Google, який підтримує створення, навчання та розгортання нейронних мереж різної складності. Він забезпечує масштабованість обчислень, підтримку апаратного прискорення за допомогою GPU та TPU, а також інтеграцію з хмарними платформами. У контексті систем виявлення вторгнень TensorFlow використовується для побудови глибоких нейронних мереж, зокрема згорткових і рекурентних моделей, що застосовуються для аналізу послідовностей мережевого трафіку та виявлення аномалій [16]. Keras є високорівневим API для створення нейронних мереж, працює поверх TensorFlow та значно спрощує

процес розробки моделей. Завдяки інтуїтивно зрозумілому синтаксису Keras дозволяє швидко реалізовувати та тестувати різні архітектури нейронних мереж, що є особливо корисним на етапі експериментальних досліджень. У системах виявлення вторгнень Keras часто застосовується для реалізації моделей класифікації мережевого трафіку та автоенкодерів для виявлення аномалій [17]. PyTorch є фреймворком глибокого навчання, орієнтованим на дослідницькі задачі та експериментування. Його ключовою особливістю є динамічна модель обчислювального графа, яка забезпечує гнучкість у побудові та налагодженні складних нейронних архітектур. PyTorch широко використовується у наукових дослідженнях у галузі кібербезпеки, зокрема для розробки нестандартних моделей глибокого навчання та дослідження поведінкових характеристик мережевого трафіку.

III. РЕЗУЛЬТАТИ ДОСЛІДЖЕННЯ

Центральним компонентом розроблюваної системи моніторингу мережевої безпеки є інтелектуальний модуль аналізу (ядро штучного інтелекту). Саме цей модуль відповідає за інтерпретацію підготовлених мережевих даних, виявлення аномалій та класифікацію трафіку з метою визначення потенційних атак. Для реалізації інтелектуального модуля аналізу в межах даної роботи обрано алгоритм Random Forest, який належить до класу ансамблевих методів машинного навчання та базується на побудові множини дерев рішень з подальшим агрегуванням їх результатів. Вибір Random Forest зумовлений специфікою задачі аналізу мережевого трафіку, яка характеризується наявністю великої кількості різнорідних ознак, нелінійними залежностями між ними та можливими шумами у даних. У таких умовах використання одиничних моделей (наприклад, одного дерева рішень або лінійних класифікаторів) може призводити до перенавчання або недостатньої узагальнювальної здатності.

Основні компоненти системи на етапі використання інтелектуального модуля аналізу: модуль збору трафіку, модуль інженерії ознак, модуль нормалізації даних, ядро ШІ (Random Forest) та модуль прийняття рішень. Діаграма взаємодії модулів представлена на рис. 1.

На першому етапі модуль збору мережевого трафіку перехоплює пакети або потоки даних та передає їх у сирому вигляді до модуля інженерії ознак. Даний етап відбувається без залучення інтелектуального аналізу та спрямований на накопичення вхідних даних. Далі модуль інженерії ознак виконує агрегацію та обчислення релевантних характеристик трафіку, таких як тривалість з’єднання, кількість пакетів, обсяг переданих байтів, напрямок потоку та значення прапорів транспортних протоколів. Сформований вектор ознак передається до модуля нормалізації.

На наступному кроці модуль нормалізації та кодування приводить отримані ознаки до формату, сумісного з вимогами моделі машинного навчання. Числові значення масштабуються, а категоріальні параметри кодуються у числовий вигляд. Після цього підготовлений набір даних надсилається до інтелектуального ядра системи.

Ядро ШІ, реалізоване на основі алгоритму Random Forest, отримує вектор ознак та виконує процедуру класифікації. Кожне дерево рішень у складі ансамблю формує власний прогноз щодо класу трафіку, після чого результати агрегуються шляхом голосування. На виході формується фінальне рішення про належність трафіку до нормального або аномального (шкідливого) класу.

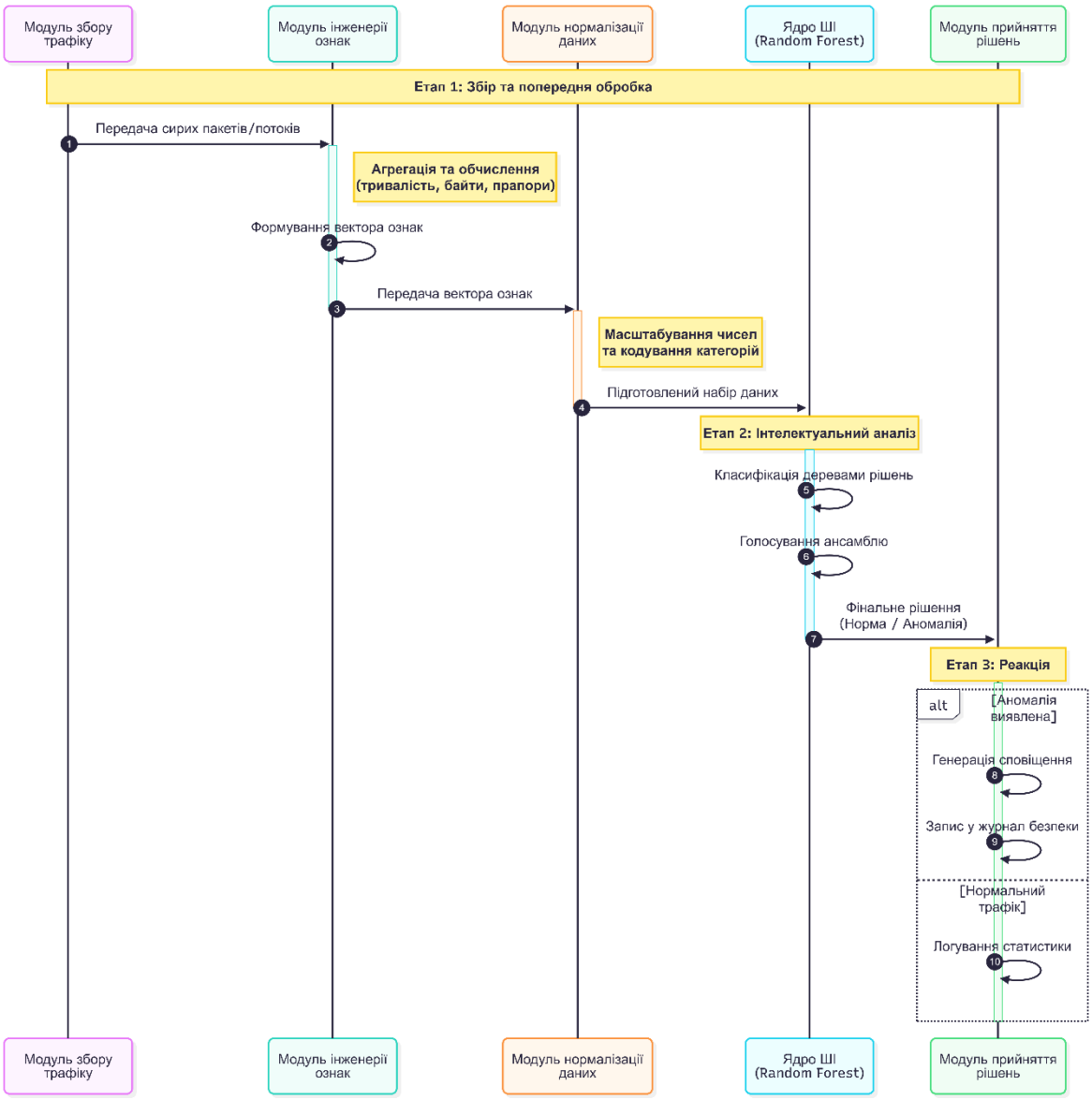

Рисунок 1 – Діаграма послідовності вибору та застосування моделі Random Forest

На завершальному етапі модуль прийняття рішень отримує результат класифікації та, залежно від типу виявленої події, ініціює відповідну реакцію системи: генерацію сповіщення, запис події до журналу безпеки або передачу інформації до модуля візуалізації.

В якості ілюстрації реалізації функціональних модулів системи засобами Python наведемо опис деяких з них. Інтелектуальний модуль розташовується між модулем попередньої обробки та модулем зберігання/візуалізації. Його функціональні обов’язки включають:

– завантаження навченої моделі з файлового сховища;
– приймання векторів ознак мережевих потоків;
– виконання класифікації у режимі реального часу або пакетної обробки;
– передавання результатів аналізу до наступних модулів системи.

Модуль реалізовано таким чином, щоб його можна було легко замінити або

розширити іншими моделями без зміни загальної архітектури.

Після завершення процесу навчання та валідації модель Random Forest зберігається у серіалізованому вигляді з використанням бібліотеки Joblib. Це дозволяє уникнути повторного навчання під час запуску системи та значно зменшує час ініціалізації.

Після завантаження моделі інтелектуальний модуль отримує на вхід вектори ознак, сформовані модулем попередньої обробки. Кожен вектор відповідає окремому мережевому потоку.

Лістинг 1. – Класифікація мережевого трафіку

```
def classify_flow(model, feature_vector):
    prediction = model.predict([feature_vector])
    probability = model.predict_proba([feature_vector])

    return {
        "class": int(prediction[0]),
        "confidence": float(max(probability[0]))
    }
```

У результаті класифікації визначається клас потоку (нормальний або аномальний), а також рівень впевненості моделі у прийнятому рішенні.

Для підвищення продуктивності система підтримує обробку потоків у пакетному режимі, що дозволяє аналізувати великі обсяги трафіку.

Модуль збору мережевого трафіку реалізовано як незалежний компонент, який функціонує асинхронно відносно інших модулів системи. Зібрані пакети передаються у стандартизованому форматі (словник або структура даних), що забезпечує слабку зв’язаність між модулями та спрощує масштабування системи. Нижче наведено приклад програмної реалізації модуля.

Лістинг 2. – Реалізація перехоплення трафіку з використанням Scapy

```
from scapy.all import sniff, IP, TCP, UDP
from datetime import datetime

def packet_handler(packet):
    if IP in packet:
        packet_data = {
            "timestamp": datetime.now(),
            "src_ip": packet[IP].src,
            "dst_ip": packet[IP].dst,
            "protocol": packet[IP].proto,
            "packet_length": len(packet)
        }

        if TCP in packet:
            packet_data["src_port"] = packet[TCP].sport
            packet_data["dst_port"] = packet[TCP].dport
            packet_data["flags"] = str(packet[TCP].flags)
        elif UDP in packet:
            packet_data["src_port"] = packet[UDP].sport
            packet_data["dst_port"] = packet[UDP].dport

        # Передавання даних до модуля попередньої обробки
```

```
        process_packet(packet_data)

def process_packet(packet_data):
    # Заглушка для подальшої обробки
    print(packet_data)

if __name__ == "__main__":
    sniff(iface="eth0", prn=packet_handler, store=False)
```

Модуль попередньої обробки даних є проміжною, але критично важливою ланкою між модулем збору мережевого трафіку та інтелектуальним модулем аналізу. Його основне завдання полягає у перетворенні сирих пакетних даних у структурований набір ознак, придатний для подальшого використання алгоритмами машинного навчання.

На практиці мережеві пакети мають низький рівень абстракції та містять надлишкову інформацію, тому без етапу попередньої обробки ефективне навчання моделі є неможливим. Саме тому у даній роботі реалізовано класичний підхід, що включає агрегацію пакетів у потоки та виділення статистичних ознак.

Першим етапом попередньої обробки є об’єднання окремих пакетів у логічні мережеві потоки. Потік визначається п’ятіркою параметрів:

- IP-адреса джерела;
- IP-адреса призначення;
- порт джерела;
- порт призначення;
- транспортний протокол.

Агрегація за такими параметрами дозволяє аналізувати поведінку з’єднання у часі, а не окремі пакети, що є більш інформативним для задач виявлення атак.

Для кожного сформованого потоку обчислюються числові та категоріальні ознаки, які характеризують його поведінку. У межах роботи використовуються такі групи ознак:

- часові характеристики: тривалість потоку;
- об’ємні характеристики: кількість пакетів, сумарний обсяг байтів;
- статистичні показники: середній розмір пакета;
- протокольні ознаки: тип протоколу, TCP-прапори;
- напрямок трафіку: вхідний або вихідний потік.

Такі ознаки широко застосовуються у сучасних наборах даних для IDS та добре підходять для алгоритмів типу Random Forest.

У наведеному фрагменті коду всі пакети групуються у словнику flows за ключем, що відповідає ідентифікатору потоку.

Лістинг 3.– Програмна реалізація агрегації потоків

```
from collections import defaultdict
from datetime import datetime

flows = defaultdict(list)

def add_packet_to_flow(packet):
    flow_key = (
        packet.get("src_ip"),
        packet.get("dst_ip"),
        packet.get("src_port"),
```

```
            packet.get("dst_port"),
            packet.get("protocol")
        )
        flows[flow_key].append(packet)
```

Після накопичення пакетів для кожного потоку виконується розрахунок вектора ознак (лістинг 4).

Лістинг 4. – Обчислення ознак для потоку

```
    def extract_flow_features(flow_packets):
        timestamps = [p["timestamp"] for p in flow_packets]
        sizes = [p["packet_length"] for p in flow_packets]

        duration                =                max(timestamps)-
min(timestamps)).total_seconds()
        total_packets = len(flow_packets)
        total_bytes = sum(sizes)
        avg_packet_size   =   total_bytes   /   total_packets   if
total_packets > 0 else 0

        return {
            "duration": duration,
            "total_packets": total_packets,
            "total_bytes": total_bytes,
            "avg_packet_size": avg_packet_size
        }
```

У результаті для кожного мережевого потоку формується компактний набір числових характеристик, що описує його поведінку.

На фінальному етапі результати обробки потоків перетворюються у табличний формат з використанням бібліотеки Pandas, що є стандартним представленням даних для подальшого навчання моделей машинного навчання (лістинг 5).

Лістинг 5.– Програмна реалізація перетворення даних

```
    import pandas as pd

    feature_rows = []

    for flow_key, packets in flows.items():
        features = extract_flow_features(packets)
        feature_rows.append(features)

    df = pd.DataFrame(feature_rows)
```

Отримана таблиця містить по одному рядку на кожен мережевий потік та може бути безпосередньо використана для навчання, тестування або онлайн-класифікації.

Коротко відмітимо, що у межах даної роботи для реалізації модуля зберігання обрано реляційну систему управління базами даних MySQL, а для реалізації візуалізації використовуються бібліотеки Matplotlib і Seaborn, які дозволяють будувати статичні та

напівінтерактивні графіки;

Для кількісної оцінки ефективності розробленої системи виявлення вторгнень було проведено серію експериментів на тестовій вибірці мережевого трафіку, що містить як нормальні з’єднання, так і різні типи атак. Оцінювання якості здійснювалося з використанням стандартних метрик класифікації, які дозволяють комплексно проаналізувати точність та надійність роботи інтелектуального модуля аналізу.

За результатами тестування було отримано матрицю помилок, наведену в таблиці 2.

Таблиця 2. – Матриця помилок для тестової вибірки

	Прогноз: Норма	Прогноз: Атака
Факт: Норма	9 420	380
Факт: Атака	210	5 990

На основі наведених даних можна зробити такі висновки:

– більшість шкідливих з’єднань було коректно виявлено (TP = 5 990);

– кількість хибно-негативних результатів (FN = 210) є відносно невеликою, що свідчить про високу здатність системи виявляти атаки;

– наявність хибно-позитивних спрацювань (FP = 380) є допустимою для систем моніторингу та може бути зменшена подальшим налаштуванням порогів або розширенням набору ознак.

На основі отриманої матриці помилок були обчислені основні метрики якості класифікації.

Accuracy (загальна точність):

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} = \frac{5990 + 9420}{16000} \approx 0{,}963 \quad (4.1)$$

Отримане значення Accuracy ≈ 96,3 % свідчить про високий загальний рівень правильності класифікації.

Precision (прецизійність):

$$Precision = \frac{TP}{TP + FP} = \frac{5990}{5990 + 380} \approx 0{,}940 \quad (4.2)$$

Це означає, що приблизно 94 % з’єднань, класифікованих системою як атаки, дійсно є шкідливими.

Recall (повнота):

$$Recall = \frac{TP}{TP + FN} = \frac{5990}{5990 + 210} \approx 0{,}966 \quad (4.3)$$

Високе значення Recall (96,6 %) вказує на ефективне виявлення більшості атак, що є критично важливим для систем безпеки.

F1-score:

$$F1 = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall} \approx 0{,}953 \quad (4.2)$$

Отриманий F1-score підтверджує збалансованість моделі між точністю та повнотою.

Продуктивність системи оцінювалася за часом обробки мережевих потоків та загальним часом аналізу тестового набору даних. У процесі експериментів було встановлено, що:

– середній час обробки одного агрегованого мережевого потоку становить близько 1,8–2,3 мс;
– повний аналіз тестової вибірки обсягом 16 000 записів виконується менш ніж за 40 секунд;
– система здатна обробляти декілька тисяч потоків за хвилину без суттєвого зниження точності.

Отримані результати свідчать про можливість використання розробленої системи в умовах, наближених до реального часу, зокрема для моніторингу корпоративних мереж середнього розміру.

IV. ВИСНОВКИ

Аналіз сучасного стану реалізацій систем мережевого моніторингу та виявлення вторгнень, зокрема сигнатурних систем, систем аналізу потоків та комерційних рішень з елементами штучного інтелекту показав обмеженість класичних підходів до виявлення нових різновидів атак, тоді як інтелектуальні методи забезпечують вищу адаптивність, але потребують при цьому більш коректного вибору даних і моделей.

Обґрунтований вибір датасетів, що містять реалістичні сценарії атак, та практична реалізація технологічного стеку на основі мови Python і бібліотек Scapy, Pandas, NumPy та Scikit-learn, дозволив вирішити комплексну науково-практичну задачу розробки інтелектуальної системи моніторингу мережевого трафіку та виявлення вторгнень із використанням методів машинного навчання.

Реалізовані в системі механізм перехоплення трафіку, інженерія ознак, нормалізація даних та класифікація потоків із використанням алгоритму Random Forest дає змогу отримати низький рівень хибно-позитивних спрацювань навіть у багатокласовому режимі, що є критично важливим для практичного застосування системи, оскільки надмірна кількість помилкових тривог знижує довіру до системи безпеки. Аналіз хибно-негативних результатів засвідчив, що більшість атак виявляється коректно, а помилки здебільшого пов’язані зі схожістю окремих класів атак між собою.

Таким чином, розроблена система не поступається традиційним IDS за точністю виявлення відомих атак, а в частині адаптивності до різних типів трафіку та зменшення хибних спрацювань демонструє кращі результати. На відміну від сигнатурних систем, запропоноване рішення не потребує постійного ручного оновлення правил і здатне узагальнювати поведінкові ознаки атак.

Розроблену систему доцільно рекомендувати для практичного використання у корпоративних мережах малого та середнього масштабу як допоміжний засіб моніторингу безпеки, а також у навчальних та дослідницьких цілях. Модульна архітектура дозволяє інтегрувати систему з існуючими засобами безпеки, зокрема SIEM-платформами та системами реагування на інциденти.

V. ПЕРСПЕКТИВИ ПОДАЛЬШИХ ДОСЛІДЖЕНЬ

Подальший розвиток роботи може бути спрямований на використання більш складних моделей глибокого навчання (LSTM, CNN, Autoencoder), дослідження методів аналізу зашифрованого трафіку, розширення набору ознак за рахунок поведінкових та

часових характеристик, а також розробку модуля автоматичного реагування на інциденти безпеки в реальному часі.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

[1] European Union Agency for Cybersecurity. ENISA Threat Landscape 2024: The year in review. – Luxembourg: Publications Office of the European Union, 2024. – 134 p. ISBN 978-92-9204-633-6.

[2] Verizon Business. 2024 Data Breach Investigations Report (DBIR) [Електронний ресурс]. – 2024. – Режим доступу: https://www.verizon.com/business/resources/reports/dbir/ (дата звернення: 01.12.2025).

[3] Rose S. P., Borchert O. M., St.E. F. S. M. et al. Zero Trust Architecture // NIST Special Publication (SP) 800-207. – Gaithersburg: National Institute of Standards and Technology, 2020. – 50 p.

[4] Wagner R., Orans C. et al. Top Strategic Cybersecurity Trends for 2024 [Електронний ресурс]. – Gartner Research, 2024. – Режим доступу: https://www.gartner.com/en/articles/gartner-identifies-the-top-cybersecurity-trends-for-2024 (дата звернення: 01.12.2025).

[5] Fagan M. F., S. E. S. M. et al. IoT Device Cybersecurity Guidance for the Federal Government: Establishing IoT Device Cybersecurity Requirements // NIST Special Publication (SP) 800-213. – Gaithersburg: National Institute of Standards and Technology, 2021. – 46 p.

[6] Gartner Research. Implement a Continuous Threat Exposure Management (CTEM) Program [Електронний ресурс]. – 2023. – Режим доступу: https://www.gartner.com/en/cybersecurity/topics/continuous-threat-exposure-management (дата звернення: 02.12.2025).

[7] UCI Machine Learning Repository. KDD Cup 1999 Data [Електронний ресурс]. – 1999. – Режим доступу: https://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html (дата звернення: 07.12.2025).

[8] Canadian Institute for Cybersecurity (University of New Brunswick). NSL-KDD dataset [Електронний ресурс]. – Режим доступу: https://www.unb.ca/cic/datasets/nsl.html (дата звернення: 10.12.2025).

[9] Canadian Institute for Cybersecurity. CICIDS2017 dataset [Електронний ресурс]. – 2017. – Режим доступу: https://www.unb.ca/cic/datasets/ids-2017.html (дата звернення: 06.12.2025).

[10] Moustafa N., Slay J. UNSW-NB15: a comprehensive data set for network intrusion detection systems [Електронний ресурс] // Military Communications and Information Systems Conference (MilCIS). – 2015. – P. 1–6. – Режим доступу: https://research.unsw.edu.au/projects/unsw-nb15-dataset (дата звернення: 06.12.2025).

[11] Scapy Community. Scapy: Packet crafting for Python [Електронний ресурс]. – Режим доступу: https://scapy.net (дата звернення: 11.12.2025).

[12] PyData. Pandas: Python Data Analysis Library [Електронний ресурс]. – Режим доступу: https://pandas.pydata.org/docs/ (дата звернення: 12.12.2025).

[13] Al-Mansoori S. et al. Optimizing Cybersecurity: A Dual Phase ML Architecture for Detection and Classification of Network Attacks using TensorFlow // International Journal of Applied Mathematics and Computer Science. – 2025. – Vol. 35, No. 1.

[14] Google Brain Team. TensorFlow: An Open Source Machine Learning Framework for Everyone [Електронний ресурс]. – 2025. – Режим доступу: https://www.tensorflow.org (дата звернення: 13.12.2025).

[15] PyTorch Foundation. PyTorch: From Research to Production [Електронний ресурс]. – 2025. – Режим доступу: https://pytorch.org (дата звернення: 13.12.2025).

[16] Shah S. A. R., Issac B. Performance Comparison of Intrusion Detection Systems and Application of Machine Learning to Snort System // Future Generation Computer Systems. – 2018. – Vol. 80. – P. 157–170.

[17] CADL: Cognitive-Adaptive Deception Layer [Електронний ресурс] // arXiv preprint arXiv:2510.02424. – 2025. – Режим доступу: https://arxiv.org/abs/2510.02424 (дата звернення: 14.12.2025).

Отримано 10.12.2025 р.

NETWORK ACTIVITY MONITORING SYSTEM USING ELEMENTS OF ARTIFICIAL INTELLIGENCE

Kurashvili Dmitry
Serhii Sabanov,
Department of Information Technologies
Zaporizhzhia Institute of Economics and Information Technologies
Zaporizhzhia, Ukraine

Abstract - The paper considers issues related to solving the complex scientific and practical problem of developing an intelligent network traffic monitoring system to detect intrusions using machine learning methods. The paper covers the analysis of modern approaches to network security, design and software implementation of key system modules, as well as experimental research into the effectiveness of the proposed solution. Analysis of the state of network monitoring and intrusion detection systems, in particular signature systems (Snort, Suricata) and flow analysis systems (NetFlow, sFlow) showed that classical approaches are characterized by limited ability to detect new and modified attacks, while intelligent methods provide higher adaptability, but require the correct selection of data and models. In this regard, a system was proposed that uses modern datasets with realistic attack scenarios. The system is created on the basis of a technological stack implemented in Python using the Scapy, Pandas, NumPy and Scikit-learn libraries. A traffic interception mechanism, feature engineering, data normalization, and flow classification using the Random Forest algorithm are implemented. Unlike signature systems, the proposed solution does not require constant manual updating of rules and is able to generalize behavioral signs of attacks.

Keywords*: IDS/IPS, NMS, SIEM, machine learning, network attack, network traffic monitoring, signature analysis, artificial intelligence*

Recieved 10.12.2025 p.