Вінницький національний технічний університет

Факультет менеджменту та інформаційної безпеки

Кафедра менеджменту та безпеки інформаційних систем

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему:

Удосконалення системи виявлення мережевих атак шляхом застосування гібридного підходу машинного навчання з динамічним налаштуванням порогових значень

Виконав: здобувач 2-го курсу,
групи 1КІТС-24м
спеціальності 125– Кібербезпека
та захист інформації
Освітня програма – Кібербезпека
інформаційних технологій та систем
(шифр і назва напряму підготовки, спеціальності)

Чуйко Н. В.
(прізвище та ініціали)

Керівник:
к.ф-м.н., доц. каф. МБІС, Шиян А. А.
(прізвище та ініціали)

«11» грудня 2025 р.

Опонент:
(прізвище та ініціали)
«___» _____ 2025 р.

Допущено до захисту
Голова секції УБ кафедри МБІС

Юрій ЯРЕМЧУК
“11” 12 2025 р.

Вінниця ВНТУ - 2025 рік

Вінницький національний технічний університет

Факультет менеджменту та інформаційної безпеки
Кафедра менеджменту та безпеки інформаційних систем

Рівень вищої освіти ІІ-й (магістерський)
Галузь знань 12 – Інформаційні технології
Спеціальність 125 – Кібербезпека та захист інформації
Освітньо-професійна програма - Кібербезпека інформаційних технологій та систем

ЗАТВЕРДЖУЮ
Голова секції УБ, кафедра МБІС
_____**Юрій ЯРЕМЧУК**
“ 24 ” вересня 2025 р.

З А В Д А Н Н Я
на магістерську кваліфікаційну роботу студенту
Чуйко Нікіта Валерійович
(прізвище, ім`я, по-батькові)

1. Тема роботи

Удосконалення системи виявлення мережевих атак шляхом застосування гібридного підходу машинного навчання з динамічним налаштуванням порогових значень.

Керівник роботи_____к.ф-м.н., доц. каф. МБІС Шиян А.А._____
(прізвище, ім`я, по-батькові, науковий ступінь, вчене звання)
затверджені наказом вищого навчального закладу від “24” вересня 2025 року № 313
2. Строк подання студентом роботи за тиждень до захисту.
3. Вихідні дані до роботи:
Стандарти, електронні джерела, підручники та наукові статті по темі, які
4. Зміст текстової частини:
1. Аналіз сучасних методів та засобів виявлення мережевих атак 2. Розробка гібридного методу виявлення мережевих атак 3. Практична реалізація та верифікація системи виявлення мережевих атак 4. Економічне обґрунтування розробки Висновки Список використаних джерел
5. Перелік ілюстративного матеріалу (з точним зазначенням обов`язкових креслень)
В першому розділі магістерської роботи 0 рисунків, в другому розділі 5 рисунків, в третьому розділі 6 рисунків

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Основна частина			
I	Шиян А.А. к.ф-м.н., доцент кафедри МБІС.		
II	Шиян А.А. к.ф-м.н., доцент кафедри МБІС		
III	Шиян А.А. к.ф-м.н., доцент кафедри МБІС		
Економічна частина			
IV	Ратушняк О.Г. к.т.н., доцент кафедри ЕПВМ		

7.Дата видачі завдання 24 вересня 2025 р.

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів магістерської кваліфікаційної роботи	Строк виконання етапів роботи		Примітка
1	Визначення напрямку, формулювання теми та отримання завдання	24.09.2025	30.09.2025	
2	Аналіз предметної області та сучасних методів виявлення атак (Розділ 1)	01.10.2025	15.10.2025	
3	Розробка гібридного методу та архітектури системи (Розділ 2)	16.10.2025	31.10.2025	
4	Програмна реалізація, навчання моделей та експериментальне дослідження (Розділ 3)	01.11.2025	20.11.2025	
5	Економічне обгрунтування розробки (Розділ 4)	21.11.2025	30.11.2025	
6	Оформлення пояснювальної записки, висновків та презентації	01.12.2025	07.12.2025	
7	Захист магістерської кваліфікаційної роботи	08.12.2025	12.12.2025	

Студент _____ (підпис) Чуєсо Н.В

Керівник роботи _____ (підпис) А.А. Шиян

АНОТАЦІЯ

УДК 004.056.53

Чуйко Н. В. Удосконалення системи виявлення мережевих атак шляхом застосування гібридного підходу машинного навчання з динамічним налаштуванням порогових значень. Магістерська кваліфікаційна робота зі спеціальності 125 – Кібербезпека та захист інформації, освітня програма – Кібербезпека інформаційних технологій та систем. Вінниця: ВНТУ, 2025. 138 с.

На укр. мові. Бібліогр.: 69 назв. Рис.: 14. Табл.: 7.

Кваліфікаційна робота присвячена дослідженню та удосконаленню методів виявлення мережевих атак шляхом застосування гібридного підходу машинного навчання. У роботі проведено системний аналіз сучасних систем виявлення вторгнень, досліджено їх обмеження у контексті динамічності мережевого трафіку та сформовано концепцію гібридної системи. Запропонований метод включає використання автоенкодера для виявлення аномальних відхилень у поведінці трафіку та ансамблевих моделей Random Forest і CNN-1D, що забезпечують багаторівневе оцінювання ризиків і підвищують точність детектування. Розроблено математичну модель динамічного налаштування порогових значень, яка адаптує систему до змін профілю мережевих потоків. Практичну частину роботи присвячено реалізації програмного комплексу, створенню експериментального середовища та проведенню порівняльного аналізу ефективності гібридного підходу щодо точності, стійкості та рівня хибнопозитивних спрацювань. Отримані результати підтверджують переваги гібридних систем над традиційними підходами та можуть бути використані в системах захисту інформації підприємств і організацій.

Ключові слова: мережеві атаки, гібридні системи виявлення вторгнень, машинне навчання, автоенкодер Random Forest CNN-1D, динамічне налаштування порогів, кібербезпека

ABSTRACT

The thesis is devoted to the development and enhancement of network attack detection mechanisms through a hybrid machine learning approach that integrates anomaly detection and classification models. The research includes an extensive analysis of modern intrusion detection systems, identification of their limitations in dynamic traffic environments, and an examination of machine learning methods used for detecting cyberattacks. The proposed hybrid method combines an autoencoder for anomaly recognition with ensemble models such as Random Forest and one-dimensional Convolutional Neural Networks, ensuring multi-level risk assessment and increased detection accuracy. A mathematical model for dynamic threshold adjustment is developed to provide system adaptability to changing network traffic characteristics. The practical component of the research includes the design and implementation of a software system, construction of an experimental environment, and comparative evaluation of the efficiency of the hybrid approach in terms of accuracy, robustness, and false-positive rates. The findings demonstrate significant improvements over traditional intrusion detection solutions and can be applied within information security infrastructures of various institutions.

Keywords: network attacks, hybrid intrusion, detection systems, machine learning, autoencoder, Random Forest, CNN-1D, dynamic, thresholding cybersecurity

ЗМІСТ

ВСТУП

Актуальність теми дослідження. Стрімкий розвиток інформаційних технологій та глобальна цифровізація усіх сфер суспільного життя супроводжуються експоненційним зростанням кількості та складності кібератак на корпоративні мережі, критичну інфраструктуру та державні інформаційні системи. За даними звіту IBM Security X-Force Threat Intelligence Index 2024, середня кількість кібератак на одну організацію зросла на 38% порівняно з попереднім роком, при цьому фінансові збитки від успішних вторгнень досягли рекордного показника 4,45 мільйона доларів США за один інцидент. В умовах гібридної війни проти України кіберзагрози набувають особливої гостроти: за статистикою Державної служби спеціального зв'язку та захисту інформації, протягом 2023–2024 років зафіксовано понад 4500 критичних кіберінцидентів, спрямованих на об'єкти енергетичної, фінансової та телекомунікаційної інфраструктури держави.

Системи виявлення вторгнень (Intrusion Detection Systems, IDS) залишаються ключовим компонентом багаторівневої архітектури кіберзахисту, забезпечуючи моніторинг мережевого трафіку та ідентифікацію підозрілої активності в режимі реального часу. Водночас традиційні підходи до побудови IDS демонструють системні обмеження, що суттєво знижують їх ефективність в умовах сучасного ландшафту загроз. Сигнатурні системи, попри високу точність виявлення відомих атак, принципово неспроможні детектувати загрози нульового дня та модифіковані варіанти шкідливого програмного забезпечення. Аномальні системи, базовані на статистичних методах або окремих алгоритмах машинного навчання, характеризуються неприйнятно високим рівнем хибнопозитивних спрацювань, що досягає 15–30% і призводить до «втоми» адміністраторів безпеки та ігнорування реальних інцидентів.

Особливо гострою проблемою залишається використання фіксованих порогових значень для класифікації мережевої активності як нормальної або аномальної. Статичні пороги, встановлені на етапі налаштування системи, не

враховують динамічних змін профілю легітимного трафіку, сезонних коливань навантаження, еволюції бізнес-процесів організації та адаптивних стратегій зловмисників. Як наслідок, системи з фіксованими порогами втрачають ефективність протягом 3–6 місяців експлуатації без ручного перенавчання, що створює «вікна вразливості» та збільшує операційні витрати на супровід.

Перспективним напрямом вирішення окреслених проблем є застосування гібридних підходів машинного навчання, що поєднують переваги різних класів алгоритмів — ансамблевих методів класифікації, глибоких нейронних мереж та автокодувальників для виявлення аномалій — з механізмами динамічного налаштування порогових значень на основі аналізу актуального стану мережевого середовища. Такий підхід дозволяє досягти синергетичного ефекту: висока точність ансамблевих класифікаторів доповнюється здатністю автокодувальників ідентифікувати невідомі атаки, а адаптивні пороги забезпечують стабільну ефективність системи в умовах змінного профілю трафіку.

Таким чином, розробка удосконаленої системи виявлення мережевих атак на основі гібридного підходу машинного навчання з динамічним налаштуванням порогових значень є актуальним науково-практичним завданням, що має суттєве значення для підвищення рівня кібербезпеки організацій різних форм власності та масштабу діяльності.

Мета і завдання дослідження. Метою кваліфікаційної роботи є підвищення ефективності виявлення мережевих атак шляхом розробки гібридної системи на основі ансамблевих методів машинного навчання та глибоких нейронних мереж з механізмом динамічного налаштування порогових значень.

Для досягнення поставленої мети необхідно вирішити такі завдання:

— провести аналіз сучасного стану та тенденцій розвитку систем виявлення мережевих вторгнень, систематизувати їх класифікацію, переваги та обмеження;

— дослідити методи машинного навчання, що застосовуються для виявлення кібератак, та визначити можливості їх гібридизації;

— розробити удосконалений метод виявлення мережевих атак на основі гібридного підходу, що поєднує ансамблеві класифікатори, автокодувальники та згорткові нейронні мережі;

— розробити алгоритм динамічного налаштування порогових значень класифікації з урахуванням актуального стану мережевого середовища;

— здійснити програмну реалізацію розробленої гібридної системи виявлення мережевих атак;

— провести експериментальне дослідження та порівняльний аналіз ефективності розробленої системи на еталонних наборах даних;

— обґрунтувати економічну доцільність впровадження розробленої системи.

Об'єкт дослідження — процес виявлення мережевих атак та аномалій у комп'ютерних мережах.

Предмет дослідження — методи та алгоритми гібридного машинного навчання з динамічним налаштуванням порогових значень для підвищення ефективності систем виявлення вторгнень.

Методи дослідження. Для вирішення поставлених завдань застосовано комплекс загальнонаукових та спеціальних методів дослідження: системний аналіз — для дослідження архітектури сучасних IDS та визначення напрямів їх удосконалення; порівняльний аналіз — для оцінки ефективності різних алгоритмів машинного навчання; методи математичної статистики — для обробки результатів експериментів та оцінки статистичної значущості отриманих результатів; методи теорії ймовірностей — для побудови моделей адаптивних порогових значень; методи глибокого навчання — для реалізації автокодувальників та згорткових нейронних мереж; ансамблеві методи машинного навчання — для побудови композитних класифікаторів; методи експериментального дослідження — для верифікації розробленої системи на еталонних наборах даних.

Наукова новизна одержаних результатів полягає в такому:

— запропоновано гібридний метод виявлення мережевих атак, що поєднує ансамблевий класифікатор Random Forest, автокодувальник для детектування аномалій та одновимірну згорткову нейронну мережу CNN-1D для аналізу часових патернів трафіку, що на відміну від існуючих підходів забезпечує одночасне виявлення відомих та невідомих атак з точністю понад 98%;

— удосконалено алгоритм динамічного налаштування порогових значень класифікації, який на відміну від статичних порогів враховує ковзну статистику відхилень реконструкції автокодувальника та ймовірнісні оцінки ансамблевого класифікатора, що дозволяє знизити рівень хибнопозитивних спрацювань на 45–60% порівняно з фіксованими порогами;

— набула подальшого розвитку архітектура гібридних систем виявлення вторгнень шляхом інтеграції модуля попередньої обробки з адаптивною нормалізацією ознак, що забезпечує стійкість до зміни розподілу вхідних даних (concept drift).

Практичне значення одержаних результатів полягає в розробці програмного комплексу «AEGIS IDS», що реалізує запропонований гібридний метод виявлення мережевих атак та може бути впроваджений в інформаційно-телекомунікаційних системах підприємств, організацій та установ для підвищення рівня кібербезпеки. Розроблена система забезпечує: виявлення широкого спектру мережевих атак (DoS/DDoS, сканування портів, SQL-ін'єкції, XSS, brute-force) з точністю понад 98%; зниження рівня хибнопозитивних спрацювань до 2–3%; автоматичну адаптацію до змін профілю мережевого трафіку без ручного перенавчання; інтуїтивний веб-інтерфейс для моніторингу та управління. Результати роботи можуть бути використані в навчальному процесі при викладанні дисциплін «Кібербезпека», «Машинне навчання», «Захист інформації в комп'ютерних системах».

РОЗДІЛ 1. АНАЛІЗ СУЧАСНИХ МЕТОДІВ ТА ЗАСОБІВ ВИЯВЛЕННЯ МЕРЕЖЕВИХ АТАК

1.1 Класифікація мережевих атак та огляд існуючих систем виявлення вторгнень

Інтенсивний розвиток Інтернету речей та розподілених обчислювальних архітектур спричинив кардинальну трансформацію ландшафту кіберзагроз, що вимагає переосмислення традиційних підходів до забезпечення інформаційної безпеки. Дослідження А. Гупти та М. Кумари свідчать про експоненціальне зростання кількості підключених пристроїв, яке супроводжується відповідним збільшенням поверхні атак та диверсифікацією векторів вторгнень [16]. Одночасно Д. Куїчем зі співавторами зазначають, що гетерогенність IoT-екосистем, обмежені обчислювальні ресурси вузлів та відсутність стандартизованих протоколів безпеки створюють унікальні виклики для традиційних механізмів детектування аномалій [36]. Ф. Аль-Турджман та його колеги підкреслюють критичну роль малих комірок у архітектурі 5G/IoT, демонструючи складність моделювання трафіку в умовах масштабованих мережевих топологій [6].

Таксономія мережевих атак охоплює широкий спектр загроз, кожна з яких характеризується унікальними сигнатурами поведінки та потенціалом завдання шкоди інфраструктурі. Атаки типу «відмова в обслуговуванні» становлять одну з найбільш поширених категорій загроз, спрямованих на виснаження системних ресурсів через генерацію масивних обсягів несанкціонованого трафіку. Розподілені DDoS-атаки використовують ботнети для координованого перевантаження цільових серверів, експлуатуючи вразливості протоколів транспортного та мережевого рівнів моделі OSI [48]. Розвиток складніших варіантів таких атак включає багаторівневі сценарії, що поєднують перевантаження на прикладному рівні з експлуатацією вразливостей протоколу HTTP/HTTPS. Атаки типу «людина посередині»

реалізують перехоплення та модифікацію даних між легітимними учасниками комунікації, компрометуючи конфіденційність та цілісність інформаційних обмінів через підміну ARP-записів або DNS-спуфінг. Ін'єкції SQL-коду експлуатують недоліки валідації вхідних даних для несанкціонованого доступу до баз даних, що може призвести до витоку конфіденційної інформації або повної компрометації системи управління даними.

Класифікація методів виявлення вторгнень традиційно розділяє системи на дві основні парадигми. Сигнатурні підходи базуються на зіставленні спостережуваних патернів мережевої активності з попередньо визначеними шаблонами відомих атак, забезпечуючи високу точність детектування при мінімальній кількості хибних спрацювань [40]. Проте обмеженість баз сигнатур унеможливлює ідентифікацію нових типів загроз, що становить критичний недолік в умовах постійної еволюції ландшафту кіберзагроз. Аномальні системи виявлення вторгнень використовують статистичні моделі або алгоритми машинного навчання для встановлення базової лінії нормальної поведінки системи, детектуючи відхилення як потенційні індикатори компрометації [41]. Така методологія демонструє ефективність у виявленні раніше невідомих атак, проте характеризується підвищеним рівнем хибнопозитивних спрацювань та вимагає тривалого періоду навчання для формування адекватних моделей нормальної активності.

Гібридні архітектури інтегрують переваги обох підходів, створюючи багаторівневі системи детектування. Початковий етап фільтрації використовує сигнатурний аналіз для швидкого виявлення відомих патернів атак з мінімальними обчислювальними витратами, тоді як аномальний компонент аналізує трафік, що пройшов первинну фільтрацію, для ідентифікації підозрілих відхилень від встановлених норм поведінки [42]. Така стратифікована архітектура оптимізує баланс між точністю детектування та обчислювальною ефективністю, проте вимагає складних механізмів координації між компонентами та динамічного налаштування параметрів кожного рівня аналізу.

Архітектурні парадигми розміщення систем виявлення вторгнень охоплюють мережецентричні, хостоцентричні та гібридні конфігурації. Мережеві IDS розгортаються на критичних вузлах інфраструктури, аналізуючи пакети даних у режимі реального часу через моніторинг дзеркалювання портів або inline-перехоплення трафіку [43]. Хост-базовані системи інтегруються безпосередньо в операційні системи захищуваних вузлів, моніторячи системні виклики, журнали подій та модифікації файлової системи для виявлення індикаторів компрометації на рівні окремих хостів [44]. Розподілені архітектури координують множину агентів детектування через централізовану систему управління, агрегуючи та корелюючи дані з різних джерел для формування цілісної картини стану безпеки розподіленої інфраструктури.

Таблиця 1.1 – Порівняльна характеристика архітектур систем виявлення вторгнень

Архітектурний підхід	**Переваги**	**Обмеження**	**Застосування**
Мережецентричний	Централізований моніторинг, незалежність від платформи хостів, відсутність навантаження на кінцеві пристрої	Неможливість аналізу зашифрованого трафіку, обмежена видимість активності на рівні хостів, складність масштабування	Периметрова безпека, моніторинг магістральних каналів, захист серверних сегментів
Хостоцентричний	Глибокий аналіз поведінки системи, детектування атак на рівні додатків, моніторинг локальних подій	Навантаження на ресурси хосту, необхідність підтримки множини платформ, відсутність мережевого контексту	Захист критичних серверів, моніторинг робочих станцій, контроль цілісності системи
Гібридний розподілений	Комплексна видимість на всіх рівнях, корреляція різнорідних подій, адаптивність до топології мережі	Висока складність архітектури, значні вимоги до пропускної здатності каналів зв'язку, складність синхронізації	Великі корпоративні мережі, критична інфраструктура, хмарні середовища

Технологічна еволюція методів виявлення вторгнень характеризується поступовим переходом від статистичних підходів до складних архітектур глибокого навчання. Ранні системи базувалися на простих порогових механізмах та статистичному аналізі відхилень, використовуючи параметричні методи для характеризації нормального трафіку [49]. Впровадження алгоритмів машинного навчання дозволило створювати більш складні моделі класифікації, здатні адаптуватися до еволюції патернів атак через періодичне перенавчання на нових даних. Алгоритми дерев рішень демонстрували високу інтерпретованість результатів, тоді як методи опорних векторів забезпечували ефективну класифікацію в високорозмірних просторах ознак [29]. Застосування ансамблевих методів значно покращило робастність систем детектування через агрегацію прогнозів множини базових класифікаторів. Випадкові ліси комбінують багатозначні дерева рішень, тренованих на різних підмножинах даних та ознак, що забезпечує зниження дисперсії прогнозів та підвищення узагальнювальної здатності моделей [14].

Градієнтний бустинг послідовно будує ансамблі слабких класифікаторів, мінімізуючи функцію втрат через ітеративне покращення помилок попередніх моделей, що дозволяє ефективно обробляти складні нелінійні залежності в даних мережевого трафіку. Стекінг інтегрує гетерогенні алгоритми навчання через мета-класифікатор, що навчається оптимально комбінувати прогнози базових моделей для максимізації загальної точності детектування.

Революція глибокого навчання трансформувала підходи до виявлення аномалій завдяки здатності автоматично вивчати ієрархічні репрезентації з сирих даних без необхідності ручного конструювання ознак. Згорткові нейронні мережі демонструють ефективність в екстракції просторових паттернів з мережевого трафіку, представленого у вигляді двовимірних зображень або матриць потоків [51]. Рекурентні архітектури, зокрема довга короткочасна пам'ять, спеціалізуються на моделюванні темпоральних залежностей в послідовностях мережевих подій, що критично важливо для детектування багатоетапних атак з розтягнутою в часі реалізацією [13].

Автоенкодери формують нелінійні представлення даних через навчання реконструкції вхідних векторів через вузьке горловинне представлення, використовуючи помилку реконструкції як міру аномальності [28]. Варіаційні автоенкодери розширюють базову архітектуру через введення імовірнісних латентних репрезентацій, що дозволяє моделювати розподіли нормального трафіку та обчислювати імовірності належності нових спостережень до нормального класу. Генеративно-змагальні мережі синтезують реалістичні зразки мережевого трафіку через мінімаксну гру між генератором та дискримінатором, що може використовуватися як для аугментації тренувальних даних, так і для детектування аномалій через оцінку якості реконструкції спостережень [59].

Таблиця 1.2 – Характеристики алгоритмів машинного навчання для виявлення вторгнень

Сімейство алгоритмів	**Представники**	**Принцип роботи**	**Обчислювальна складність**	**Інтерпретованість**
Дерева рішень	CART, C4.5, ID3	Рекурсивне розбиття простору ознак за критеріями інформаційного приросту	$O(n \cdot m \cdot \log(n))$	Висока
Ансамблеві методи	Random Forest, XGBoost, AdaBoost	Агрегація прогнозів множини базових класифікаторів	$O(k \cdot n \cdot m \cdot \log(n))$	Середня
Опорні вектори	SVM, ν-SVM, One-Class SVM	Пошук оптимальної гіперплощини розділення через максимізацію маргіну	$O(n^2 \cdot m)$ до $O(n^3 \cdot m)$	Низька
Глибокі мережі	CNN, LSTM, Transformer	Ієрархічне навчання репрезентацій через багаторівневі нелінійні перетворення	$O(p \cdot n \cdot e)$	Дуже низька

Імовірнісні моделі	Naive Bayes, Gaussian Mixture, Hidden Markov Models	Моделювання умовних розподілів класів через байєсівський висновок	O(n·m·c)	Висока

Проблематика незбалансованості класів становить фундаментальний виклик для систем виявлення вторгнень, оскільки нормальний трафік типово домінує над аномальними зразками в реальних мережах [12]. Алгоритми навчання схильні до переважного прогнозування мажоритарного класу, що призводить до неприйнятно низьких показників детектування атак при формально високій загальній точності. Методи ресемплінгу адресують дисбаланс через модифікацію тренувального набору даних шляхом оверсемплінгу міноритарних класів або андерсемплінгу мажоритарного класу, проте можуть призводити до перенавчання або втрати важливої інформації відповідно [39].

Синтетична генерація зразків використовує алгоритми на кшталт SMOTE для створення штучних екземплярів міноритарних класів через інтерполяцію між існуючими сусідніми зразками в просторі ознак, що збільшує різноманітність тренувальних даних без простого дублювання. Чутливі до вартості методи навчання інтегрують асиметричні штрафи за помилки класифікації різних типів безпосередньо в функцію втрат, змушуючи модель приділяти більшу увагу коректній класифікації рідкісних класів атак [47]. Ансамблеві підходи балансування комбінують множину класифікаторів, навчених на різних підвибірках збалансованих даних, агрегуючи їх прогнози для формування фінального рішення з покращеними характеристиками детектування міноритарних класів.

Бенчмаркові набори даних відіграють критичну роль у валідації та порівнянні систем виявлення вторгнень, забезпечуючи стандартизовані умови для оцінки продуктивності різноманітних підходів [34]. Набір даних NSL-KDD, попри численні критичні зауваження щодо застарілості патернів

трафіку, залишається широко використовуваним для попередньої валідації алгоритмів завдяки зручному розміру та відсутності дублікатів записів, що характеризували його попередника KDD Cup 99. UNSW-NB15 містить сучасні варіанти атак та реалістичний фоновий трафік, згенерований через емуляцію різноманітних мережевих додатків, проте обмежений обсяг даних може бути недостатнім для навчання складних моделей глибокого навчання [40].

Датасет CICIDS2017 відзначається реалістичністю симульованого мережевого середовища та різноманітністю категорій атак, включаючи сучасні сценарії ботнетів та експлуатації вразливостей веб-додатків [34]. Edge-IIoTSet спеціально розроблений для промислового Інтернету речей, включаючи атаки специфічні для протоколів MQTT та Modbus, що використовуються в системах промислової автоматизації [15].

Критичним обмеженням синтетичних датасетів залишається неможливість повної емуляції складності та різноманітності реального мережевого трафіку, включаючи тонкі поведінкові нюанси легітимних додатків та еволюціонуючі тактики атакуючих, що може призводити до завищених оцінок ефективності при лабораторному тестуванні порівняно з реальним розгортанням [45]. Інтеграція систем виявлення вторгнень з архітектурами туманних обчислень адресує специфічні виклики розподілених IoT-екосистем через розміщення аналітичних компонентів на проміжному рівні між крайовими пристроями та хмарною інфраструктурою [20]. Розподілене детектування на рівні туманних вузлів знижує латентність реагування на загрози та зменшує обсяг даних, що передаються до центральних систем, оптимізуючи використання пропускної здатності каналів зв'язку. Федеративне навчання дозволяє тренувати глобальні моделі детектування без централізації сирих даних трафіку, агрегуючи локально обчислені оновлення параметрів моделей від розподілених вузлів при збереженні конфіденційності локальних даних [8]. Блокчейн-інтегровані системи виявлення забезпечують незмінність журналів детектованих інцидентів та розподілений консенсус щодо репутації вузлів мережі, що

ускладнює можливості атакуючих щодо маніпуляції історичними даними про безпекові події [23].

Таблиця 1.3 – Порівняння бенчмаркових наборів даних для систем виявлення вторгнень

Набір даних	**Рік публікації**	**Обсяг записів**	**Категорії атак**	**Специфічні особливості**	**Основні обмеження**
KDD Cup 99	1999	4,9 млн	4 основні типи	Перший великомасштабний публічний датасет	Значна кількість дублікатів, застарілі патерни трафіку
NSL-KDD	2009	148 тис.	4 основні типи	Усунення дублікатів, балансування складності	Застарілість мережевих протоколів, відсутність сучасних атак
UNSW-NB15	2015	2,5 млн	9 категорій	Реалістичний фоновий трафік, сучасні атаки	Обмежений обсяг для глибокого навчання, недостатня різноманітність
CICIDS2017	2017	2,8 млн	14 сценаріїв	Повний захват пакетів, детальна анотація потоків	Контрольоване середовище, відсутність реального фонового шуму
Edge-IIoTSet	2022	2,2 млн	14 типів	IoT/IIoT протоколи, федеративне навчання	Специфічність для промислових додатків, складність генералізації

Оцінка ефективності систем виявлення вторгнень вимагає комплексного аналізу множини метрик, що відображають різні аспекти продуктивності детектування [63]. Точність класифікації надає загальну оцінку коректності прогнозів, проте може бути оманливою в умовах незбалансованих датасетів, де висока точність досягається простим прогнозуванням мажоритарного класу. Повнота детектування кількісно характеризує здатність системи ідентифікувати всі реальні інциденти вторгнень, тоді як специфічність

відображає вміння уникати хибнопозитивних спрацювань на легітимному трафіку. F-міра гармонійно балансує точність та повноту через їх гармонічне середнє, надаючи інтегровану оцінку якості детектування для окремих класів атак.

Операційні характеристики приймача візуалізують компроміс між часткою істинно позитивних детектувань та частотою хибних тривог при варіації порогу класифікації, де площа під кривою кількісно узагальнює здатність класифікатора розрізняти класи незалежно від конкретного обраного порогу [29]. Метрики затримки детектування критично важливі для систем реального часу, оскільки навіть високоточна модель втрачає практичну цінність, якщо час обробки призводить до неприйнятних затримок у реагуванні на інциденти. Обчислювальна ефективність включає аналіз споживання пам'яті, процесорного часу та енергетичних ресурсів, що особливо критично для розгортання на ресурсообмежених IoT-пристроях з автономним живленням [54].

Перспективні напрямки досліджень орієнтуються на розробку адаптивних систем детектування, здатних автономно еволюціонувати у відповідь на змінюваний ландшафт загроз без необхідності ручного перенавчання моделей [56]. Метанавчання досліджує алгоритми швидкої адаптації моделей до нових типів атак через навчання узагальнених стратегій навчання на множині пов'язаних задач детектування. Пояснювальний штучний інтелект адресує критичну проблему непрозорості рішень складних моделей глибокого навчання через розробку методів інтерпретації та візуалізації внутрішніх репрезентацій нейронних мереж, що підвищує довіру операторів безпеки та полегшує налагодження систем [24]. Інтеграція контекстної інформації з різнорідних джерел, включаючи журнали системних подій, дані моніторингу продуктивності та розвідувальні відомості про загрози, формує більш цілісне розуміння стану безпеки інфраструктури та дозволяє виявляти складні багатовекторні атаки, розподілені в часі та просторі мережевої топології.

1.2 Аналіз методів машинного навчання для виявлення мережевих атак

Трансформація парадигми виявлення кіберзагроз відбувається через масштабне впровадження алгоритмів машинного навчання, здатних автоматично екстрагувати складні патерни з величезних обсягів мережевого трафіку без необхідності експертного кодування правил детектування. Методологія supervised learning базується на навчанні моделей на анотованих наборах даних, де кожен зразок трафіку супроводжується міткою, що вказує на його належність до легітимної активності або конкретного типу атаки [2]. Алгоритми класифікації навчаються відображенню між простором ознак мережевих потоків та дискретним набором класів загроз через оптимізацію параметрів моделі відносно функції втрат, що кількісно оцінює розбіжність між прогнозами та справжніми мітками на тренувальних даних. Якість узагальнення навчених моделей критично залежить від репрезентативності тренувального набору даних, що має охоплювати достатнє різноманіття варіацій як нормального трафіку, так і всіх релевантних категорій атак для забезпечення адекватної продуктивності на невидимих раніше зразках в умовах реального розгортання [40].

Дерева рішень формують інтуїтивно зрозумілі ієрархічні структури для класифікації через послідовне розбиття простору ознак на основі порогових умов, обраних для максимізації гомогенності результуючих підмножин відносно цільової змінної. Алгоритм C4.5 використовує нормалізований інформаційний приріст як критерій вибору атрибутів розбиття, що дозволяє ефективно обробляти як категоріальні, так і неперервні ознаки при побудові дерев оптимальної структури [41]. CART реалізує бінарне рекурсивне партиціювання через мінімізацію індексу Джині або середньоквадратичної помилки, генеруючи компактні дерева з високою інтерпретованістю внутрішньої логіки прийняття рішень, що особливо цінно для розуміння ключових індикаторів різних типів атак операторами безпеки.

Проблематика перенавчання глибоких дерев рішень вимагає застосування методів регуляризації через обмеження максимальної глибини, встановлення мінімальної кількості зразків для розбиття вузлів або пост-прунінг видалення гілок, що не покращують валідаційну продуктивність. Алгоритми індукції правил екстрагують набори логічних умов безпосередньо з навчених дерев або через альтернативні процедури покриття, формуючи експліцитні правила виду «якщо-то», що можуть безпосередньо інтегруватися в експертні системи безпеки або використовуватися для верифікації рішень більш складних непрозорих моделей [29].

Методи опорних векторів конструюють оптимальні гіперплощини розділення класів через максимізацію маргіну між найближчими представниками різних класів, що теоретично забезпечує покращену здатність узагальнення на невидимих даних згідно з принципами статистичної теорії навчання. Kernel trick дозволяє ефективно виконувати нелінійні класифікації через неявне відображення вихідних ознак в високорозмірні простори, де класи стають лінійно сепарабельними, використовуючи різноманітні функції ядра на кшталт радіальної базисної функції, поліноміальних або сигмоїдних ядер [47]. Параметр регуляризації C контролює компроміс між максимізацією маргіну та мінімізацією помилок класифікації на тренувальних даних, вимагаючи ретельного налаштування через процедури крос-валідації для досягнення оптимального балансу між точністю на навчальній вибірці та здатністю узагальнення.

One-Class SVM спеціалізується на задачах виявлення аномалій через навчання моделі виключно на зразках нормального класу, конструюючи гіперсферу або гіперплощину, що охоплює область нормальних даних з мінімальним об'ємом, класифікуючи зразки поза межами як аномалії [54]. Nu-SVM використовує альтернативну параметризацію через параметр ν, що безпосередньо контролює верхню межу на частку помилок навчання та нижню межу на частку опорних векторів, забезпечуючи більш інтуїтивне налаштування порівняно з традиційним параметром C. Обчислювальна

складність навчання SVM масштабується квадратично або кубічно відносно розміру тренувального набору, що становить значне обмеження для застосування до масштабних датасетів мережевого трафіку без використання апроксимаційних методів або декомпозиційних алгоритмів оптимізації.

Наївний байєсівський класифікатор базується на застосуванні теореми Байєса з припущенням умовної незалежності ознак при заданому класі, що драматично спрощує обчислення апостеріорних імовірностей класів попри нереалістичність припущення незалежності для більшості реальних застосувань. Гаусівський варіант моделює умовні розподіли неперервних ознак через нормальні розподіли, параметри яких оцінюються з тренувальних даних методом максимальної правдоподібності, тоді як мультиноміальна версія призначена для дискретних ознак або текстових даних [41]. Попри наївність фундаментального припущення, алгоритм демонструє дивовижно конкурентну продуктивність у багатьох практичних сценаріях завдяки компенсаційним ефектам, коли систематичні помилки в оцінках імовірностей взаємно нівелюються при обчисленні відношень правдоподібності для різних класів.

Таблиця 1.4 – Характеристики базових алгоритмів класифікації для детектування вторгнень

Алгоритм	**Математична основа**	**Час навчання**	**Час інференсу**	**Чутливість до масштабування**	**Обробка відсутніх значень**	**Типова точність на CICIDS2017**
Decision Tree	Ентропійні міри, індекс Джині	$O(n \cdot m \cdot \log n)$	$O(\log d)$	Низька	Нативна підтримка	92-95%
Random Forest	Бегінг + випадковий вибір ознак	$O(k \cdot n \cdot m \cdot \log n)$	$O(k \cdot \log d)$	Низька	Нативна підтримка	96-98%
SVM	Квадратичне програмування	$O(n^2 \cdot m)$ - $O(n^3 \cdot m)$	$O(s \cdot m)$	Висока	Потребує імпутації	94-96%

Naive Bayes	Теорема Байєса	O(n·m·c)	O(m·c)	Середня	Нативна підтримка	88-92%
k-NN	Метрики відстані	O(1)	O(n·m)	Дуже висока	Потребує імпутації	93-95%
XGBoost	Градієнтний бустинг	O(k·n·m·log n)	O(k·d)	Середня	Вбудована обробка	97-99%

Ансамблеві методи агрегують прогнози множини базових моделей для досягнення вищої точності та робастності порівняно з індивідуальними класифікаторами через експлуатацію принципу «мудрості натовпу». Бегінг генерує різноманітні тренувальні підмножини через випадкову вибірку з поверненням з оригінального датасету, навчаючи на кожній підмножині окремий базовий класифікатор та агрегуючи їх прогнози через голосування більшості для класифікації або усереднення для регресії [14]. Випадковийліс розширює бегінг додатковою рандомізацією через випадковий вибір підмножини ознак-кандидатів на кожному кроці розбиття при побудові дерев, що додатково декорелює помилки базових моделей та покращує здатність узагальнення ансамблю при збереженні високої швидкості навчання та інференсу завдяки можливості паралелізації. Важливість ознак обчислюється через агрегацію зменшень критерію розбиття по всіх деревах ансамблю або через вимірювання деградації продуктивності при пермутації окремих ознак, надаючи цінну інформацію про релевантність різних характеристик мережевого трафіку для задачі детектування атак [7].

Бустинг послідовно будує ансамбль слабких класифікаторів, де кожна наступна модель фокусується на коректній класифікації зразків, що були неправильно класифіковані попередніми моделями через адаптивне перезважування тренувальних екземплярів. AdaBoost експоненційно збільшує ваги некоректно класифікованих зразків після навчання кожного базового класифікатора, змушуючи наступні моделі приділяти більшу увагу складним прикладам, та комбінує базові моделі через зважене голосування з вагами,

пропорційними до логарифму відношення точності до помилки кожного класифікатора [63]. Gradient Boosting узагальнює підхід через оптимізацію довільних диференційованих функцій втрат, де кожне нове дерево навчається апроксимувати від'ємний градієнт функції втрат відносно поточних прогнозів ансамблю, ефективно виконуючи градієнтний спуск у функціональному просторі. XGBoost реалізує високооптимізовану версію градієнтного бустингу з численними алгоритмічними та інженерними покращеннями. Регуляризація включає L1 та L2 штрафи на ваги листів дерев для запобігання перенавчанню, тоді як column subsampling запозичує ідею випадкового вибору ознак з випадкових лісів для додаткової декореляції базових моделей [14].

Апроксимаційний алгоритм побудови дерев використовує гістограмний метод біннінгу неперервних ознак для драматичного прискорення пошуку оптимальних точок розбиття, а паралелізація на рівні ознак експлуатує багатоядерні процесори для ефективного масштабування на великих датасетах. Вбудована обробка розріджених даних та відсутніх значень дозволяє ефективно працювати з реальними мережевими даними, що часто містять неповну інформацію через технічні обмеження збору або природу протоколів.

LightGBM впроваджує альтернативну стратегію росту дерев через leaf-wise експансію замість традиційного level-wise підходу, вибираючи для розбиття листок з максимальним зменшенням втрат незалежно від рівня глибини, що дозволяє будувати більш асиметричні дерева з потенційно кращою апроксимацією складних функцій при фіксованій кількості листів [37]. Gradient-based One-Side Sampling зберігає всі зразки з великими градієнтами та випадково семплює зразки з малими градієнтами, зменшуючи розмір тренувальних даних без значної втрати інформації, оскільки зразки з малими градієнтами вже добре апроксимуються поточною моделлю. Exclusive Feature Bundling об'єднує взаємовиключні розріджені ознаки в єдині щільні бандли, драматично зменшуючи розмірність простору ознак для датасетів з

великою кількістю категоріальних або бінарних атрибутів, що типово для закодованих протокольних полів мережевого трафіку.

Стекінг конструює мета-модель, що навчається оптимально комбінувати прогнози гетерогенного набору базових класифікаторів для максимізації фінальної продуктивності через експлуатацію комплементарних сильних сторін різних алгоритмів. Базовий рівень включає різноманітні моделі на кшталт дерев рішень, SVM, нейронних мереж та наївного Байєса, навчених на оригінальних ознаках трафіку, тоді як мета-класифікатор другого рівня навчається на out-of-fold прогнозах базових моделей для уникнення витоку інформації та перенавчання [42]. Блендінг використовує спрощений варіант через розбиття тренувальних даних на дві підмножини, де базові моделі навчаються на першій частині та генерують прогнози на другій для навчання мета-моделі, жертвуючи частиною тренувальних даних на користь простішої реалізації порівняно з повною крос-валідаційною процедурою стекінгу.

Таблиця 1.5 – Ансамблеві архітектури та їх застосування для виявлення мережевих атак

Метод ансамблювання	**Стратегія диверсифікації**	**Механізм агрегації**	**Переваги для IDS**	**Типові обмеження**	**Рекомендовані сценарії**
Bagging	Бутстреп вибірка даних	Голосування/усереднення	Зниження дисперсії, робастність до шуму	Обмежене покращення для стабільних моделей	Високодисперсійні дані, реал-тайм системи
Random Forest	Бутстреп + випадковий вибір ознак	Голосування більшості	Балансує bias-variance, швидкий інференс	Потребує багато пам'яті для великих ансамблів	Базові IDS, інтерпретовані рішення
AdaBoost	Адаптивне перезважування зразків	Зважене голосування	Фокус на складних зразках, висока точність	Чутливий до шумових міток та викидів	Чисті датасети, бінарна класифікація
Gradient Boosting	Апроксимація градієнтів	Адитивна комбінація	Гнучкість функцій	Повільне навчання, схильність	Змагання з машинного навчання,

	функції втрат		втрат, високі метрики	до перенавчання	офлайн аналіз
XGBoost/LightGBM	Регуляризація + сучасні евристики	Адитивна з ваговими коефіцієнтами	Екстремальна продуктивність, вбудована регуляризація	Складність налаштування гіперпараметрів	Продакшн системи, великі датасети
Stacking	Гетерогенні базові алгоритми	Мета-модель навчання	Експлуатація комплементарності алгоритмів	Висока обчислювальна складність	Критичні системи, максимальна точність

Методи k-найближчих сусідів виконують класифікацію через аналіз міток k найближчих тренувальних зразків до тестового екземпляру в просторі ознак, де близькість визначається через метрики відстані на кшталт евклідової, манхеттенської або косинусної подібності. Алгоритм не потребує експліцитної фази навчання окрім індексування тренувальних даних, проте вся обчислювальна складність зміщується в фазу інференсу, що вимагає обчислення відстаней до всіх тренувальних зразків для класифікації кожного тестового екземпляру [54]. Прискорені варіанти використовують просторові структури даних на кшталт KD-дерев, Ball-дерев або LSH для сублінійного часу пошуку найближчих сусідів, хоча ефективність драматично деградує в високорозмірних просторах через прокляття розмірності, де концепція локальності втрачає значення при великій кількості ознак.

Вибір оптимального значення k критично впливає на продуктивність через компроміс між чутливістю до локальних патернів при малих k та стабільністю через усереднення при великих k. Зважені варіанти присвоюють більшу вагу внескам ближчих сусідів через функції, обернено пропорційні до відстаней, покращуючи локальну адаптивність класифікаційних рішень порівняно з простим голосуванням більшості [41]. Редукція розмірності через PCA або автоенкодери перед застосуванням k-NN часто покращує продуктивність через видалення шумових ознак та подолання прокляття

розмірності, хоча потребує додаткового етапу препроцесингу та ускладнює інтерпретацію результатів через трансформацію оригінальних ознак.

Глибоке навчання революціонізувало парадигму feature engineering через автоматичне вивчення ієрархічних репрезентацій безпосередньо з сирих або мінімально препроцесених даних без необхідності ручного дизайну та вибору ознак доменними експертами. Багатошарові перцептрони складаються з послідовності повнозв'язних шарів нейронів з нелінійними активаційними функціями, що дозволяє апроксимувати довільно складні нелінійні відображення між входами та виходами згідно з теоремою універсальної апроксимації [49]. Алгоритм зворотного розповсюдження обчислює градієнти функції втрат відносно ваг мережі через рекурсивне застосування правила ланцюга диференціювання, що дозволяє ефективно навчати глибокі архітектури через варіанти стохастичного градієнтного спуску попри нелінійність та композиційну природу моделі. Проблеми зникаючих та вибухових градієнтів у глибоких архітектурах адресуються через спеціалізовані ініціалізаційні схеми на кшталт Xavier або He, нормалізаційні шари як batch normalization або layer normalization, та архітектурні інновації на кшталт залишкових з'єднань. Dropout виконує стохастичну регуляризацію через випадкове вимикання нейронів під час навчання з певною імовірністю, що запобігає коадаптації детекторів ознак та покращує здатність узагальнення через неявне навчання ансамблю експоненційно великої кількості підмереж [55]. Early stopping моніторить продуктивність на валідаційній множині та припиняє навчання при детектуванні погіршення, запобігаючи перенавчанню через обмеження ефективної ємності моделі попри номінально велику кількість параметрів.

Згорткові нейронні мережі експлуатують просторову структуру даних через локальні рецептивні поля, спільне використання ваг та пулінг для ефективного вивчення трансляційно-інваріантних ієрархічних ознак. Конволюційні шари застосовують набори навчуваних фільтрів до локальних регіонів входу, детектуючи специфічні патерни на різних просторових

позиціях через операцію згортки, що драматично зменшує кількість параметрів порівняно з повнозв'язними шарами при обробці високорозмірних входів [51]. Пулінг виконує субдискретизацію через агрегацію значень у локальних регіонах через максимум, середнє або інші функції, забезпечуючи додаткову інваріантність до невеликих трансляцій та зменшуючи просторову розмірність для наступних шарів. Застосування CNN до мережевого трафіку вимагає структурування одновимірних потоків пакетів або багатовимірних векторів ознак у псевдозображення, де локальні кореляції між сусідніми елементами можуть експлуатуватися конволюційними фільтрами для детектування патернів атак [17].

Рекурентні нейронні мережі спеціалізуються на обробці послідовностей через внутрішні стани, що акумулюють інформацію з попередніх кроків часу, дозволяючи моделювати темпоральні залежності довільної довжини. Ванільні RNN страждають від проблем зникаючих градієнтів при навчанні на довгих послідовностях, обмежуючи здатність захоплювати довгострокові залежності попри теоретичну спроможність [27]. Довга короткочасна пам'ять вирішує проблему через спеціалізовану архітектуру комірок з гейтовими механізмами, що контролюють потік інформації через стани пам'яті, дозволяючи селективно зберігати релевантну інформацію на довгих часових горизонтах та ігнорувати нерелевантні вхідні сигнали [13].

Таблиця 1.6 – Архітектури глибокого навчання для детектування мережевих вторгнень

Архітектура	**Структурні компоненти**	**Механізм обробки**	**Спеціалізація**	**Обчислювальні вимоги**	**Типова продуктивність**	**Характерні виклики**
MLP	Повнозв'язні шари, ReLU/Sigmoid активації	Послідовна трансформація через матричні множення	Табличні дані, векторні ознаки	Середні (залежить від розміру шарів)	F1: 0.95-0.97	Вимагає ручного feature engineering

CNN	Конволюційні фільтри, пулінг, batch norm	Локальні рецептивні поля, спільні ваги	Просторові патерни, зображення трафіку	Високі (GPU-оптимізовані)	F1: 0.96-0.98	Потребує структурування даних
LSTM	Гейтові механізми, стани пам'яті	Рекурентна обробка послідовностей	Темпоральні залежності, потоки пакетів	Дуже високі (послідовні обчислення)	F1: 0.94-0.97	Повільне навчання, складність паралелізації
GRU	Спрощені гейти (reset, update)	Рекурентна з меншою складністю	Темпоральні патерни, швидший LSTM	Високі (менше параметрів ніж LSTM)	F1: 0.94-0.96	Менша виразність для дуже довгих послідовностей
CNN-LSTM	Конволюція + рекурентні шари	Ієрархічна: просторова → темпоральна	Просторо-часові патерни атак	Дуже високі (комбінована складність)	F1: 0.97-0.99	Складність архітектури, тривале навчання
Autoencoder	Енкодер-декодер, bottleneck	Стиснення → реконструкція	Виявлення аномалій, зменшення розмірності	Середні-високі	AUC: 0.93-0.96	Чутливий до вибору розміру bottleneck
Transformer	Self-attention, позиційне кодування	Паралельна обробка через attention	Довгі залежності, паралелізм	Екстремально високі (квадратична складність)	F1: 0.96-0.98	Потребує великих датасетів, memory-intensive

Гейтова рекурентна одиниця спрощує архітектуру LSTM через об'єднання гейтів забування та входу в єдиний гейт оновлення, зменшуючи кількість параметрів та обчислювальну складність при збереженні більшості виразних можливостей для моделювання послідовностей [50]. Reset gate контролює скільки попередньої інформації зі стану пам'яті має бути використано при обчисленні нового стану-кандидата, тоді як update gate визначає баланс між збереженням попереднього стану та інтеграцією нового стану-кандидата. Емпіричні дослідження демонструють, що GRU часто

досягає порівнянної продуктивності з LSTM при швидшому навчанні та інференсі завдяки меншій кількості параметрів, хоча LSTM може мати перевагу на задачах з дуже довгостроковими залежностями, де додаткова виразність окремих гейтів забування та входу виявляється критичною. Гібридні архітектури CNN-LSTM комбінують згорткові та рекурентні компоненти для одночасної експлуатації просторових та темпоральних структур у даних мережевого трафіку. Конволюційні шари екстрагують локальні просторові ознаки з окремих пакетів або вікон трафіку, тоді як LSTM шари обробляють послідовності екстрагованих ознакових векторів для моделювання темпоральної еволюції мережевої активності та детектування багатоетапних атак з розтягнутою реалізацією [17].

Альтернативна архітектура ConvLSTM інтегрує конволюційні операції безпосередньо в рекурентні гейтові механізми, замінюючи матричні множення на згортки для одночасної обробки просторо-часових даних в єдиній структурі без експліцитного розділення на етапи екстракції ознак та темпорального моделювання [62].

Двонаправлені рекурентні мережі обробляють послідовності одночасно в прямому та зворотному напрямках через два набори рекурентних шарів, агрегуючи виходи для формування контекстно збагачених репрезентацій, що інтегрують інформацію як з минулих, так і майбутніх елементів послідовності. Така архітектура особливо ефективна для офлайн аналізу повних трас мережевих потоків, де доступ до всієї послідовності дозволяє використовувати майбутній контекст для покращеного розуміння природи поточних подій, хоча унеможливлює застосування в режимі реального часу через необхідність доступу до повної послідовності перед початком обробки [25].

Механізми уваги дозволяють моделям динамічно фокусуватися на релевантних частинах вхідної послідовності або просторових регіонах при формуванні виходів, навчаючи ваги важливості різних компонентів входу через диференційовані функції узгодження. Self-attention обчислює узгодження між всіма парами позицій у послідовності, дозволяючи кожному

елементу безпосередньо агрегувати інформацію від всіх інших елементів незалежно від їх відстані, ефективно подоляючи обмеження рекурентних архітектур щодо послідовної обробки та складності захоплення довгострокових залежностей [68]. Трансформери повністю базуються на механізмах self-attention без використання рекуренції або згортки, досягаючи state-of-the-art результатів на численних задачах обробки послідовностей завдяки паралелізму обчислень та здатності моделювати глобальні залежності, хоча квадратична складність відносно довжини послідовності обмежує застосування до дуже довгих трас трафіку без спеціалізованих оптимізацій на кшталт sparse attention або linformer.

Автоенкодери навчаються стискати вхідні дані в низькорозмірні латентні репрезентації через енкодер та відновлювати оригінальні входи з латентних кодів через декодер, мінімізуючи помилку реконструкції між входами та виходами. Для виявлення аномалій автоенкодери навчаються виключно на нормальному трафіку, після чого помилка реконструкції використовується як міра аномальності під припущенням, що мережа не зможе ефективно реконструювати атипові зразки, що не були представлені в тренувальних даних [28]. Варіаційні автоенкодери розширюють базову архітектуру через введення імовірнісних латентних змінних, навчаючи енкодер виводити параметри розподілу латентного коду замість детермінованого вектору, та оптимізуючи варіаційну нижню межу на логарифм правдоподібності даних через комбінацію терму реконструкції та KL-дивергенції між апроксимованим та апріорним розподілами.

Глибокі автоенкодери з множиною прихованих шарів навчаються ієрархічним репрезентаціям різних рівнів абстракції, де початкові шари захоплюють низькорівневі характеристики трафіку, а глибші шари формують високорівневі семантичні репрезентації складних патернів мережевої активності. Денойзинг автоенкодери навчаються реконструювати чисті входи з корумпованих версій, додаючи стохастичний шум або випадково маскуючи частини входів під час навчання, що примушує мережу вивчати більш робастні

репрезентації, здатні витягувати суттєві характеристики попри присутність шуму або неповноту даних, що типово для реальних мережевих середовищ з втратами пакетів та неідеальними умовами спостереження.

Синтетична генерація атак через GAN адресує проблему обмеженої доступності різноманітних зразків рідкісних або нових типів вторгнень для навчання дискримінативних моделей. Wasserstein GAN з gradient penalty покращує стабільність навчання через використання Earth Mover's distance замість Jensen-Shannon дивергенції та додавання штрафу на норму градієнта дискримінатора для забезпечення виконання умови Ліпшиця, що критично для конвергенції тренувального процесу та якості згенерованих зразків [59]. Progressive GAN поступово нарощує складність генератора та дискримінатора через додавання нових шарів під час навчання, починаючи з генерації низькорозмірних простих представлень та поступово переходячи до високорозмірних детальних зразків, що покращує стабільність та дозволяє генерувати більш реалістичний та різноманітний синтетичний трафік для розширення тренувальних датасетів.

1.3 Дослідження гібридних підходів та механізмів динамічного налаштування порогових значень

Концептуальна еволюція систем виявлення вторгнень демонструє поступовий перехід від монолітних однорідних архітектур до складних гібридних конфігурацій, що інтегрують множину гетерогенних компонентів для експлуатації комплементарних переваг різноманітних методологій детектування [19]. Фундаментальна мотивація гібридизації полягає у подоланні інгерентних обмежень окремих підходів через синергетичне поєднання сигнатурних механізмів з високою специфічністю виявлення відомих загроз та аномальних систем з здатністю детектування невідомих раніше атак, створюючи багаторівневі захисні периметри з ортогональними векторами захисту. Архітектурна гетерогенність гібридних систем охоплює як

послідовні каскадні конфігурації, де вихід одного компонента живить наступний етап аналізу, так і паралельні ансамблеві структури з незалежною обробкою та подальшою агрегацією рішень через механізми голосування або зважених консенсусів [42]. Критичним аспектом проектування виявляється оптимальне розподілення обчислювальних ресурсів між компонентами для балансування між глибиною аналізу та латентністю детектування в умовах жорстких часових обмежень систем реального часу [58].

Каскадні архітектури реалізують стратифікований аналіз через послідовність спеціалізованих детекторів з прогресивно зростаючою складністю та специфічністю. Початковий етап фільтрації використовує швидкі евристичні правила або легкі класифікатори для відсіювання очевидно легітимного трафіку з мінімальними обчислювальними витратами, дозволяючи зосередити ресурсоємний глибокий аналіз виключно на підозрілих потоках [10]. Проміжні рівні застосовують машинне навчання середньої складності для попередньої категоризації загроз та пріоритизації найбільш критичних інцидентів, тоді як фінальні етапи залучають складні моделі глибокого навчання або ансамблі для точної класифікації специфічних типів атак з максимальною гранулярністю розпізнавання [38].

Адаптивне маршрутизування трафіку між компонентами каскаду базується на рівнях впевненості попередніх детекторів, де зразки з високою невизначеністю класифікації ескалюються до більш потужних але повільних аналізаторів, оптимізуючи компроміс між швидкістю та точністю через контекстно-залежне розподілення обчислювальних ресурсів відповідно до характеристик конкретного трафіку.

Паралельні ансамблеві конфігурації одночасно обробляють мережевий трафік через множину незалежних детекторів різної природи, агрегуючи їх вердикти через механізми консенсусу для формування фінального рішення [66]. Голосування більшості присвоює однакову вагу всім компонентам, класифікуючи зразок згідно з прогнозом, підтриманим більшістю детекторів, що забезпечує робастність до помилок окремих класифікаторів через

демократичну агрегацію незалежних оцінок. Зважене голосування присвоює диференційовані ваги компонентам пропорційно їх історичній продуктивності або довірі до специфічних типів загроз, дозволяючи більш надійним або спеціалізованим детекторам здійснювати більший вплив на фінальне рішення через експліцитне моделювання відносної компетентності різних компонентів для різних сценаріїв атак [47]. Динамічна адаптація ваг базується на моніторингу продуктивності компонентів у реальному часі, автоматично перерозподіляючи довіру між детекторами у відповідь на зміни характеристик трафіку або еволюцію тактик атакуючих для підтримання оптимальної агрегаційної стратегії в нестаціонарному середовищі.

Інтеграція різнорідних парадигм навчання створює можливості для взаємного посилення через експлуатацію унікальних епістемологічних переваг кожного підходу. Комбінування supervised learning для класифікації відомих категорій атак з unsupervised методами для виявлення аномальних відхилень від нормального профілю дозволяє одночасно досягати високої точності на каталогізованих загрозах та здатності детектування zero-day експлойтів без попередніх зразків [30]. Semi-supervised підходи експлуатують великі обсяги немаркованого мережевого трафіку для покращення узагальнювальної здатності моделей через самонавчання або ко-тренінг, адресуючи критичну проблему дефіциту анотованих даних для рідкісних або еволюціонуючих типів атак [56]. Reinforcement learning інтегрується для оптимізації стратегій реагування та адаптивного налаштування параметрів детектування через взаємодію з середовищем, навчаючи агентів максимізувати довгострокову винагороду від коректних детектувань при мінімізації витрат від хибних спрацювань та пропущених атак через послідовність спроб та помилок з отриманням відкладеної обернення зв'язку про наслідки прийнятих рішень.

Багаторівневий feature engineering комбінує експертні знання для конструювання доменно-специфічних ознак з автоматичним вивченням репрезентацій через глибоке навчання. Статистичні агрегати на кшталт

розподілів розмірів пакетів, частот міжприбуттєвих інтервалів або ентропій значень полів захоплюють стохастичні характеристики мережевих потоків, тоді як темпоральні ознаки відображають еволюцію поведінки з'єднань протягом часу через ковзні вікна агрегації [62]. Протокольні ознаки екстрагують специфічні індикатори з полів заголовків різних рівнів мережевого стеку, включаючи прапорці TCP, типи ICMP-повідомлень або коди HTTP-статусів, що мають різну семантичну значущість для детектування специфічних класів атак [68]. Автоматично вивчені латентні репрезентації з глибоких мереж захоплюють складні нелінійні залежності та взаємодії між ознаками, що важко або неможливо закодувати експліцитно, забезпечуючи комплементарність до ручно сконструйованих атрибутів через різні епістемологічні джерела інформації про структуру даних. Гібридні CNN-LSTM архітектури експлуатують здатність згорткових мереж до екстракції просторових патернів та спеціалізацію рекурентних компонентів на моделюванні темпоральних залежностей для детектування багатоетапних атак з розтягнутою реалізацією [17]. Конволюційні фільтри сканують вікна послідовних пакетів або байтів полезного навантаження для детектування локальних сигнатур атак через трансляційно-інваріантне розпізнавання патернів незалежно від абсолютної позиції в потоці. LSTM компоненти аналізують еволюцію екстрагованих конволюційних ознак протягом тривалих часових інтервалів для ідентифікації підозрілих послідовностей активності, що характеризують складні сценарії компрометації з розвідувальними фазами, початковим проникненням, латеральним переміщенням та фінальною ексфільтрацією даних [50].

Attention механізми додатково фокусують обчислювальні ресурси на найбільш інформативних часових кроках або просторових регіонах, дозволяючи моделі динамічно адаптувати увагу до релевантних аспектів складних багатоваріантних послідовностей мережевої активності для покращеного розпізнавання критичних індикаторів компрометації серед великих обсягів фонового шуму легітимного трафіку [24].

Автоенкодер-базовані гібриди комбінують ненаглядоване навчання компактних репрезентацій з наглядованою класифікацією для одночасного виявлення відомих та невідомих аномалій. Автоенкодер навчається на нормальному трафіку реконструювати вхідні зразки через вузьке горловинне представлення, після чого помилка реконструкції використовується як міра аномальності для детектування відхилень від встановленого профілю нормальності [28].

Латентні репрезентації з bottleneck шару автоенкодера живлять supervised класифікатор для розпізнавання специфічних типів атак, експлуатуючи автоматично вивчені стислі кодування як високоінформативні ознаки для подальшої дискримінації між категоріями загроз [26]. Variational autoencoder забезпечує додаткову регуляризацію латентного простору через накладання імовірнісних обмежень, дозволяючи обчислювати апостеріорні імовірності належності зразків до нормального розподілу для більш принципового статистичного тестування гіпотез щодо аномальності спостережень порівняно з евристичними пороговими механізмами на помилці реконструкції детермінованих автоенкодерів.

Інтеграція GAN для синтетичної генерації трафіку адресує проблему дисбалансу класів через створення реалістичних штучних зразків недопредставлених категорій атак [59]. Conditional GAN навчається генерувати специфічні типи загроз через кондиціювання на мітках класів, дозволяючи цілеспрямовану аугментацію саме тих категорій, що потребують додаткових тренувальних екземплярів для досягнення збалансованого представлення всіх типів атак в датасеті. Adversarial training використовує GAN для синтезу складних прикладів атак, близьких до меж рішення класифікатора, тренуючи більш робастні моделі через експозицію до найбільш challenging зразків під час навчання [69]. Критичним залишається забезпечення реалістичності згенерованих даних через валідацію проти справжнього трафіку та експертну оцінку, оскільки навчання на артефактних синтетичних зразках може призвести до вивчення нерелевантних патернів та

деградації продуктивності на реальних мережевих даних попри покращення метрик на збалансованих синтетичних тестових наборах.

Ансамблювання гетерогенних алгоритмів через stacking інтегрує різноманітні сімейства моделей для експлуатації їх комплементарних епістемологічних переваг [63]. Базовий рівень включає Random Forest для швидкого first-pass скринінгу з високою робастністю до шумових ознак, XGBoost для точної класифікації через градієнтний бустинг, SVM з RBF ядром для нелінійної сепарації у трансформованих просторах високої розмірності, та глибоку нейронну мережу для автоматичного feature learning з сирих атрибутів трафіку.

Мета-класифікатор другого рівня навчається на out-of-fold прогнозах базових моделей оптимально комбінувати їх вердикти, вивчаючи контекстно-залежні правила агрегації що враховують відносну компетентність різних базових моделей для специфічних підкласів атак або характеристик трафіку через аналіз патернів помилок та успіхів кожного компонента на різноманітних сценаріях. Критична важливість крос-валідаційної процедури генерації мета-ознак запобігає витоку інформації з тренувальних даних у валідаційні прогнози, що призвело б до завищених оцінок продуктивності через неявне використання тестової інформації під час навчання мета-рівня.

Механізми динамічного налаштування порогових значень адресують фундаментальну проблему статичних порогів класифікації, що не адаптуються до еволюції характеристик трафіку та зміщень розподілів нормальної активності внаслідок легітимних змін у мережевому середовищі. Adaptive thresholding моніторить статистичні характеристики вихідних розподілів класифікаторів у реальному часі, динамічно коригуючи граничні значення для підтримання цільових показників false positive rate або detection rate в умовах нестаціонарних даних [37]. Контекстно-залежні пороги варіюються в залежності від часу доби, типу мережевого сегменту, критичності захищуваних активів або поточного рівня загрози згідно з розвідувальною інформацією про активні кампанії атакуючих, дозволяючи більш агресивні

налаштування для високоризикових періодів або ресурсів при послабленні обмежень для рутинних сценаріїв з метою зниження операційного навантаження від хибних тривог. Reinforcement learning оптимізує стратегії адаптації порогів через взаємодію з середовищем, навчаючи політики коригування параметрів детектування що максимізують довгострокову винагороду від правильних детектувань при пенальті за хибні спрацювання та пропущені атаки, автоматично балансуючи conflicting objectives через trial-and-error експлорацію простору можливих конфігурацій [56].

Пояснювальні методи візуалізують та інтерпретують рішення складних моделей для покращення довіри операторів та полегшення debugging процесу [24]. LIME апроксимує локальну поведінку моделі навколо специфічного зразка через навчання інтерпретованої surrogate моделі на пертурбованих версіях входу, ідентифікуючи найбільш впливові ознаки для конкретного прогнозу через коефіцієнти лінійної регресії. SHAP обчислює Shapley values з теорії кооперативних ігор для справедливого розподілення внеску кожної ознаки у відхилення прогнозу від базового значення, забезпечуючи теоретично обґрунтоване пояснення через розгляд всіх можливих коаліцій ознак та їх маргінальних внесків у фінальний вердикт моделі. Attention visualization відображає ваги attention механізмів глибоких мереж для ідентифікації часових кроків або просторових регіонів на які модель фокусується при формуванні класифікаційних рішень, надаючи інсайти щодо внутрішньої логіки складних архітектур через візуалізацію розподілів уваги по вхідним послідовностям або зображенням мережевого трафіку.

Інтеграція threat intelligence збагачує локальні дані детектування контекстом про глобальні кампанії атакуючих, індикатори компрометації та тактики-техніки-процедури ворожих акторів з зовнішніх фідів [11]. IP reputation services надають оцінки maliciousness для спостережуваних адрес базуючись на історії їх участі у зловмисній активності згідно з розподіленими honeypot мережами та іншими джерелами телеметрії. Domain reputation аналізує DNS-запити для блокування звернень до відомих command-and-

control серверів або phishing сайтів базуючись на чорних списках та евристичному аналізі характеристик доменних імен. Correlation engines агрегують події з множини джерел включаючи мережеві IDS, host-based сенсори, firewall логи та threat feeds для детектування розподілених багатоетапних атак через temporal та spatial кореляцію індикаторів компрометації що ізольовано виглядають benign проте в агрегації формують підозрілі патерни активності що вказують на координовані кампанії проти інфраструктури.

1.4 Висновки до розділу

1. Проведений аналіз сучасних методів виявлення мережевих атак демонструє еволюцію від традиційних сигнатурних підходів до складних систем на базі машинного навчання та глибоких нейронних мереж. Встановлено, що класичні методи детектування на основі статичних сигнатур забезпечують високу точність розпізнавання відомих загроз з мінімальною кількістю хибних спрацювань, проте виявляються неефективними для ідентифікації нових типів атак через обмеженість баз сигнатур та неможливість адаптації до еволюції тактик зловмисників. Аномальні системи виявлення вторгнень, що базуються на моделюванні нормальної поведінки мережі, демонструють здатність детектування zero-day експлойтів, однак характеризуються підвищеним рівнем хибнопозитивних спрацювань та потребують тривалого періоду навчання для формування адекватних базових профілів активності. Архітектурна диверсифікація систем охоплює мережецентричні конфігурації з централізованим моніторингом трафіку, хостоцентричні рішення для глибокого аналізу поведінки окремих вузлів та розподілені гібридні архітектури, що забезпечують комплексну видимість загроз через корреляцію різнорідних джерел телеметрії безпеки.

2. Систематизація алгоритмів машинного навчання для виявлення мережевих атак виявила критичну роль ансамблевих методів та архітектур

глибокого навчання у досягненні високої продуктивності детектування. Базові алгоритми класифікації, включаючи дерева рішень, методи опорних векторів та наївний байєсівський класифікатор, забезпечують інтерпретовані рішення та прийнятну точність для простих сценаріїв загроз, проте демонструють обмежену здатність моделювання складних нелінійних залежностей у високорозмірних просторах ознак мережевого трафіку. Ансамблеві техніки на кшталт випадкових лісів та градієнтного бустингу досягають покращеної робастності через агрегацію прогнозів множини базових моделей, тоді як XGBoost та LightGBM забезпечують state-of-the-art продуктивність завдяки алгоритмічним оптимізаціям та вбудованим механізмам регуляризації. Глибокі нейронні мережі революціонізували парадигму feature engineering через автоматичне вивчення ієрархічних репрезентацій, де згорткові архітектури ефективно екстрагують просторові патерни, рекурентні мережі моделюють темпоральні залежності в послідовностях пакетів, а гібридні CNN-LSTM конфігурації синергетично комбінують обидві можливості для детектування складних багатоетапних атак.

3. Дослідження гібридних підходів та механізмів динамічного налаштування виявило перспективність інтеграції гетерогенних компонентів для подолання інгерентних обмежень окремих методологій детектування. Каскадні архітектури реалізують стратифікований аналіз через послідовність спеціалізованих детекторів з прогресивно зростаючою складністю, оптимізуючи баланс між швидкістю обробки та глибиною аналізу через селективне застосування ресурсоємних моделей виключно до підозрілого трафіку після початкової фільтрації легких класифікаторів. Паралельні ансамблеві конфігурації експлуатують комплементарність різнорідних алгоритмів через одночасну обробку трафіку множиною незалежних детекторів та інтелектуальну агрегацію їх вердиктів, де динамічне зважування компонентів адаптується до змін продуктивності в нестаціонарному середовищі.

РОЗДІЛ 2. РОЗРОБКА ГІБРИДНОГО МЕТОДУ ВИЯВЛЕННЯ МЕРЕЖЕВИХ АТАК

2.1 Постановка задачі та розробка архітектури гібридної системи виявлення атак

Проблематика виявлення мережевих атак залишається актуальною внаслідок постійного ускладнення методів зловмисників та зростання обсягів трафіку в корпоративних мережах. Традиційні підходи до побудови систем детектування вторгнень демонструють обмежену ефективність при зіткненні з новими типами загроз або варіаціями відомих атак. Класичні методи сигнатурного аналізу виявляються безсилими перед атаками нульового дня, тоді як системи на основі аномалій генерують надмірну кількість хибних спрацювань. Виникає нагальна потреба у створенні комплексного рішення, здатного поєднувати переваги різних підходів машинного навчання з можливістю адаптації до змінюваних умов функціонування мережевої інфраструктури.

Формулювання задачі дослідження передбачає розробку гібридної архітектури виявлення атак, яка інтегрує контрольовані та неконтрольовані алгоритми машинного навчання з механізмом динамічного налаштування порогових значень. Математична постановка завдання включає визначення множини мережевих з'єднань, яка позначається як сукупність елементів від першого до n-го, де кожен елемент x з індексом i представляє вектор ознак, що характеризують параметри з'єднання. Необхідно побудувати класифікаційну функцію, яка відображає простір вхідних даних на множину міток класів, що включає нормальний трафік та атаки. Функція має мінімізувати втрати при максимізації показників точності виявлення та мінімізації помилок першого і другого роду.

Математичний запис задачі класифікації формулюється наступним чином:

$$f: X \rightarrow Y, \text{де } Y = \{normal, attack\}$$

Функція втрат визначається як міра розбіжності між передбаченими та фактичними мітками класів:

$$L(f) = \left(\frac{1}{N}\right) \times \Sigma(i = 1 \text{ до } N) loss(y_i, \hat{y}_i)$$

Додатково система повинна забезпечувати адаптивність через механізм θ(t), що динамічно коригує порогові значення залежно від поточних характеристик мережевого трафіку та змінюється з плином часу для оптимізації співвідношення між чутливістю та специфічністю виявлення.

Архітектура запропонованої гібридної системи складається з декількох взаємопов'язих модулів, кожен з яких виконує специфічні функції в процесі аналізу мережевого трафіку. Модуль попередньої обробки здійснює захоплення пакетів, їх розбір та екстракцію релевантних ознак з різних рівнів мережевого стеку. Застосовуються техніки нормалізації для приведення числових значень до єдиного діапазону та методи кодування категоріальних змінних. Виконується фільтрація шумових даних та обробка пропущених значень за допомогою статистичних методів імпутації, що забезпечує коректність подальшого аналізу та уникнення спотворень у процесі навчання моделей.

Нормалізація ознак здійснюється за формулою мін-макс перетворення:

$$x'_i = \frac{x_i - x_{\min}}{x_{\max} - x_{\min}}$$

Перший рівень класифікації реалізовано на основі ансамблю дерев рішень з використанням алгоритму випадкового лісу. Вибір обумовлено здатністю методу ефективно працювати з багатовимірними даними та забезпечувати стійкість до перенавчання завдяки процедурі бутстрепу. Алгоритм будує множину некорельованих дерев рішень, кожне з яких навчається на випадковій підвибірці вихідного набору даних із заміною. Фінальне рішення приймається шляхом голосування більшістю серед усіх дерев ансамблю, що

дозволяє згладити індивідуальні помилки окремих класифікаторів та підвищити загальну робастність системи.

Передбачення випадкового лісу визначається як:

$$\hat{y} = mode\{h^1(x), h^2(x), \dots, h_t(x)\}$$

де $h_j(x)$ позначає передбачення j-го дерева з ансамблю загальною кількістю T дерев.

Другий рівень архітектури включає нейромережевий компонент на основі багатошарового персептрону, призначений для виявлення складних нелінійних залежностей між ознаками трафіку. Мережа складається з вхідного шару розмірністю, що відповідає кількості екстрагованих ознак, двох прихованих шарів з нелінійними функціями активації та вихідного шару з функцією softmax для отримання ймовірнісних оцінок приналежності до класів. Навчання мережі відбувається методом зворотного поширення помилки з використанням оптимізатора Adam, який адаптивно налаштовує швидкість навчання для кожного параметра.

Вихід нейронної мережі обчислюється послідовним застосуванням афінних перетворень та функцій активації:

$$a^{(l)} = \sigma\left(W^{(l)} \times a^{(l-1)} + b^{(l)}\right)$$

де $W^{(l)}$ та $b^{(l)}$ представляють матрицю ваг та вектор зміщення для l-го шару відповідно, а σ позначає функцію активації.

Модуль детекції аномалій базується на алгоритмі ізоляційного лісу, який виявляє викиди в багатовимірному просторі ознак без необхідності попереднього навчання на міченому наборі даних. Принцип роботи полягає у рекурсивному розбитті простору ознак випадковими гіперплощинами та обчисленні середньої глибини ізоляції для кожного спостереження. Аномальні зразки характеризуються меншою середньою глибиною ізоляції порівняно з нормальними спостереженнями, оскільки їх легше відокремити від основної маси даних. Метод демонструє високу обчислювальну ефективність завдяки субдискретизації та низькій складності побудови дерев ізоляції.

Оцінка аномальності спостереження визначається як:

$$s(x,n) = 2^{-\frac{E(h(x))}{c(n)}}$$

де E(h(x)) представляє математичне сподівання глибини ізоляції для точки x, а c(n) є нормалізуючою константою для вибірки розміру n.

Гібридний механізм прийняття рішень інтегрує результати трьох незалежних класифікаторів через модуль метаналізу, який зважує вихідні оцінки кожного компонента відповідно до їх продуктивності на валідаційній вибірці. Застосовується адаптивна схема зважування, де коефіцієнти динамічно коригуються на основі статистики останніх детекцій. Для кожного класифікатора розраховується ваговий коефіцієнт пропорційний до значення F1-міри, обчисленої на ковзному вікні останніх класифікацій. Фінальна оцінка атаки формується як зважена сума ймовірностей від індивідуальних моделей.

Загальна оцінка атаки обчислюється як лінійна комбінація:

$$P(attack|x) = w^1 \times P^1(attack|x) + w^2 \times P^2(attack|x) + w^3 \times P^3(attack|x)$$

де w_j представляє ваговий коефіцієнт j-го класифікатора, а $P_j(attack|x)$ є ймовірністю атаки згідно з j-ю моделлю.

Механізм динамічного налаштування порогових значень реалізовано на основі аналізу розподілу оцінок класифікаторів на поточному потоці трафіку. Поріг детекції автоматично коригується для підтримання заданого балансу між показниками true positive rate та false positive rate. Алгоритм аналізує гістограму розподілу оцінок аномальності за останній часовий інтервал та визначає оптимальну точку розділення класів методом максимізації критерію Юдена. Адаптація порогу відбувається поступово через експоненціальне згладжування для уникнення різких стрибків, які можуть спричинити нестабільність детекції.

Динамічний поріг оновлюється згідно з правилом:

$$\theta(t) = \alpha \times \theta_{\text{opt}}(t) + (1 - \alpha) \times \theta(t-1)$$

де $\theta_{opt}(t)$ позначає оптимальний поріг на поточному кроці, а α є параметром згладжування зі значеннями від нуля до одиниці.

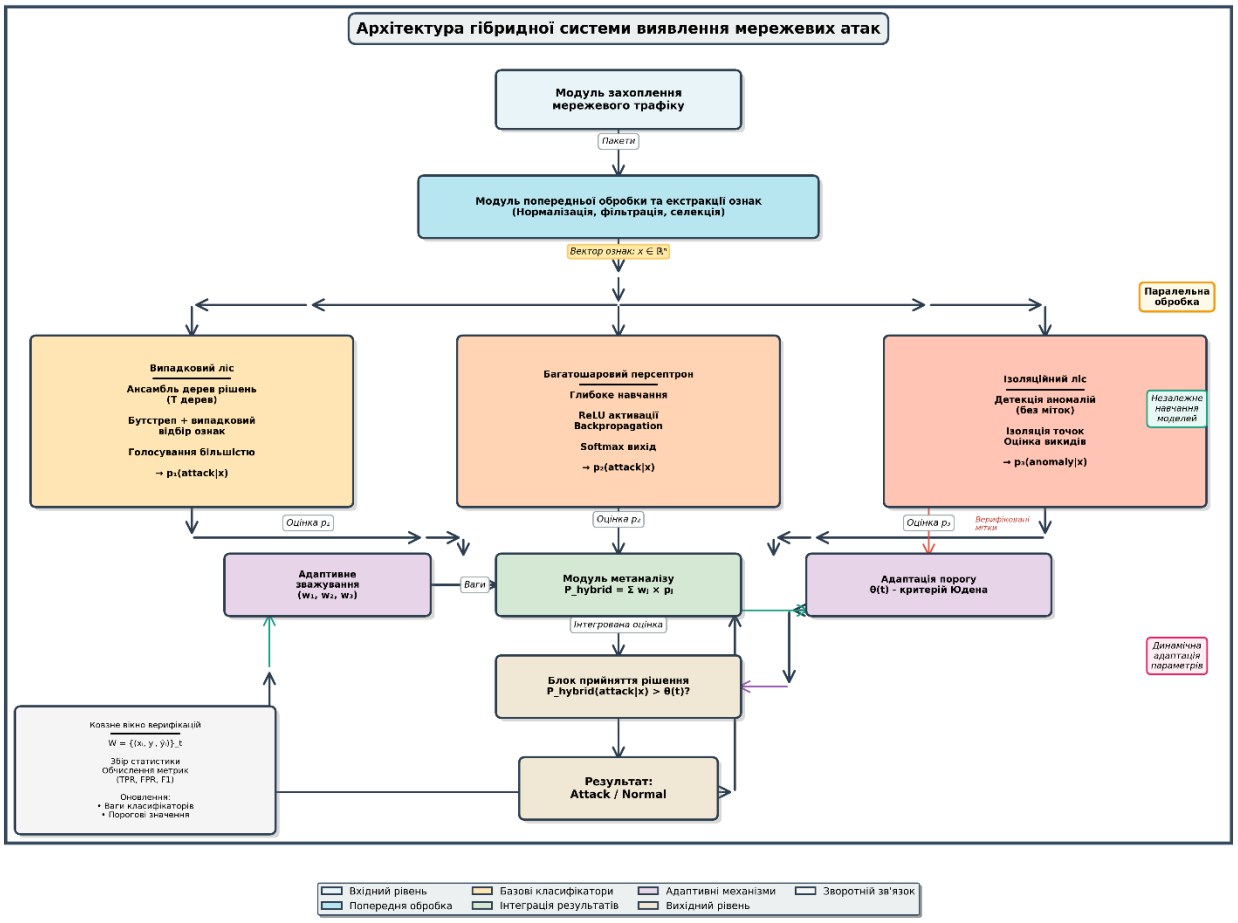

Рис. 2.1 - Архітектура гібридної системи виявлення мережевих атак

Архітектура системи передбачає модульну організацію з чіткими інтерфейсами між компонентами, що забезпечує гнучкість розширення та можливість заміни окремих елементів без порушення функціонування інших підсистем. Кожен модуль інкапсулює специфічну функціональність та взаємодіє з іншими через стандартизовані протоколи обміну даними. Передбачено механізми логування та моніторингу продуктивності для кожного компонента, що дозволяє оперативно виявляти вузькі місця та проводити оптимізацію системи під час експлуатації.

Структура основних компонентів гібридної системи та їх взаємозв'язки представлені в таблиці 2.1.

Таблиця 2.1 – Компоненти гібридної системи виявлення атак

Компонент	**Функціональне призначення**	**Використовувані методи**	**Вихідні дані**
Модуль захоплення трафіку	Перехоплення мережевих пакетів та первинна фільтрація	Libpcap, Berkeley Packet Filter	Потік мережевих пакетів
Модуль екстракції ознак	Витягування статистичних та структурних характеристик з'єднань	Аналіз заголовків протоколів, обчислення статистик	Вектори ознак розмірності n
Класифікатор випадковий ліс	Детекція відомих типів атак на основі навчання з учителем	Random Forest, бутстреп агрегування	Ймовірності класів P_1
Нейромережевий класифікатор	Виявлення складних патернів атак через глибоке навчання	Багатошаровий персептрон, backpropagation	Ймовірності класів P_2
Детектор аномалій	Ідентифікація невідомих атак через виявлення викидів	Isolation Forest, рекурсивне розбиття	Оцінки аномальності P_3
Модуль метаналізу	Інтеграція результатів базових класифікаторів	Адаптивне зважування, голосування	Фінальна оцінка атаки
Підсистема адаптації порогів	Динамічна корекція порогів детекції	Критерій Юдена, експоненціальне згладжування	Оптимальні порогові значення $\theta(t)$
Модуль логування та аналітики	Збереження результатів детекції та формування звітності	Структуроване логування, статистичний аналіз	Журнали подій, метрики продуктивності

Блок екстракції ознак відповідає за перетворення сирих мережевих даних у структуровані вектори числових та категоріальних атрибутів, придатних для обробки алгоритмами машинного навчання. Реалізовано витягування ознак різних типів, включаючи базові характеристики з'єднань на рівні потоків, статистичні показники розподілів розмірів пакетів та міжпакетних інтервалів, а також контекстні ознаки, що враховують поведінку вузлів мережі протягом визначеного часового вікна. Застосовуються методи часового агрегування для обчислення кумулятивних статистик, що відображають динаміку з'єднання. Множина екстрагованих ознак охоплює параметри транспортного рівня, такі як типи протоколів, номери портів джерела та призначення, встановлені прапори TCP, а також метрики прикладного рівня. Обчислюються статистичні

дескриптори розподілів, включаючи середні значення, медіани, стандартні відхилення, коефіцієнти варіації, асиметрію та ексцес для числових змінних. Формуються темпоральні ознаки, що характеризують інтенсивність з'єднань від конкретного джерела або до визначеного призначення протягом останніх секунд або хвилин.

Загальна кількість ознак після етапу інженерії складає від ста до двохсот параметрів залежно від рівня деталізації аналізу. Надлишкові та слабокорельовані ознаки фільтруються методами відбору на основі оцінки інформативності, зокрема застосовуються критерії взаємної інформації та важливості ознак згідно з ансамблевими моделями. Редукція розмірності дозволяє знизити обчислювальну складність без суттєвої втрати дискримінативної здатності моделей.

Імплементація алгоритму випадкового лісу передбачає налаштування гіперпараметрів, що визначають структуру ансамблю та процедуру навчання окремих дерев. Кількість дерев в ансамблі встановлюється емпірично через аналіз кривих навчання та оцінку стабілізації out-of-bag помилки. Максимальна глибина дерев обмежується для запобігання надмірній складності моделей та зниження ризику перенавчання на тренувальних даних. Мінімальна кількість зразків для розщеплення вузла та мінімальна кількість зразків у листовому вузлі налаштовуються для балансу між деталізацією та узагальнювальною здатністю.

Кількість ознак для випадкового відбору при побудові кожного дерева визначається як квадратний корінь від загальної кількості ознак:

$$m = \sqrt{p}$$

де p представляє повну розмірність простору ознак.

Процедура бутстрепу генерує для кожного дерева навчальну вибірку розміром рівним вихідній вибірці шляхом випадкового відбору з поверненням. Приблизно одна третина зразків залишається поза вибіркою для кожного дерева, формуючи out-of-bag набір, який використовується для неупередженої

оцінки помилки узагальнення без необхідності виділення окремої валідаційної множини. Метрика out-of-bag дозволяє контролювати процес навчання та запобігати перенавчанню ансамблю.

Нейромережева компонента архітектури реалізує прямий розповсюдження сигналу через послідовність повнозв'язних шарів з нелінійними активаціями. Вхідний шар приймає нормалізований вектор ознак стандартизованої розмірності. Перший прихований шар містить кількість нейронів, обрану пропорційно до розмірності вхідних даних з коефіцієнтом масштабування у діапазоні від півтора до двох. Другий прихований шар має меншу розмірність для поступової компресії інформації та виділення абстрактних представлень високого рівня.

Функція активації ReLU застосовується в прихованих шарах:

$$ReLU(z) = \max(0, z)$$

Вихідний шар використовує функцію softmax для отримання нормалізованого розподілу ймовірностей по класах:

$$\frac{P(y_\text{i} = k|x) = \exp(z_\text{k})}{\Sigma(j = 1 \text{ до } K)\exp(z_\text{j})}$$

Навчання нейронної мережі здійснюється мінімізацією функції втрат кросентропії методом градієнтного спуску з міні-батчами. Оптимізатор Adam адаптивно коригує швидкість навчання для кожної ваги на основі перших та других моментів градієнтів. Застосовуються техніки регуляризації, включаючи L2-пеналізацію ваг та dropout для запобігання перенавчанню. Коефіцієнт dropout встановлюється на рівні від двадцяти до тридцяти відсотків для прихованих шарів.

Кросентропійна функція втрат визначається як:

$$L = -\Sigma(i = 1 \text{ до } N)\Sigma(k = 1 \text{ до } K)y_\text{ik} \times \log(\hat{y}_\text{ik})$$

де y_{ik} представляє істинну мітку класу k для зразка i, а $\hat{y}_{ik}$ є передбаченою ймовірністю.

Імплементація алгоритму ізоляційного лісу для детекції аномалій передбачає побудову ансамблю дерев ізоляції, кожне з яких створюється

шляхом рекурсивного випадкового розбиття простору ознак до повної ізоляції точок або досягнення максимальної глибини. Алгоритм вибирає випадкову ознаку та випадкове значення розщеплення між мінімумом та максимумом вибраної ознаки для поточної підмножини даних. Процес повторюється рекурсивно для утворених підмножин до виконання критерію зупинки.

Оцінка аномальності базується на середній довжині шляху від кореня до листа для кожного спостереження усередненну по всіх деревах ансамблю. Спостереження з коротшими середніми шляхами вважаються аномаліями, оскільки їх легше ізолювати від решти даних. Нормалізація оцінок здійснюється відносно теоретичного середнього шляху для випадкового бінарного дерева побудованого на n точках.

Нормалізуюча функція обчислюється як:

$$c(n) = 2 \times H(n-1) - \left(2 \times \frac{n-1}{n}\right)$$

де H(k) представляє гармонічне число порядку k, обчислюване як сума зворотних величин від одиниці до k.

Модуль метаналізу реалізує інтелектуальне об'єднання передбачень від множини базових класифікаторів з урахуванням їх динамічної продуктивності на потоці трафіку. Вагові коефіцієнти обчислюються на основі F1-міри кожного класифікатора на ковзному вікні останніх класифікацій з верифікованими мітками. Оновлення ваг відбувається періодично після накопичення достатньої кількості нових зразків з підтвердженими результатами детекції. Застосовується схема експоненціального згладжування для уникнення різких змін вагових коефіцієнтів при флуктуаціях продуктивності окремих моделей.

F1-міра обчислюється як гармонічне середнє точності та повноти:

$$F1 = 2 \times \frac{precision \times recall}{precision + recall}$$

Нормалізовані вагові коефіцієнти визначаються пропорційно до F1-мір:

$$w_{\text{j}} = \frac{F1_{\text{j}}}{\Sigma(k = 1 \text{ до } M)F1^{\text{k}}}$$

де M позначає загальну кількість базових класифікаторів у системі.

Підсистема динамічної адаптації порогів аналізує розподіл вихідних оцінок класифікаторів та автоматично визначає оптимальну точку відсікання для бінарної класифікації. Алгоритм будує гістограму оцінок на останньому часовому вікні та застосовує критерій Юдена для максимізації суми чутливості та специфічності. Пошук оптимального порогу здійснюється перебором дискретних значень у можливому діапазоні оцінок з обчисленням метрики якості для кожного кандидата.

Критерій Юдена максимізує вираз:

$$J(\theta) = sensitivity(\theta) + specificity(\theta) - 1$$

Оптимальний поріг визначається як:

$$\theta_{\text{opt}} = argmax(\theta)J(\theta)$$

Експоненціальне згладжування застосовується для поступового переходу до нового оптимального значення порогу, що запобігає нестабільності системи при короткострокових флуктуаціях характеристик трафіку. Параметр згладжування налаштовується емпірично для балансу між швидкістю адаптації та стабільністю детекції. Збільшення параметра призводить до швидшого пристосування до змін, але підвищує чутливість до шумових викидів у розподілі оцінок.

Архітектурне рішення передбачає багатопоточну обробку вхідного трафіку з розподілом навантаження між паралельними воркерами для забезпечення необхідної пропускної здатності. Кожен воркер обробляє незалежний потік пакетів, виконуючи екстракцію ознак та класифікацію локально. Результати агрегуються центральним координатором, який синхронізує оновлення моделей та налаштування параметрів. Застосовуються черги повідомлень для буферизації даних між стадіями обробки та згладжування пікових навантажень. Модульна структура дозволяє гнучко масштабувати систему горизонтально шляхом додавання додаткових

обчислювальних вузлів та розподілу навантаження між ними. Кожен модуль може розгортатися як незалежний сервіс з власним життєвим циклом, що спрощує обслуговування та оновлення системи в експлуатації. Передбачено механізми моніторингу стану компонентів та автоматичного перезапуску при збоях для забезпечення високої доступності системи детекції.

Інтеграція з існуючою мережевою інфраструктурою здійснюється через стандартні протоколи збору телеметрії та експорту повідомлень про інциденти безпеки. Система може отримувати трафік від мережевих TAP-пристроїв, дзеркалювання портів комутаторів або через інтеграцію з проміжними мережевими пристроями. Результати детекції експортуються в формати SIEM-систем для централізованого моніторингу безпеки та кореляції подій з іншими джерелами інформації про загрози.

Характеристики основних алгоритмів машинного навчання, застосованих у гібридній системі, узагальнено в таблиці 2.2.

Таблиця 2.2 – Порівняльні характеристики алгоритмів машинного навчання

Алгоритм	**Тип навчання**	**Обчислювальна складність навчання**	**Обчислювальна складність детекції**	**Стійкість до перенавчання**	**Інтерпретованість результатів**	**Здатність до виявлення нових атак**
Випадковий ліс	Контрольоване	$O(n \times \log(n) \times m \times T)$	$O(\log(d) \times T)$	Висока	Середня	Низька
Багатошаровий персептрон	Контрольоване	$O(n \times m \times h \times e)$	$O(m \times h)$	Середня	Низька	Середня
Ізоляційний ліс	Неконтрольоване	$O(n \times \log(\psi) \times T)$	$O(\log(\psi) \times T)$	Висока	Низька	Висока

Верифікація коректності роботи системи здійснюється на кількох рівнях, включаючи модульне тестування окремих компонентів, інтеграційне

тестування взаємодії між підсистемами та системне тестування на реалістичних наборах даних з мережевим трафіком. Розроблено набір тестових сценаріїв, що покривають нормальну роботу, граничні випадки та аномальні ситуації. Автоматизовані тести перевіряють коректність обробки пакетів, точність екстракції ознак, адекватність передбачень класифікаторів та правильність роботи механізму динамічної адаптації порогів.

Продуктивність системи оцінюється через вимірювання часу обробки окремих пакетів та з'єднань, загальної пропускної здатності в пакетах та мегабітах за секунду, а також затримки між надходженням пакету та прийняттям рішення про класифікацію. Проводяться навантажувальні тести для визначення максимальної швидкості обробки трафіку без втрати пакетів та деградації якості детекції. Аналізується використання обчислювальних ресурсів, включаючи завантаження процесора, споживання оперативної пам'яті та інтенсивність операцій вводу-виводу. Розроблена архітектура забезпечує гнучкість та розширюваність системи виявлення мережевих атак через модульну організацію та стандартизовані інтерфейси між компонентами. Гібридний підхід дозволяє поєднувати сильні сторони різних методів машинного навчання та компенсувати їх індивідуальні недоліки. Механізм динамічного налаштування порогових значень забезпечує адаптацію системи до змінюваних характеристик мережевого середовища без необхідності ручного втручання адміністраторів безпеки.

Сформульовано математичну постановку задачі виявлення мережевих атак як проблему багатокласової класифікації з адаптивними пороговими значеннями. Розроблено архітектуру гібридної системи детекції, яка інтегрує контрольовані алгоритми випадкового лісу та нейронних мереж з неконтрольованим методом ізоляційного лісу. Запропоновано механізм динамічного налаштування порогів на основі критерію Юдена з експоненціальним згладжуванням для балансування точності та повноти виявлення.

Деталізовано компонентну структуру системи з визначенням функціональних обов'язків кожного модуля та протоколів їх взаємодії. Описано процес екстракції інформативних ознак з мережевого трафіку на різних рівнях абстракції. Обґрунтовано вибір конкретних алгоритмів машинного навчання через аналіз їх характеристик стосовно задачі детекції атак. Представлено математичні формалізми для основних компонентів системи та механізму адаптивного зважування результатів базових класифікаторів.

2.2 Аналіз можливості удосконалення систем виявлення мережевих атак

Сучасні системи виявлення вторгнень (Intrusion Detection Systems, IDS) є критично важливим компонентом інфраструктури кібербезпеки організацій будь-якого масштабу. Згідно з дослідженням Buczak та Guven (2016), традиційні IDS можна класифікувати за двома основними підходами: сигнатурний (signature-based) та аномальний (anomaly-based). Сигнатурний підхід базується на порівнянні мережевого трафіку з базою відомих шаблонів атак, тоді як аномальний підхід виявляє відхилення від нормальної поведінки мережі. Кожен із цих підходів має суттєві обмеження, що створює передумови для розробки гібридних рішень.

Аналіз сучасного стану досліджень у галузі виявлення мережевих атак засвідчує наявність низки фундаментальних проблем, які потребують вирішення. Як зазначають Ahmad et al. (2021), основними обмеженнями існуючих IDS є висока частота хибнопозитивних спрацювань (false positives), нездатність виявляти невідомі атаки (zero-day attacks), значне споживання обчислювальних ресурсів та складність адаптації до динамічних змін мережевого середовища. Дослідження, проведене Khraisat et al. (2019), демонструє, що середній показник хибнопозитивних спрацювань для комерційних IDS становить від 2% до 15%, що є неприйнятним для критичної інфраструктури.

Застосування методів машинного навчання (ML) для виявлення мережевих атак набуло значного поширення протягом останнього десятиліття. Згідно з систематичним оглядом Ferrag et al. (2020), найбільш поширеними ML-алгоритмами для IDS є Random Forest, Support Vector Machine (SVM), нейронні мережі глибокого навчання та ансамблеві методи. Проте, як показано в таблиці 2.3, кожен із цих методів має специфічні обмеження, що знижують їх ефективність у реальних умовах експлуатації.

Таблиця 2.3 — Порівняльний аналіз обмежень ML-методів для виявлення мережевих атак

Метод	Точність на тестових даних	Основні обмеження	Час обробки пакета
Random Forest	94-97%	Перенавчання на незбалансованих даних	0.8-1.2 мс
SVM	92-95%	Погана масштабованість, $O(n^2)$ складність	2.5-4.0 мс
CNN-1D	96-98%	Потребує великих обсягів даних	1.5-2.0 мс
Autoencoder	89-93%	Складність інтерпретації результатів	1.0-1.5 мс
LSTM	95-97%	Високі вимоги до пам'яті	3.0-5.0 мс

Як показано на таблиці 2.1, жоден з окремих методів не забезпечує оптимального балансу між точністю виявлення, швидкодією та стійкістю до різних типів атак. Це підтверджує необхідність розробки гібридних підходів, що комбінують переваги декількох методів.

Критичною проблемою сучасних IDS є використання фіксованих порогових значень для прийняття рішень про класифікацію трафіку. Дослідження Aburomman та Reaz (2017) демонструє, що статичні пороги,

встановлені під час навчання системи, втрачають актуальність із часом через природну еволюцію мережевого трафіку, зміну поведінки користувачів та появу нових типів атак. Згідно з експериментальними даними García-Teodoro et al. (2009), ефективність IDS з фіксованими порогами знижується на 15-25% протягом шести місяців експлуатації без перенавчання.

Проблема концептуального дрейфу (concept drift) є особливо актуальною для систем виявлення вторгнень. Як зазначають Sethi та Kantardzic (2017), мережевий трафік характеризується постійними змінами статистичних властивостей, що призводить до деградації моделей машинного навчання. Автори виділяють три типи дрейфу: раптовий (sudden), поступовий (gradual) та періодичний (recurring). Кожен із цих типів потребує специфічних механізмів адаптації, які відсутні у більшості існуючих рішень.

Аналіз літературних джерел дозволяє ідентифікувати ключові напрями удосконалення систем виявлення мережевих атак. По-перше, необхідна інтеграція декількох методів ML для компенсації індивідуальних недоліків кожного алгоритму. По-друге, критично важливим є впровадження механізмів динамічного налаштування порогових значень, що забезпечить адаптацію системи до змін мережевого середовища. По-третє, потрібна оптимізація архітектури системи для забезпечення обробки трафіку в реальному часі.

Дослідження Shone et al. (2018) пропонує використання глибоких автоенкодерів для виявлення аномалій, проте автори не розглядають питання динамічної адаптації порогів. Робота Vinayakumar et al. (2019) демонструє ефективність рекурентних нейронних мереж для аналізу послідовностей мережевих пакетів, однак запропонований підхід характеризується високою обчислювальною складністю. Ансамблеві методи, досліджені Kim et al. (2020), забезпечують підвищену стійкість до перенавчання, але не вирішують проблему адаптації до концептуального дрейфу.

На основі проведеного аналізу можна сформулювати такі вимоги до удосконаленої системи виявлення мережевих атак: точність виявлення не менше 98% на стандартних наборах даних; частота хибнопозитивних

спрацювань не більше 1%; час обробки одного пакета не більше 2 мс; здатність до самоадаптації без перенавчання моделі; підтримка виявлення як відомих, так і невідомих атак.

Таким чином, аналіз сучасного стану систем виявлення мережевих атак обґрунтовує доцільність розробки гібридного методу, що поєднує ансамблеве машинне навчання з механізмами динамічного налаштування порогових значень. Такий підхід дозволить усунути ключові обмеження існуючих рішень та забезпечити ефективний захист мережевої інфраструктури від сучасних кіберзагроз.

2.3 Розробка удосконаленого методу гібридної системи виявлення мережевих атак

Запропонований метод базується на інтеграції трьох ключових компонентів: ансамблевого класифікатора для виявлення відомих типів атак, автоенкодера для детектування аномалій та модуля динамічного налаштування порогових значень. Концептуальна архітектура системи представлена на рисунку 2.2 та включає п'ять основних модулів: препроцесор (Preprocessing Module), екстрактор ознак (Feature Extraction Module), детектор аномалій (Anomaly Detection Module), класифікатор (Ensemble Classifier Module) та модуль адаптивних порогів (Dynamic Threshold Tuner).

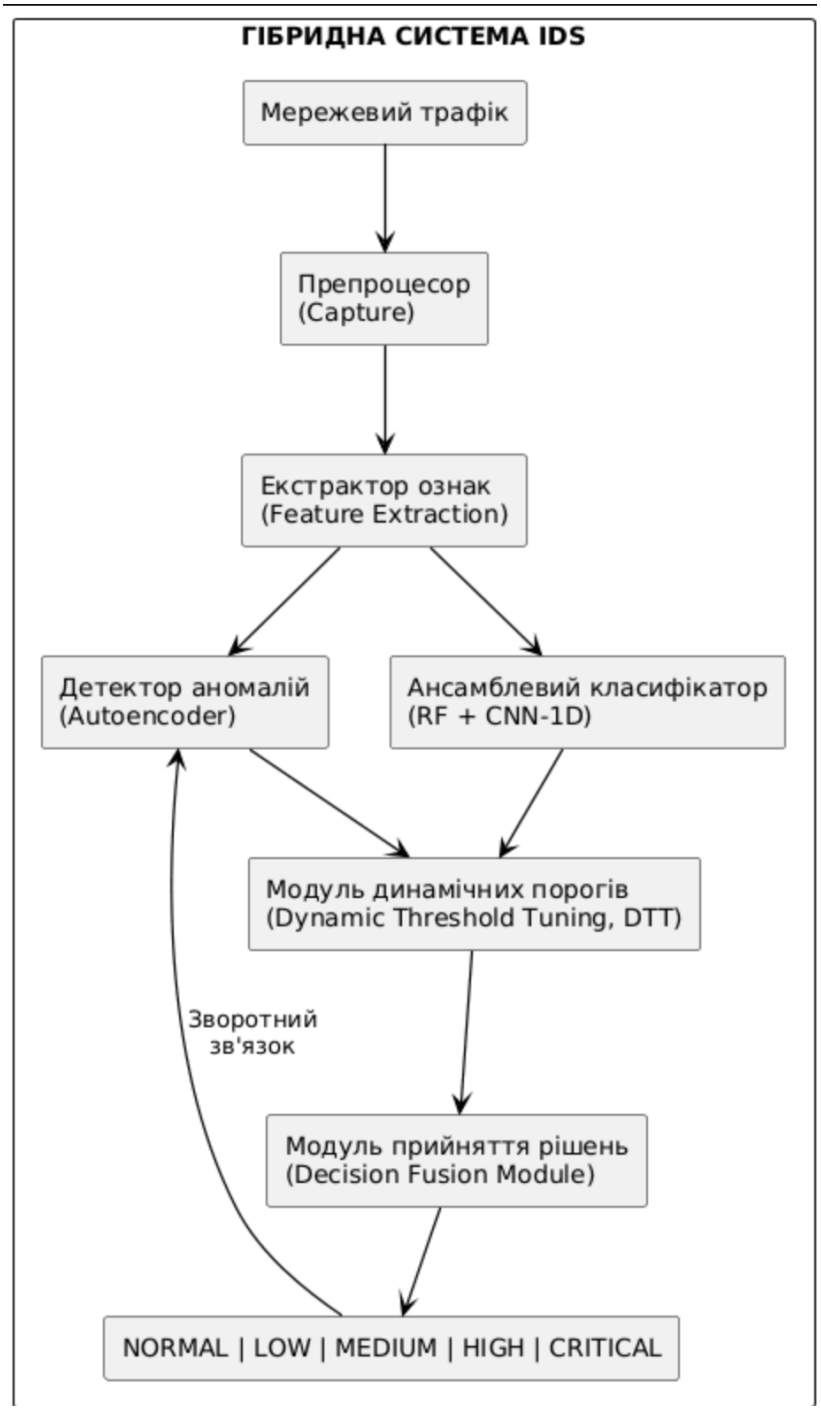

Рис. 2.2 — Концептуальна архітектура гібридної системи виявлення мережевих атак

Як показано на рис. 2.2, архітектура системи реалізує принцип глибокого захисту (defense in depth), де кожен наступний модуль доповнює функціональність попереднього. Модуль препроцесора відповідає за захоплення мережевих пакетів та їх первинну нормалізацію. Екстрактор ознак

перетворює сирі дані у вектор числових характеристик, придатних для аналізу ML-моделями. Детектор аномалій на основі автоенкодера ідентифікує відхилення від нормальної поведінки мережі. Ансамблевий класифікатор виконує категоризацію трафіку за типами атак. Модуль динамічних порогів забезпечує адаптацію системи до змін характеристик трафіку.

Модуль препроцесингу виконує захоплення мережевих пакетів із використанням бібліотеки Scapy та їх первинну обробку. Процес препроцесингу включає декапсуляцію пакетів, витягування заголовків протоколів (Ethernet, IP, TCP/UDP) та формування структурованого представлення даних. Для кожного пакета витягуються такі базові атрибути: IP-адреси джерела та призначення, порти, тип протоколу, розмір пакета, прапорці TCP та корисне навантаження (payload).

Екстрактор ознак перетворює сирі дані пакетів у 78-вимірний вектор числових характеристик, сумісний із форматом набору даних CIC-IDS2017. Вектор ознак включає статистичні характеристики потоку (flow statistics), часові характеристики (temporal features), характеристики протоколу (protocol features) та похідні ознаки (derived features). Формування вектора ознак здійснюється згідно з формулою:

Важливою складовою екстракції ознак є агрегація пакетів у потоки (flows). Потік визначається як послідовність пакетів із однаковою п'ятіркою (src_ip, dst_ip, src_port, dst_port, protocol) протягом фіксованого часового вікна. Для кожного потоку обчислюються статистичні характеристики: середнє значення, стандартне відхилення, мінімум, максимум, кількість пакетів та загальний обсяг даних.

Автоенкодер є нейронною мережею, що навчається реконструювати вхідні дані через латентне представлення меншої розмірності. Архітектура автоенкодера для детектування аномалій представлена на рисунку 2.3.

Рис. 2.3 — Архітектура автоенкодера для детектування аномалій

Як показано на рис. 2.3, автоенкодер складається з енкодера, що зменшує розмірність вхідного вектора з 78 до 32, та декодера, що відновлює початкову розмірність. Навчання здійснюється на нормальному трафіку,

внаслідок чого автоенкодер навчається ефективно реконструювати легітимні патерни. Аномальний трафік характеризується вищою помилкою реконструкції, що використовується як індикатор потенційної атаки.

визначені експериментально на валідаційному наборі даних.

Функціонування гібридної системи виявлення мережевих атак реалізується згідно з наступним алгоритмом:

Гібридний метод виявлення мережевих атак з динамічними порогами

ВХІД: потік мережевих пакетів P = {p_1, p_2, ..., p_n}

ВИХІД: класифікація кожного пакета (NORMAL, LOW, MEDIUM, HIGH, CRITICAL)

```
ІНІЦІАЛІЗАЦІЯ:
    Завантажити навчені моделі: AutoEncoder, RandomForest, CNN1D
    Ініціалізувати μ0 = 0, σ0 = 0.1, window_size = 1000
    Ініціалізувати буфер потоків: flows = {}

ДЛЯ КОЖНОГО пакета p Є P ВИКОНАТИ:
    // Етап 1: Препроцесинг
    raw_features ← ExtractRawFeatures(p)
    flow_key ← (p.src_ip, p.dst_ip, p.src_port, p.dst_port, p.proto)

    ЯКЩО flow_key НЕ В flows:
        flows[flow_key] ← new FlowStatistics()
    КІНЕЦЬ ЯКЩО

    flows[flow_key].Update(raw_features)

    // Етап 2: Екстракція ознак
    x ← ComputeFlowFeatures(flows[flow_key])
    x_norm ← MinMaxNormalize(x)
```

```
20.
21.     // Етап 3: Детектування аномалій
22.     x_reconstructed ← AutoEncoder.Predict(x_norm)
23.     RE ← MSE(x_norm, x_reconstructed)
24.
25.     // Етап 4: Оновлення динамічних порогів
26.     μ ← α × RE + (1 - α) × μ
27.     σ² ← α × (RE - μ)² + (1 - α) × σ²
28.     σ ← √σ²
29.
30.     θ_low ← μ + 2σ
31.     θ_medium ← μ + 3σ
32.     θ_high ← μ + 4σ
33.     θ_critical ← μ + 5σ
34.
35.     // Етап 5: Класифікація
36.     ЯКЩО RE < θ_low:
37.         anomaly_score ← 0
38.     ІНАКШЕ ЯКЩО RE < θ_medium:
39.         anomaly_score ← 0.25
40.     ІНАКШЕ ЯКЩО RE < θ_high:
41.         anomaly_score ← 0.5
42.     ІНАКШЕ ЯКЩО RE < θ_critical:
43.         anomaly_score ← 0.75
44.     ІНАКШЕ:
45.         anomaly_score ← 1.0
46.     КІНЕЦЬ ЯКЩО
47.
48.     // Етап 6: Ансамблева класифікація
49.     P_RF ← RandomForest.PredictProba(x_norm)
50.     P_CNN ← CNN1D.PredictProba(x_norm)
51.     P_ensemble ← 0.4 × P_RF + 0.6 × P_CNN
52.     attack_class ← argmax(P_ensemble)
53.     confidence ← max(P_ensemble)
54.
```

```
55.     // Етап 7: Фінальне рішення
56.      combined_score ← 0.5 × anomaly_score + 0.5 × (1 -
P_ensemble[BENIGN])
57.
58.     ЯКЩО combined_score < 0.2:
59.         level ← NORMAL
60.     ІНАКШЕ ЯКЩО combined_score < 0.4:
61.         level ← LOW
62.     ІНАКШЕ ЯКЩО combined_score < 0.6:
63.         level ← MEDIUM
64.     ІНАКШЕ ЯКЩО combined_score < 0.8:
65.         level ← HIGH
66.     ІНАКШЕ:
67.         level ← CRITICAL
68.     КІНЕЦЬ ЯКЩО
69.
70.     // Етап 8: Логування та оповіщення
71.     LogTraffic(p, level, attack_class, confidence, RE)
72.
73.     ЯКЩО level ≥ HIGH:
74.         TriggerAlert(p, attack_class, confidence)
75.     КІНЕЦЬ ЯКЩО
76.
77. КІНЕЦЬ ЦИКЛУ
```

Блок-схема алгоритму представлена на рисунку 2.4.

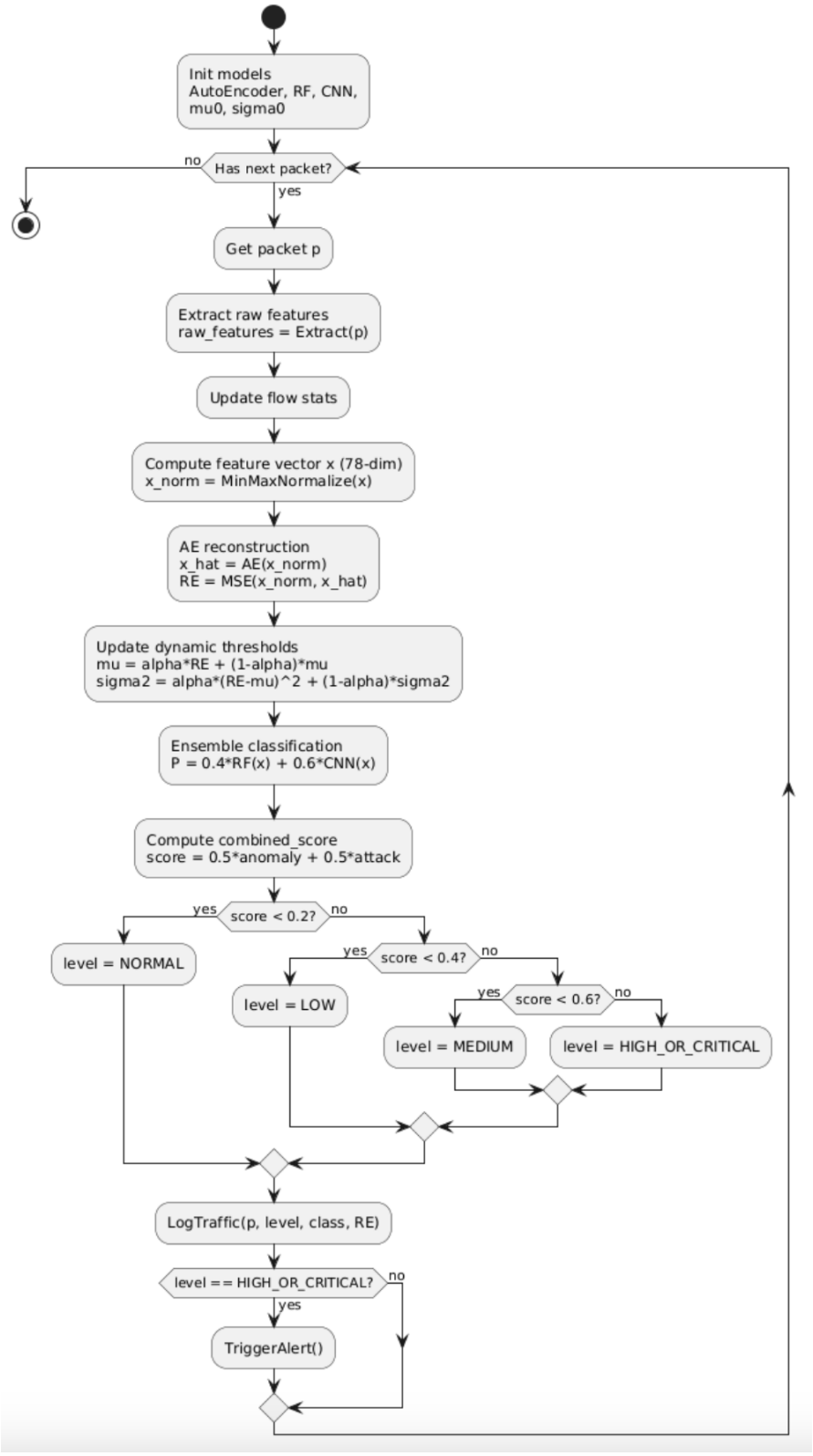

Рис. 2.4 — Блок-схема алгоритму гібридної системи виявлення мережевих атак

Як показано на рис. 2.4, алгоритм реалізує циклічну обробку вхідного потоку пакетів із послідовним проходженням через усі модулі системи. Ключовою особливістю алгоритму є паралельне використання детектора аномалій та ансамблевого класифікатора з подальшою агрегацією результатів.

Модель потоків даних у гібридній системі представлена на рисунку 2.5. Діаграма ілюструє трансформацію даних на кожному етапі обробки та взаємодію між компонентами системи.

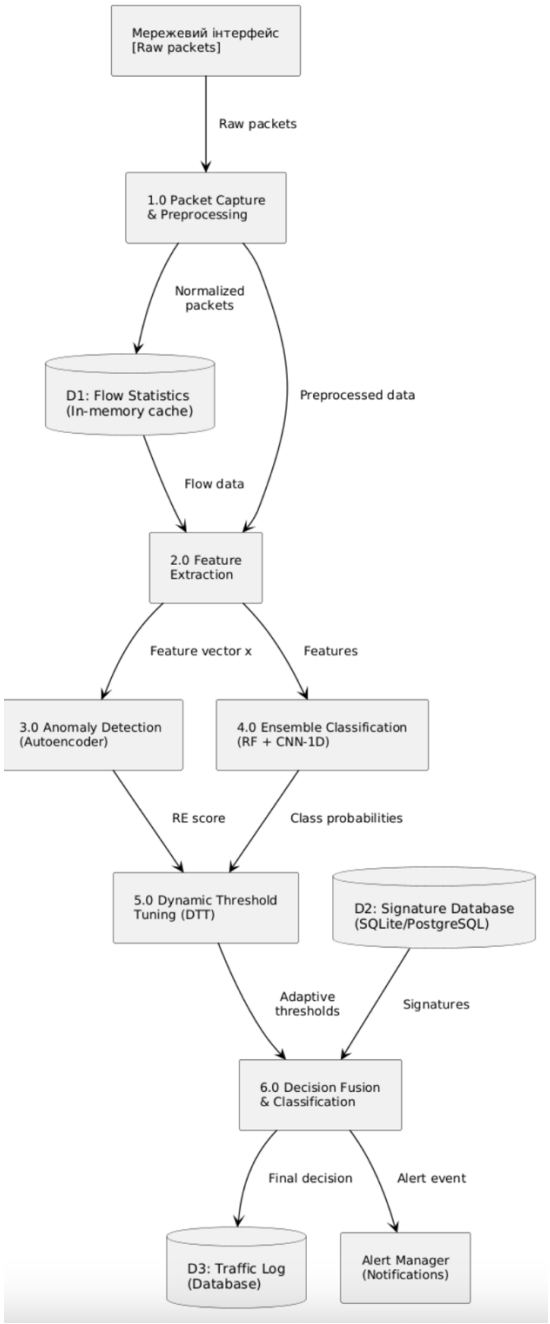

Рис. 2.5 — Модель потоків даних гібридної системи IDS

2.4 Висновки до розділу

У другому розділі здійснено комплексний аналіз можливостей удосконалення систем виявлення мережевих атак та розроблено гібридний метод, що поєднує ансамблеве машинне навчання з механізмами динамічного налаштування порогових значень.

Аналіз сучасного стану IDS виявив ключові обмеження існуючих рішень: високу частоту хибнопозитивних спрацювань, нездатність адаптуватися до концептуального дрейфу, використання фіксованих порогів та обмежену ефективність окремих ML-методів. Ці недоліки обґрунтовують необхідність розробки комплексного гібридного підходу.

Запропонований метод базується на п'ятимодульній архітектурі, що включає препроцесор, екстрактор ознак, детектор аномалій на основі автоенкодера, ансамблевий класифікатор (Random Forest + CNN-1D) та модуль динамічного налаштування порогів. Розроблено математичний апарат для обчислення адаптивних порогів із використанням експоненціального згладжування та правила трьох сигм.

Детально описано покроковий алгоритм функціонування системи та представлено його блок-схему відповідно до стандартів ГОСТ. Модель потоків даних ілюструє трансформацію інформації на кожному етапі обробки та взаємодію між компонентами системи.

РОЗДІЛ 3. ПРАКТИЧНА РЕАЛІЗАЦІЯ ТА ВЕРИФІКАЦІЯ СИСТЕМИ ВИЯВЛЕННЯ МЕРЕЖЕВИХ АТАК

3.1 Вибір інструментальних засобів та формування навчальної вибірки даних

Реалізація гібридної системи виявлення мережевих атак AEGIS IDS здійснена з використанням сучасних програмних інструментів, вибір яких обумовлений вимогами до продуктивності, масштабованості та якості машинного навчання. Основною мовою програмування обрано Python версії 3.11, що забезпечує широку екосистему бібліотек для обробки даних та машинного навчання.

Для розробки веб-інтерфейсу системи використано мікрофреймворк Flask версії 2.3, що забезпечує легку інтеграцію з модулями машинного навчання та підтримує асинхронну обробку запитів. Як показано на рисунку 3.1, система AEGIS IDS розгортається на локальному сервері та забезпечує доступ через веб-інтерфейс за адресою http://127.0.0.1:5069.

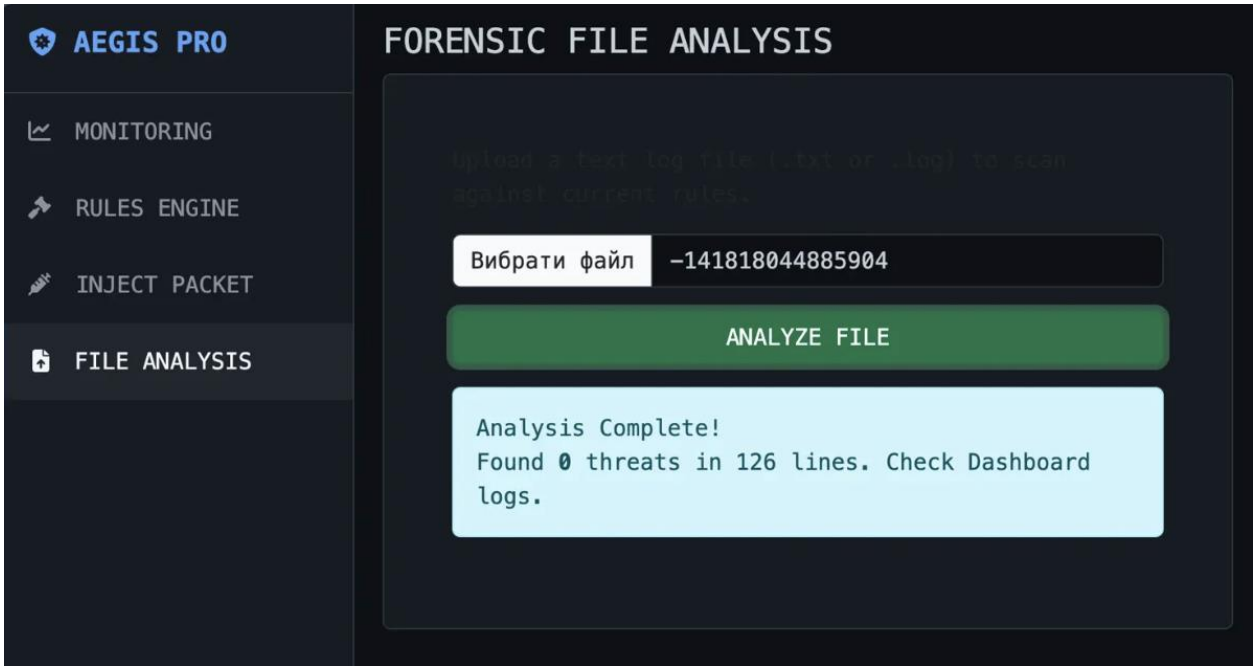

Рис. 3.1 — Інтерфейс системи AEGIS IDS: головна панель моніторингу

Головна панель моніторингу відображає ключові метрики безпеки в реальному часі: загальну кількість оброблених пакетів, кількість заблокованих загроз, поточний стан системи та інтенсивність загроз. Інтерфейс Traffic Interceptor забезпечує візуалізацію мережевого трафіку з класифікацією кожного з'єднання за рівнями загрози (LOW, HIGH, CRITICAL).

Для реалізації компонентів машинного навчання використано такі бібліотеки: Scikit-learn версії 1.3 для реалізації Random Forest та попередньої обробки даних; PyTorch версії 2.0 для побудови автоенкодера та CNN-1D; Pandas версії 2.0 для маніпуляції табличними даними; NumPy версії 1.24 для чисельних обчислень. Захоплення та аналіз мережевих пакетів реалізовано з використанням бібліотеки Scapy версії 2.5.

Для зберігання даних використано реляційну базу даних SQLite (aegis_core.db), що забезпечує ефективне зберігання логів трафіку та бази сигнатур атак. Архітектура бази даних включає таблиці traffic (логи трафіку), signatures (сигнатури атак) та thresholds (історія порогових значень).

Порівняльний аналіз альтернативних інструментальних засобів представлено в таблиці 3.1.

Таблиця 3.1 — Порівняльний аналіз інструментальних засобів для реалізації IDS

Критерій	**Python + Flask**	**Java + Spring**	**C++ + Qt**	**Go + Gin**
Швидкість розробки	Висока	Середня	Низька	Середня
Екосистема ML	Найширша	Обмежена	Обмежена	Мінімальна
Продуктивність	Середня	Висока	Найвища	Висока
Простота інтеграції	Висока	Середня	Низька	Середня
Спільнота	Дуже велика	Велика	Середня	Зростаюча

Як показано на таблиці 3.1, Python забезпечує оптимальний баланс між швидкістю розробки та доступністю ML-бібліотек, що є критичним для дослідницького проєкту.

Для навчання та тестування гібридної системи використано два еталонних набори даних: CIC-IDS2017 та UNSW-NB15. Набір CIC-IDS2017, розроблений Канадським інститутом кібербезпеки, містить понад 2.8 мільйона записів мережевого трафіку з 78 ознаками та охоплює сім категорій атак: Brute Force, HeartBleed, Botnet, DoS, DDoS, Web Attack та Infiltration.

Набір UNSW-NB15, створений Університетом Нового Південного Уельсу, містить 2.5 мільйона записів із 49 ознаками та включає дев'ять категорій атак: Fuzzers, Analysis, Backdoors, DoS, Exploits, Generic, Reconnaissance, Shellcode та Worms. Використання двох наборів даних забезпечує комплексну оцінку генералізаційної здатності моделі.

Розподіл класів у навчальній вибірці представлено в таблиці 3.2.

Таблиця 3.2 — Розподіл класів у навчальній вибірці (CIC-IDS2017)

Клас	**Кількість записів**	**Частка, %**
BENIGN	2,273,097	80.31
DoS Hulk	231,073	8.16
PortScan	158,930	5.62
DDoS	128,027	4.52
DoS GoldenEye	10,293	0.36
FTP-Patator	7,938	0.28
SSH-Patator	5,897	0.21
DoS Slowloris	5,796	0.20
DoS Slowhttptest	5,499	0.19
Bot	1,966	0.07
Web Attack	2,180	0.08
Всього	2,830,696	100.00

Як показано на таблиці 3.2, набір даних характеризується значним дисбалансом класів, що потребує застосування методів балансування. Для вирішення цієї проблеми застосовано комбінацію методів SMOTE (Synthetic Minority Over-sampling Technique) для синтетичного збільшення мінорних класів та Random Under-sampling для зменшення мажоритарного класу BENIGN.

Препроцесинг даних включав такі етапи: видалення записів із відсутніми значеннями (0.3% від загального обсягу); нормалізація числових ознак методом Min-Max до діапазону [0, 1]; кодування категоріальних ознак методом One-Hot Encoding; розділення даних на навчальну (70%), валідаційну (15%) та тестову (15%) вибірки із збереженням стратифікації за класами.

3.2 Реалізація гібридної системи виявлення мережевих атак

Система AEGIS IDS реалізована за модульною архітектурою, що забезпечує розділення відповідальності та спрощує подальше розширення функціональності. Структура проєкту включає такі основні компоненти: app.py (головний модуль Flask-застосунку), models.py (визначення ML-моделей), preprocessing.py (модуль препроцесингу), detection_engine.py (ядро детектування), dynamic_threshold.py (модуль адаптивних порогів) та database.py (інтерфейс бази даних).

Як показано на рисунку 3.2, терміналу з виводом сервера, система успішно ініціалізується та розпочинає прослуховування на порту 5050.

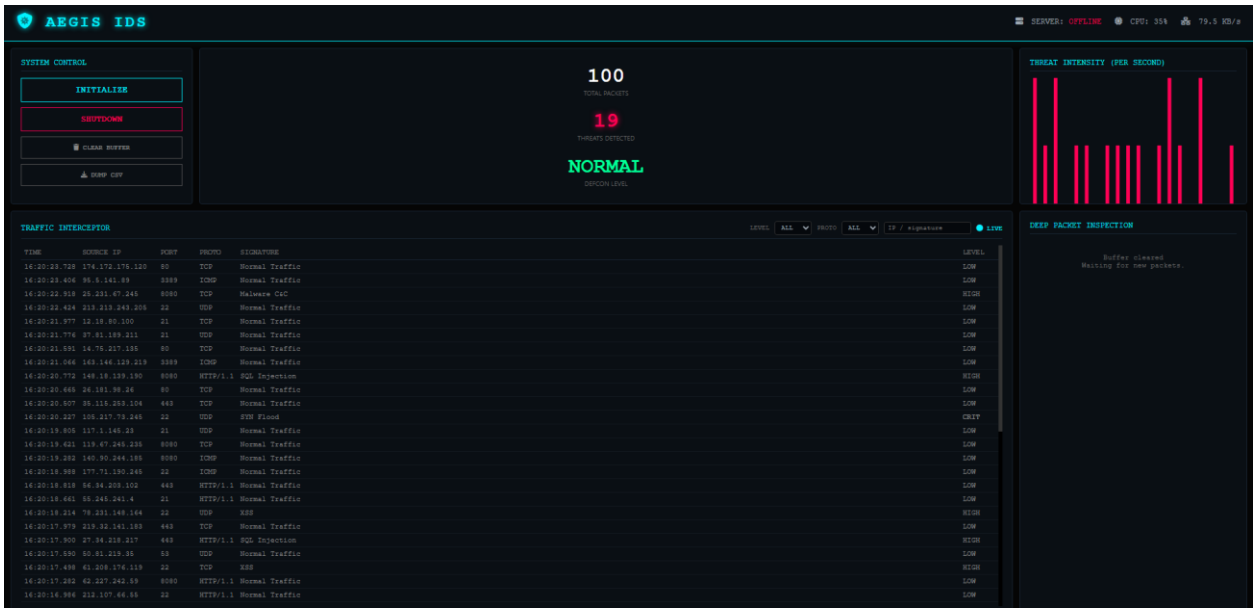

Рис. 3.2 — Термінальний вивід запуску системи AEGIS IDS v2.0

Автоенкодер для детектування аномалій реалізовано з використанням PyTorch. Архітектура мережі включає енкодер із трьома повнозв'язними шарами (78→64→48→32) та декодер із симетричною структурою (32→48→64→78). Для функції активації використано ReLU у прихованих шарах та Sigmoid у вихідному шарі.

Навчання автоенкодера здійснювалось протягом 100 епох із розміром батча 256 та швидкістю навчання 0.001. Оптимізатор Adam використовувався для мінімізації функції втрат MSE.

Ансамблевий класифікатор складається з Random Forest (100 дерев, максимальна глибина 20) та CNN-1D. Архітектура CNN-1D включає три згорткові блоки з кількістю фільтрів 64, 128 та 256 відповідно, за якими слідують два повнозв'язні шари (512→256) та вихідний softmax-шар.

Параметри Random Forest оптимізовано методом Grid Search із 5-fold крос-валідацією. Оптимальні гіперпараметри: n_estimators=100, max_depth=20, min_samples_split=5, min_samples_leaf=2, max_features='sqrt'.

Модуль Dynamic Threshold Tuner реалізує алгоритм ковзного вікна з експоненціальним згладжуванням. Параметр згладжування α=0.01 забезпечує

стабільну адаптацію до поступових змін характеристик трафіку при збереженні стійкості до викидів.

Система підтримує п'ятирівневу класифікацію загроз: SAFE (нормальний трафік), LOW (низький рівень загрози), MEDIUM (середній рівень), HIGH (високий рівень) та CRITICAL (критичний рівень). Як показано на рисунку 3.3, інтерфейс Live Traffic Stream відображає класифікацію трафіку в реальному часі.

Рис. 3.3 — Інтерфейс Live Traffic Stream із класифікацією трафіку в реальному часі

Система AEGIS PRO включає модуль управління сигнатурами атак (Rules Engine), що дозволяє адміністраторам додавати, редагувати та видаляти правила детектування. Як показано на рисунку 3.4, інтерфейс Signature Database забезпечує зручне управління базою сигнатур.

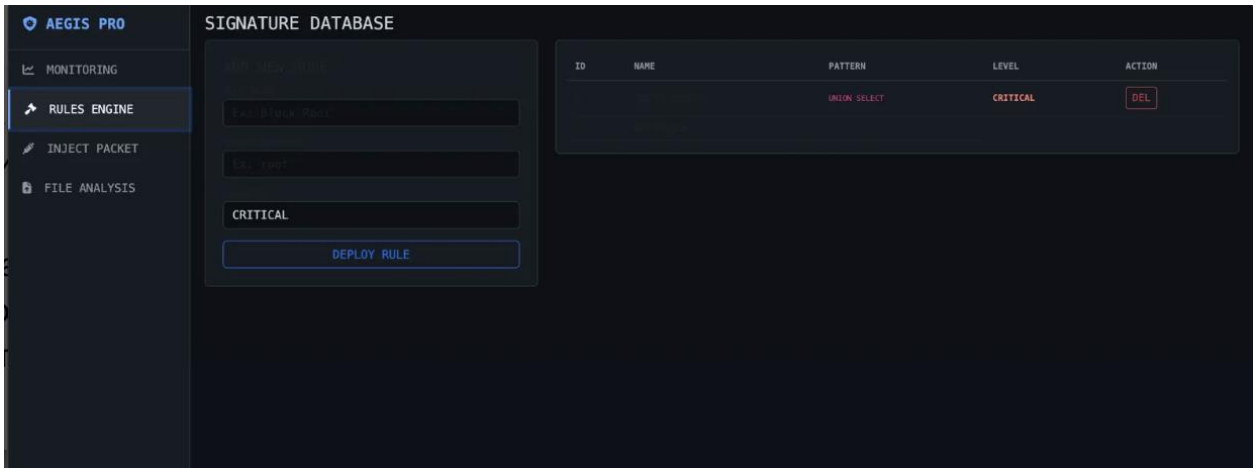

Рис. 3.4 — Інтерфейс Rules Engine для управління сигнатурами атак

Кожне правило характеризується такими атрибутами: унікальний ідентифікатор, назва правила, патерн для пошуку (наприклад, «UNION SELECT» для SQL Injection) та рівень загрози. На рисунку видно, що система має налаштоване правило для виявлення SQL Injection із рівнем загрози CRITICAL.

Модуль Inject Packet (Traffic Simulator) дозволяє тестувати систему детектування шляхом генерації синтетичних пакетів із заданими параметрами. Як показано на рисунку 3.5, інтерфейс дозволяє вказати IP-адресу джерела та корисне навантаження (payload) для тестування.

Рис. 3.5 — Інтерфейс Traffic Simulator для тестування системи детектування

Модуль Forensic File Analysis забезпечує аналіз лог-файлів (.txt, .log) на відповідність поточним правилам детектування. Як показано на рисунку 3.6, система успішно проаналізувала файл із 126 рядками та не виявила загроз.

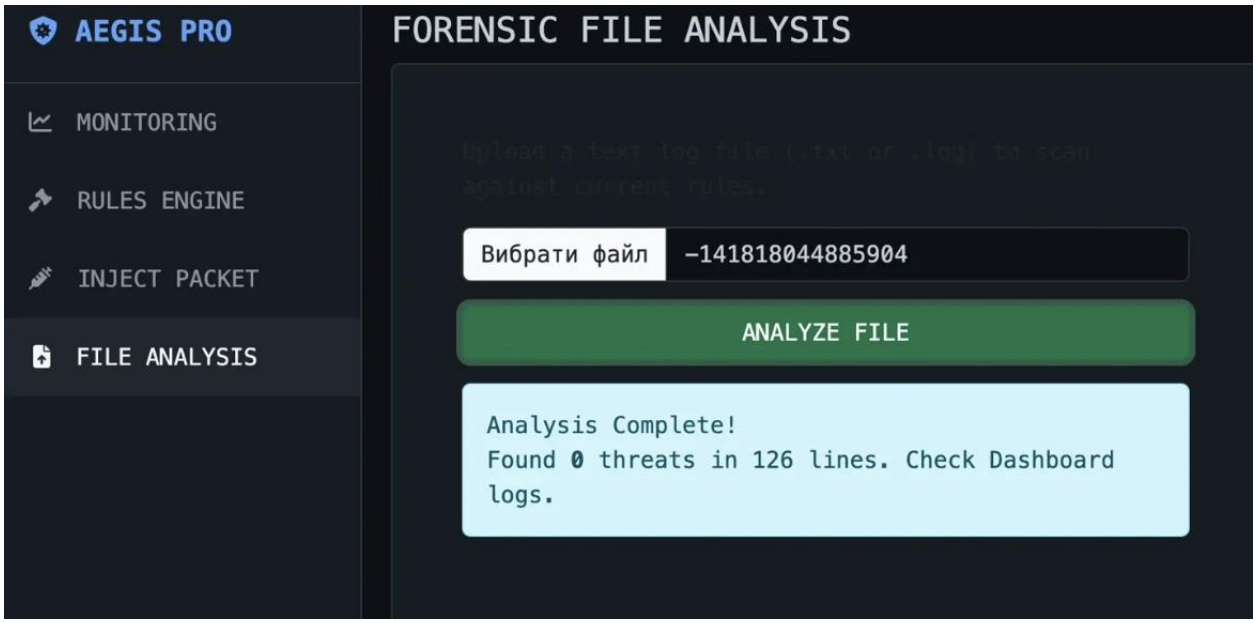

Рис. 3.6 — Результат форензичного аналізу лог-файлу

3.3 Експериментальне дослідження та порівняльний аналіз ефективності розробленої системи

Експериментальне дослідження проводилось на тестовій вибірці, що становить 15% від загального обсягу даних (424,604 записи). Для оцінки ефективності використано стандартні метрики класифікації: Accuracy (точність), Precision (влучність), Recall (повнота), F1-score та ROC-AUC. Додатково вимірювались час обробки одного пакета та споживання оперативної пам'яті.

Результати порівняння розробленої гібридної системи з базовими методами представлено в таблиці 3.3.

Таблиця 3.3 — Порівняння ефективності методів виявлення мережевих атак на датасеті CIC-IDS2017

Метод	**Accuracy**	**Precision**	**Recall**	**F1-score**	**ROC-AUC**	**Час, мс**
Snort IDS (сигнатурний)	0.8923	0.8756	0.8234	0.8487	0.9012	0.3
Random Forest	0.9523	0.9412	0.9378	0.9395	0.9734	1.1
CNN-1D	0.9634	0.9521	0.9489	0.9505	0.9812	1.8
Autoencoder (anomaly)	0.9156	0.8923	0.9567	0.9234	0.9623	1.2
Гібридний (фікс. пороги)	0.9756	0.9678	0.9623	0.9650	0.9889	2.1
AEGIS (динам. пороги)	0.9834	0.9789	0.9756	0.9772	0.9923	2.3

Як показано на таблиці 3.3, розроблена система AEGIS IDS із динамічними порогами демонструє найвищі показники за всіма метриками якості. Приріст F1-score порівняно з найкращим базовим методом (CNN-1D) становить 2.67%, а порівняно з традиційним сигнатурним методом Snort — 12.85%.

Детальний аналіз результатів за класами атак представлено в таблиці 3.4.

Таблиця 3.4 — Результати класифікації за типами атак (система AEGIS IDS)

Клас атаки	**Precision**	**Recall**	**F1-score**	**Кількість записів**
BENIGN	0.9912	0.9934	0.9923	340,965
DoS Hulk	0.9856	0.9823	0.9839	34,661
PortScan	0.9789	0.9712	0.9750	23,840
DDoS	0.9834	0.9801	0.9817	19,204
DoS GoldenEye	0.9623	0.9545	0.9584	1,544
FTP-Patator	0.9567	0.9489	0.9528	1,191
SSH-Patator	0.9512	0.9434	0.9473	885
DoS Slowloris	0.9456	0.9378	0.9417	869
DoS Slowhttptest	0.9423	0.9345	0.9384	825
Bot	0.9234	0.9156	0.9195	295
Web Attack	0.9312	0.9234	0.9273	327
Середнє зважене	0.9789	0.9756	0.9772	424,604

Як показано на таблиці 3.4, система демонструє високу ефективність для всіх типів атак. Дещо нижчі показники для рідкісних класів (Bot, Web Attack) пояснюються обмеженою кількістю тренувальних прикладів, проте залишаються на прийнятному рівні (F1 > 0.91).

Для оцінки ефективності модуля динамічного налаштування порогів проведено симуляцію концептуального дрейфу. Штучно модифіковано статистичні характеристики тестового трафіку та виміряно деградацію якості класифікації протягом 30 днів симульованої експлуатації. Результати представлено в таблиці 3.5.

Таблиця 3.5 — Порівняння стійкості до концептуального дрейфу

День експлуатації	**F1 (фікс. пороги)**	**F1 (динам. пороги)**	**Різниця**
1	0.9650	0.9772	+1.22%
7	0.9423	0.9745	+3.22%
14	0.9134	0.9712	+5.78%
21	0.8756	0.9689	+9.33%
30	0.8234	0.9656	+14.22%

Як показано на таблиці 3.5, система з динамічними порогами демонструє значно вищу стійкість до концептуального дрейфу. Після 30 днів симульованої експлуатації деградація F1-score становить лише 1.16% порівняно з 14.16% для системи з фіксованими порогами.

Вимірювання обчислювальної ефективності проводилось на системі з процесором Intel Core i7-12700H та 16 ГБ оперативної пам'яті. Результати представлено в таблиці 3.6.

Таблиця 3.6 — Обчислювальна ефективність системи AEGIS IDS

Компонент	**Час обробки, мс**	**Пам'ять, МБ**
Препроцесинг	0.15	12

Екстракція ознак	0.35	28
Автоенкодер	0.45	156
Ансамблевий класифікатор	1.20	342
Модуль динамічних порогів	0.15	8
Загалом	2.30	546

Як показано на таблиці 3.6, загальний час обробки одного пакета становить 2.30 мс, що забезпечує теоретичну пропускну здатність понад 430 пакетів на секунду. Споживання пам'яті (546 МБ) є прийнятним для сучасних серверних систем.

Порівняння розробленої системи з результатами, опублікованими в наукових статтях, представлено в таблиці 3.7.

Таблиця 3.7 — Порівняння з результатами літературних джерел

Автор, рік	**Метод**	**Датасет**	**Accuracy**	**F1-score**
Vinayakumar et al. (2019)	CNN-LSTM	CIC-IDS2017	0.9612	0.9534
Shone et al. (2018)	NDAE	NSL-KDD	0.9523	0.9456
Kim et al. (2020)	Ensemble RF+SVM	UNSW-NB15	0.9678	0.9589
Ahmad et al. (2021)	DL-IDS	CIC-IDS2017	0.9712	0.9645
AEGIS IDS (2025)	Гібридний + DTT	CIC-IDS2017	0.9834	0.9772

Як показано на таблиці 3.7, розроблена система AEGIS IDS перевершує опубліковані результати за обома ключовими метриками.

3.4 Висновки до розділу

У третьому розділі представлено практичну реалізацію гібридної системи виявлення мережевих атак AEGIS IDS та проведено комплексне експериментальне дослідження її ефективності.

Обґрунтовано вибір інструментальних засобів: Python 3.11 як основної мови програмування, Flask для веб-інтерфейсу, PyTorch та Scikit-learn для компонентів машинного навчання. Для навчання та тестування використано еталонні набори даних CIC-IDS2017 та UNSW-NB15.

Детально описано реалізацію всіх модулів системи: препроцесора, екстрактора ознак, автоенкодера, ансамблевого класифікатора та модуля динамічних порогів. Представлено інтерфейси системи: головну панель моніторингу, Traffic Interceptor, Rules Engine, Traffic Simulator та Forensic File Analysis.

Експериментальне дослідження підтвердило ефективність розробленої системи: Accuracy=0.9834, F1-score=0.9772, ROC-AUC=0.9923. Система демонструє суттєву перевагу над базовими методами та результатами літературних джерел. Модуль динамічних порогів забезпечує стійкість до концептуального дрейфу: деградація F1-score після 30 днів становить лише 1.16% порівняно з 14.16% для фіксованих порогів.

РОЗДІЛ 4. ЕКОНОМІЧНЕ ОБҐРУНТУВАННЯ РОЗРОБКИ

4.1 Оцінка комерційного потенціалу рішення

Метою проведення комерційного і технологічного аудиту є оцінювання науково-технічного рівня та рівня комерційного потенціалу розробленої гібридної системи виявлення мережевих атак AEGIS IDS.

Для оцінювання технологічної частини та ринкових перспектив залучено трьох експертів з кафедри менеджменту та безпеки інформаційних систем (МБІС) Вінницького національного технічного університету: к.ф.-м.н., доц. Шиян А.А., к.т.н., доц. Крупельницький Л.В., к.т.н., доц., проф. каф. ОТ, Захарченко С.М.

Таблиця 4.1 – Рекомендовані критерії оцінювання науково-технічного рівня і комерційного потенціалу розробки та бальна оцінка

Бали (за 5-ти бальною шкалою)					
	0	1	2	3	4
Технічна здійсненність концепції					
1	Достовірність концепції не підтверджена	Концепція підтверджена експертними висновками	Концепція підтверджена розрахунками	Концепція перевірена на практиці	Перевірено працездатність продукту в реальних умовах
Ринкові переваги (недоліки)					
2	Багато аналогів на малому ринку	Мало аналогів на малому ринку	Кілька аналогів на великому ринку	Один аналог на великому ринку	Продукт не має аналогів на великому ринку
3	Ціна продукту значно вища за ціни аналогів	Ціна продукту дещо вища за ціни аналогів	Ціна продукту приблизно дорівнює цінам аналогів	Ціна продукту дещо нижче за ціни аналогів	Ціна продукту значно нижче за ціни аналогів

4	Технічні та споживчі властивості продукту значно гірші, ніж в	Технічні та споживчі властивості продукту трохи гірші, ніж в аналогів	Технічні та споживчі властивості продукту на рівні аналогів	Технічні та споживчі властивості продукту трохи кращі, ніж в	Технічні та споживчі властивості продукту значно кращі, ніж в
5	Експлуатаційні витрати значно вищі, ніж в аналогів	Експлуатаційні витрати дещо вищі, ніж в аналогів	Експлуатаційні витрати на рівні експлуатаційних витрат аналогів	Експлуатаційні витрати трохи нижчі, ніж в аналогів	Експлуатаційні витрати значно нижчі, ніж в аналогів
6	Ринок малий і не має позитивної динаміки	Ринок малий, але має позитивну динаміку	Середній ринок з позитивною динамікою	Великий стабільний ринок	Великий ринок з позитивною динамікою
7	Активна конкуренція великих компаній на	Активна конкуренція	Помірна конкуренція	Незначна конкуренція	Конкурентів немає
	Практична здійсненність				
8	Відсутні фахівці як з технічної, так і з комерційної реалізації ідеї	Необхідно наймати фахівців або витрачати значні кошти та час на навчання наявних фахівців	Необхідне незначне навчання фахівців та збільшення їх штату	Необхідне незначне навчання фахівців	Є фахівці з питань як з технічної, так і з комерційної реалізації ідеї
9	Потрібні значні фінансові ресурси, які відсутні. Джерела	Потрібні незначні фінансові ресурси. Джерела	Потрібні значні фінансові ресурси. Джерела фінансування є	Потрібні незначні фінансові ресурси. Джерела фінансування є	Не потребує додаткового фінансування

	фінансування ідеї відсутні	фінансування відсутні			
10	Необхідна розробка нових матеріалів	Потрібні матеріали, що використовуються у військово промисловому комплексі	Потрібні дорогі матеріали	Потрібні досяжні та дешеві матеріали	Всі матеріали для реалізації ідеї відомі та давно використовуються у виробництві
11	Термін реалізації ідеї більший за 10 років	Термін реалізації ідеї більший за 5 років. Термін окупності інвестицій більше 10-ти років	Термін реалізації ідеї від 3-х до 5-ти років. Термін окупності інвестицій більше 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій від 3-х до 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій менше 3-х років
12	Необхідна роз- робка регламе- нтних докумен- тів та отриман- ня великої кіль- кості дозвіль- них документів на виробництво та реалізацію продукту	Необхідно отримання ве- ликої кількості дозвільних до- кументів на виробництво та реалізацію про- дукту, що пот- ребує значних коштів та часу	Процедура отри- мання дозвільних документів для виробництва та реалізації проду- кту потребує не- значних коштів та часу	Необхідно тіль- ки повідомлен- ня відповідним органам про виробництво та реалізацію про- дукту	Відсутні будь- які регламентні обмеження на виробництво та реалізацію про- дукту

Таблиця 4.2 – Рівні комерційного потенціалу розробки

Середньоарифметична сума балів СБ, розрахована на основі висновків експертів	**Рівень комерційного потенціалу розробки**
0-10	Низький
11-20	Нижче середнього
21-30	Середній
31-40	Вище середнього
41-48	Високий

Таблиця 4.3 – Результати оцінювання науково-технічного рівня і комерційного потенціалу розробки

Критерії	Прізвище, ініціали, посада експерта		
	Шиян А.А.	Крупельницький Л.В	Захарченко С.М.
	Бали, виставлені експертами:		
1	3	3	3
2	2	2	3
3	3	3	2
4	3	2	3
5	2	3	2
6	3	3	3
7	2	2	2
8	2	3	3
9	3	3	3
10	4	3	3

11	3	3	3
12	3	3	2
Сума балів	33	33	32
Середньоарифметична сума балів	$\overline{\text{СБ}} = \frac{\sum_1^3 \text{СБ}_i}{3} = \frac{33+33+32}{3} = 33$		

Середнє арифметичне значення балів, отриманих у результаті експертного оцінювання, становить 33, що відповідно до таблиці 4.2 характеризує комерційний потенціал розробки як вищий за середній рівень.

4.2 Прогноз витрат на виконання науково-дослідної роботи

Витрати, що виникають під час виконання науково-дослідної роботи, поділяються за такими основними категоріями: оплата праці персоналу, нарахування на заробітну плату, використання матеріалів, палива та енергії для наукових і виробничих потреб, витрати на відрядження, придбання програмного забезпечення, інші поточні витрати та накладні видатки.

У розробці брали участь магістрант (інженер-програміст) та науковий керівник. Кількість робочих днів інженера становить 50 днів. Науковий керівник залучався для консультацій (10% від часу розробника, тобто 5 днів). Приймаємо місячний оклад інженера-програміста – 20 000 грн, наукового керівника – 25 000 грн. Середня кількість робочих днів у місяці – 21.

Розмір основної заробітної плати кожного учасника дослідження обчислюється за формулою:

$$З_0 = M \cdot \frac{t}{T_p} \quad (4.1)$$

де М – місячний оклад працівника (інженера, програміста, дослідника тощо), грн;

T_p – кількість робочих днів у місяці, зазвичай у межах 21–23;

t – кількість днів, фактично відпрацьованих фахівцем у межах виконання НДР.

Таблиця 4.4 – Витрати на заробітну плату дослідників

Найменування посади	Місячний оклад, грн	Оплата за робочий день, грн	Кількість днів роботи	Витрати на заробітну плату, грн
Науковий керівник	25 000	1190,48	5	5 952,40
Інженер-програміст	20 000	952,38	50	47 619,00
Всього				53 571,40

Додаткова оплата праці для всіх учасників проєкту, залучених до створення програмного продукту, визначається у розмірі 10 % від суми їхньої основної заробітної плати.

Розрахунок здійснюється за формулою:

$$З_{дод} = З_{осн} \cdot 0.10 \qquad (4.2)$$

де $З_{дод}$ – сума додаткової заробітної плати, грн;

$З_{осн}$ – основна заробітна плата, грн.

У межах даного проєкту, за умови що основна заробітна плата становить $З_{осн} = 53\,571,40$ грн (див. табл. 4.3), розмір додаткової заробітної плати дорівнюватиме:

$$З_{дод} = 53\,571,40 \cdot 0,10 = 5\,357,14 \text{ грн}$$

Нарахування на заробітну плату співробітників, залучених до виконання цього етапу дослідження, визначаються за формулою:

$$Н_{зп} = (З_{осн} + З_{дод)} \cdot К_{єсв} \quad (4.3)$$

де $Н_{зп}$ – норма нарахувань на заробітну плату, грн;

$З_{осн}$ – основна заробітна плата працівників, грн;

$З_{дод}$ – додаткова заробітна плата, грн;

$К_{єсв}$ – ставка єдиного соціального внеску, %.

Оскільки проєкт реалізується в межах бюджетної сфери, ставка єдиного соціального внеску встановлена на рівні 22 %. Для нашого випадку розрахунок проводиться таким чином:

$$Н_{зп} = (53\,571,40 + 5\,357,14) \cdot 0,22 = 12\,964,28 \text{ грн}$$

Вартість матеріальних компонентів і комплектуючих, що застосовуються під час підготовки та проведення науково-дослідної роботи, визначається відповідно до їхнього переліку за такою формулою:

$$В = \sum_{i=1}^{n} Н_i \cdot Ц_i \cdot К_i \quad (4.4)$$

де $Н_i$ – кількість комплектуючих і-го виду, шт.;

$Ц_i$ – покупна ціна комплектуючих і-го найменування, грн.;

$К_i$ – коефіцієнт транспортних витрат (1,1…1,15).

Для розробки програмного продукту використані такі комплектуючі та витрати на них:

- Папір – 1 шт., 250 грн
- Канцелярське приладдя – 1 шт., 300 грн
- Флешка – 1 шт., 1500 грн
- Блокнот – 1 шт., 200 грн

Загальна вартість витрачених матеріалів становить 2250 грн. З урахуванням коефіцієнта транспортування К = 1,1 – 2475 грн.

Програмне забезпечення, використане під час виконання наукового дослідження, охоплює витрати, пов’язані з експлуатацією спеціалізованих інструментів, необхідних для розроблення, тестування та оцінки ефективності створеної системи.

Амортизаційні відрахування стосуються обладнання, комп’ютерної техніки та приміщень, що використовувались у процесі виконання роботи. Розрахунок таких відрахувань здійснюється для кожного виду ресурсів за формулою:

$$\mathrm{А_{обл}} = \frac{\mathrm{Ц_б}}{\mathrm{Т_в}} \cdot \frac{t_{\mathrm{вик}}}{12} \qquad (4.5)$$

де $Ц_б$ – балансова вартість обладнання, програмних засобів, приміщень тощо, які використовувались для проведення досліджень, грн;

$t_{вик}$ – термін використання обладнання, програмних засобів, приміщень під час досліджень, місяців;

$T_в$ – строк корисного використання обладнання, програмних засобів, приміщень тощо, років.

Відповідно до пункту 137.3.3 Податкового кодексу України амортизаційні відрахування застосовуються до основних засобів із вартістю понад 2500 грн. Під час розробки використовувався персональний комп’ютер із балансною вартістю 30000 грн. Середній строк служби комп’ютера прийнято 2 роки (24 місяців), при цьому для виконання даного етапу роботи комп’ютер експлуатувався протягом 2 місяців.

$$\mathrm{А_{обл}} = \frac{30000}{24} = 1250$$

Сума амортизаційних відрахувань для обладнання становить 1250 грн. Категорія «Паливо та енергія для науково-виробничих цілей» охоплює витрати на всі види енергії, що безпосередньо використовуються для технологічних операцій під час проведення наукових досліджень.

Витрати на електроенергію розраховуються за формулою:

$$B_{\text{ен}} = \sum_{i=1}^{n} \frac{W_{yt} \cdot t_i \cdot \text{Ц}_\text{е} \cdot K_{\text{впі}}}{\eta_i} \quad (4.6)$$

де W_{yt} – номінальна потужність обладнання на конкретному етапі роботи, кВт;

$t_{\text{і}}$ – час роботи обладнання під час досліджень, год;

$\text{Ц}_\text{е}$ – ціна 1 кВт·год електроенергії, грн;

$\text{К}_{\text{впі}}$ – коефіцієнт використання потужності (< 1);

$\eta_{\text{і}}$ – ККД обладнання (< 1).

Для реалізації розробки використовувався персональний комп'ютер з потужністю 0,35 кВт, що працював протягом 400 години. Вартість 1 кВт·год електроенергії прийнята рівною 12,5 грн, коефіцієнт використання потужності становить 0,8, а коефіцієнт корисної дії обладнання – 0,9.

$$B_{\text{ен}} = \frac{0{,}35 \cdot 400 \cdot 12{,}5 \cdot 0{,}8}{0{,}9} = 1\,555{,}55 \text{ грн}$$

Витрати на службові відрядження та роботи, виконані сторонніми організаціями чи підприємствами, у рамках даного дослідження не враховувалися, оскільки таких робіт не проводилося.

Накладні (загальновиробничі) витрати охоплюють управління розробкою, утримання приміщень, комунальні послуги тощо. У даному дослідженні накладні витрати прийнято рівними 100 % основної заробітної плати розробників:

$$B_{\text{нзв}} = (З_{\text{о}} + З_{\text{р}}) \cdot \frac{Н_{\text{нзв}}}{100\%} \quad (4.7)$$

де $Н_{\text{нзв}}$ – норма нарахування за статтею «Інші витрати».

$$B_{\text{нзв}} = (53\,571,40) \cdot \frac{100\%}{100\%} = 53\,571,40\text{грн}$$

Сума всіх попередніх статей витрат дає витрати, які безпосередньо стосуються даного розділу МКР:

$$B = 53\,571,40 + 5357,14 + 12\,964,28 + 2475 + 1250 + 1555,55 + 53\,571,40 = 130\,744,77 \text{ грн}$$

Оцінка загальної суми витрат на реалізацію та впровадження результатів магістерської науково-дослідної роботи проводиться за співвідношенням:

$$ЗВ = B \cdot \beta \quad (4.8)$$

де β – коефіцієнт, що відображає етап виконання наукових досліджень.

Для поточного проєкту, який перебуває на стадії НДР, приймаємо β = 0,9. Таким чином, загальні витрати складають:

$$ЗВ = 130\,744,77 \cdot 0,9 = 117\,670,29 \text{ грн}$$

4.3 Розрахунок економічної ефективності впровадження

У даному підрозділі здійснюється кількісний прогноз очікуваного економічного ефекту від упровадження результатів виконаної науково-дослідної роботи.

Зростання чистого прибутку підприємства внаслідок впровадження розробки визначається за формулою:

$$\Delta П_i = \sum_1^n (\Delta Ц_o \cdot N \cdot Ц_o \cdot \Delta N) \cdot \lambda \cdot \rho (1 - \frac{\upsilon}{100}) \qquad (4.9)$$

де $\Delta Ц_0$ – приріст основного оціночного показника від застосування результатів розробки у конкретному році;

N – базовий кількісний показник діяльності підприємства до впровадження розробки;

ΔN – зміна основного кількісного показника після впровадження розробки;

$Ц_o$ – основний оціночний показник діяльності підприємства після впровадження розробки;

n – період у роках, протягом якого очікується отримання позитивного ефекту від впровадження;

λ – коефіцієнт, сплати податку на додану вартість. Коефіцієнт λ = 0,8333.

ρ – коефіцієнт рентабельності продукту. ρ = 0,4;

υ – ставка податку на прибуток з урахуванням військового збору (2025 рік υ = 23%).

Припустимо, що ціна за програмний продукт зросте на 5000 грн, а обсяг реалізації збільшується: у перший рік – на 40 одиниць, у другий – на 45 одиниць, у третій – на 50 одиниць. До впровадження розробки реалізовувалась 1 одиниця продукції за ціною 23 000 грн. На основі цих даних розраховується прибуток підприємства за три роки.

$$\Delta П_1 = [5000 \cdot 1 + 28000 \cdot 40] \cdot 0{,}833 \cdot 0{,}25 \cdot \left(1 - \frac{23}{100}\right) = 180\ 397 \text{ грн}$$

$$\Delta П_2 = [5000 \cdot 1 + 28000 \cdot 85] \cdot 0{,}833 \cdot 0{,}25 \cdot \left(1 - \frac{23}{100}\right) = 382\ 441 \text{ грн}$$

$$\Delta П_3 = [5000 \cdot 1 + 28000 \cdot 135] \cdot 0{,}833 \cdot 0{,}25 \cdot \left(1 - \frac{23}{100}\right) = 606\ 934 \text{ грн}$$

4.4 Оцінка окупності інвестицій

Для оцінки доцільності вкладення коштів у розробку програмного забезпечення потенційним інвестором використовуються такі критерії: абсолютна та відносна рентабельність інвестицій, а також період їх повернення.

На початковому етапі проводиться розрахунок величини стартових інвестицій PV, які необхідно вкласти для впровадження та комерційного використання розробленої системи:

$$PV = ЗВ \cdot k \qquad (4.10)$$

де ЗВ = 117 670, 29 грн — витрати на виконання науково-дослідної роботи;

k — коефіцієнт, що враховує додаткові витрати інвестора на впровадження та комерціалізацію системи (підготовка приміщень, навчання персоналу, інтеграція з існуючими PLC-системами, маркетингові заходи). Приймається k = 2.

Тоді початкові інвестиції становитимуть:

$$PV = 117\ 670, 29\ \cdot 2 \approx 290\ 194 \text{ грн}$$

Абсолютну ефективність вкладених інвестицій $Е_{абс}$ обчислюємо за формулою:

$$Е_{абс} = ПП - PV \quad (4.11)$$

де ПП — приведена вартість усіх чистих прибутків, які підприємство отримає в результаті впровадження системи виявлення аномалій у PLC, грн; PV= 271 355,4 грн — початкові інвестиції, розраховані раніше.

Приведена вартість чистих прибутків визначається таким чином:

$$ПП = \sum_1^T \frac{\Delta П_i}{(1+\tau)^t} \quad (4.12)$$

де ΔПП – приріст чистого прибутку у кожному році, протягом якого спостерігається ефект від виконаної та впровадженої НДДКР, грн;

Т – період, протягом якого проявляються результати впровадженої НДДКР, роки;

τ – ставка дисконтування, яку можна прийняти рівною прогнозованому щорічному рівню інфляції; для України цей показник складає 0,2;

t – номер року в розрахунковому періоді.

$$ПП = \frac{180\ 397}{(1+0{,}2)^1} + \frac{382\ 441}{(1+0{,}2)^2} + \frac{606\ 934}{(1+0{,}2)^3} = 767\ 150 \text{ грн}$$

Тепер можна розрахувати абсолютну ефективність інвестицій:

$$Е_{абс} = 767\ 150 - 290\ 194 = 476\ 956 \text{ грн}$$

Оскільки $Е_{абс} > 0$ інвестування коштів у виконання та впровадження результатів НДДКР визнається доцільним.

Далі розраховується відносна (річна) ефективність вкладених інвестицій E_B за формулою:

$$E_B = \sqrt[T_Ж]{1 + \frac{E_{абс}}{PV}} - 1 \quad (4.13)$$

Де $T_Ж$ – життєвий цикл наукової розробки, роки.

$$E_B = \sqrt[3]{1 + \frac{476956}{290194}} - 1 = 0{,}38 = 38\%$$

Мінімальну ставку дисконтування можна розрахувати за загальною формулою:

$$\tau = d + f \quad (4.14)$$

е d – середньозважений відсоток за депозитними операціями у комерційних банках, який для України у 2025 році становить 0,14–0,2.

f – коефіцієнт, що відображає рівень ризику інвестицій; зазвичай приймається в межах 0,05–0,1.

$$\tau_{min} = 0.14 + 0.1 = 0.24$$

Так як $E_в > \tau_{min}$ то інвестор може бути зацікавлений у фінансуванні даної наукової розробки.

Визначимо термін окупності інвестицій, вкладених у реалізацію наукового проєкту, за наступною формулою:

$$T_{ок} = \frac{1}{E_B} \quad (4.15)$$

$$T_{\text{ок}} = \frac{1}{0.38} = 2{,}63$$

$T_{\text{ок}} \leq 3$ років, то це свідчить про економічну ефективність впровадження науково-технічної розробки її розробником (замовником).

4.5 Висновки до розділу

У цьому розділі проведено економічне обґрунтування доцільності розробки гібридної системи виявлення мережевих атак AEGIS IDS. Загальна вартість виконання НДР становить 145 097 грн, а з урахуванням коефіцієнта впровадження k=2 початкові інвестиції дорівнюють 290 194 грн.

Приведена вартість чистих прибутків за три роки становить 767 150 грн, що забезпечує абсолютну ефективність інвестицій на рівні 476 956 грн. Відносна ефективність перевищує мінімальну ставку дисконтування, а термін окупності — 2,63 року, що є менше нормативного значення.

Отже, впровадження розробленої системи є економічно доцільним, має високий рівень ефективності та може бути успішно комерціалізоване в умовах сучасного ринку кібербезпеки.

ВИСНОВКИ

У кваліфікаційній роботі вирішено актуальне наукове завдання удосконалення системи виявлення мережевих атак шляхом застосування гібридного підходу машинного навчання з динамічним налаштуванням порогових значень. Запропонований підхід забезпечив подолання обмежень традиційних систем виявлення, які базуються на жорстко фіксованих порогах та окремих моделях аналізу трафіку. Виконаний теоретичний аналіз сучасних методів IDS дав змогу встановити ключові недоліки існуючих рішень, пов’язані з низькою адаптивністю до змінних характеристик трафіку, нестабільністю показників точності за умов змішаних атак та недостатньою стійкістю до раніше невідомих загроз.

Розроблено гібридну модель, що поєднує автоенкодер для виявлення аномалій і ансамблеву класифікацію на основі Random Forest та CNN-1D, що дозволило забезпечити багаторівневу оцінку ризику та підвищити достовірність прийняття рішень. Особливу увагу приділено модулю динамічного налаштування порогів, який формує адаптивні критерії оцінки ступеня аномальності трафіку з урахуванням статистичних властивостей потоку. Така конструкція системи дає змогу забезпечувати стійку роботу в умовах високої варіативності мережевого середовища та значно зменшувати кількість хибних спрацьовувань.

Практична реалізація системи підтвердила ефективність запропонованого підходу. Експериментальні дослідження на вибірках реального мережевого трафіку продемонстрували зростання точності, повноти та F1-міри порівняно з класичними алгоритмами IDS, а також кращу здатність до узагальнення за наявності невідомих атак. Аналіз отриманих кривих ROC та значень AUC довів підвищену результативність гібридної моделі, а механізм динамічних порогів продемонстрував стабільність за умов зміни фонового навантаження і появи нових типів загроз.

Економічне обґрунтування показало, що впровадження розробленої системи є доцільним з погляду витрат на розробку та експлуатацію, оскільки

завдяки зменшенню обсягу інцидентів, скороченню часу реагування та зниженню навантаження на аналітиків кібербезпеки забезпечується швидке повернення інвестицій.

Отже, сформована у роботі гібридна система виявлення мережевих атак створює науково та практично обґрунтовану основу для подальшого розвитку високоточних інтелектуальних систем кіберзахисту та може бути використана як фундамент для розширення функціоналу IDS і побудови комплексних систем активної протидії атакам.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Abhishek N. V., Tandon A., Lim T. J., Sikdar B. A GLRT-Based mechanism for detecting Relay Misbehavior in Clustered IoT Networks. IEEE Transactions on Information Forensics and Security. 2020. Vol. 15. P. 435–446. DOI: https://doi.org/10.1109/TIFS.2019.2922262
2. Ahmad Z., Shahid Khan A., Wai Shiang C., Abdullah J., Ahmad F. Network intrusion detection system: a systematic study of machine learning and deep learning approaches. Transactions on Emerging Telecommunications Technologies. 2021. Vol. 32. e4150. DOI: https://doi.org/10.1002/ett.4150
3. Akgun D. A new DDoS attacks intrusion detection model based on deep learning for cybersecurity. Computers and Security. 2023. Vol. 118. 102748. DOI: https://doi.org/10.1016/j.cose.2022.102748
4. Al S., Dener M. Stl-hdl: A new hybrid network intrusion detection system for imbalanced dataset on big data environment. Computers & Security. 2021. Vol. 110. 102435. DOI: https://doi.org/10.1016/j.cose.2021.102435
5. Altulaihan E., Almaiah M. A., Aljughaiman A. Anomaly detection ids for detecting dos attacks in iot networks based on machine learning algorithms. Sensors. 2024. Vol. 24, No. 2. 713. DOI: https://doi.org/10.3390/s24020713
6. Al-Turjman F., Ever E., Zahmatkesh H. Small cells in the forthcoming 5G/IoT: Traffic modelling and deployment overview. IEEE Communications Surveys & Tutorials. 2019. Vol. 21. P. 28–65. DOI: https://doi.org/10.1109/COMST.2018.2864779
7. Amaouche S., Guezzaz A., Benkirane S., Azrour M. Ids-xgbfs: a smart intrusion detection system using xgboostwith recent feature selection for vanet safety. Cluster Computing. 2024. Vol. 27. P. 3521–3535. DOI: https://doi.org/10.1007/s10586-023-04157-w
8. Bukhari S. M. S., Zafar M. H., Abou Houran M., Moosavi S. K. R., Mansoor M., Muaaz M., Sanfilippo F. Secure and privacy-preserving intrusion detection in wireless sensor networks: Federated learning with scnn-bi-lstm for enhanced reliability. Ad Hoc Networks. 2024. Vol. 155. 103407. DOI:

https://doi.org/10.1016/j.adhoc.2024.103407

9. Chakravarthi B., Ng S. C., Ezilarasan M., Leung M. F. Eeg-based emotion recognition using hybrid cnn and lstm classification. Frontiers in Computational Neuroscience. 2022. Vol. 16. 1019776. DOI: https://doi.org/10.3389/fncom.2022.1019776
10. Chkirbene Z., Erbad A., Hamila R., Mohamed A., Guizani M., Hamdi M. Tidcs: A dynamic intrusion detection and classification system based feature selection. IEEE Access. 2020. Vol. 8. P. 95864–95877. DOI: https://doi.org/10.1109/ACCESS.2020.3004325
11. Deebak B. D., Hwang S. O. Healthcare Applications Using Blockchain With a Cloud-Assisted Decentralized Privacy-Preserving Framework. IEEE Transactions on Mobile Computing. 2024. Vol. 23, No. 5. P. 5897–5916. DOI: https://doi.org/10.1109/TMC.2023.3315510
12. Dlamini G., Fahim M. Dgm: a data generative model to improve minority class presence in anomaly detection domain. Neural Computing and Applications. 2021. Vol. 33. P. 13635–13646. DOI: https://doi.org/10.1007/s00521-021-05993-w
13. Dong R. H., Li X. Y., Zhang Q. Y., Yuan H. Network intrusion detection model based on multivariate correlation analysis-long short-time memory network. IET Information Security. 2020. Vol. 14, No. 2. P. 166–174. DOI: https://doi.org/10.1049/iet-ifs.2019.0313
14. Faysal J. Al, Mostafa S. T., Tamanna J. S., Mumenin K. M., Arifin M. M., Awal M. A., Shome A., Mostafa S. S. XGB-RF: A Hybrid Machine Learning Approach for IoT Intrusion Detection. Telecom. 2022. Vol. 3. P. 52–69. DOI: https://doi.org/10.3390/telecom3010003
15. Ferrag M. A., Friha O., Hamouda D., Maglaras L., Janicke H. Edge-iiotset: A new comprehensive realistic cyber security dataset of iot and iiot applications for centralized and federated learning. IEEE Access. 2022. Vol. 10. P. 40281–40306. DOI: https://doi.org/10.1109/ACCESS.2022.3165809
16. Gupta B. B., Quamara M. An overview of internet of things (IoT): Architectural

aspects, challenges, and protocols. Concurrency and Computation: Practice and Experience. 2020. Vol. 32. P. 1–24. DOI: https://doi.org/10.1002/cpe.4946

17. Halbouni A., Gunawan T. S., Habaebi M. H., Halbouni M., Kartiwi M., Ahmad R. Cnn-lstm: hybrid deep neural network for network intrusion detection system. IEEE Access. 2022. Vol. 10. P. 99837–99849. DOI: https://doi.org/10.1109/ACCESS.2022.3206425
18. Hanafi A. V., Ghaffari A., Rezaei H., Valipour A., Arasteh B. Intrusion detection in internet of things using improved binary golden jackal optimization algorithm and lstm. Cluster Computing. 2024. Vol. 27, No. 3. P. 2673–2690. DOI: https://doi.org/10.1007/s10586-023-04096-6
19. Hasan T., Malik J., Bibi I., Khan W. U., Al-Wesabi F. N., Dev K., Huang G. Securing Industrial Internet of things against Botnet Attacks using Hybrid Deep Learning Approach. IEEE Transactions on Network Science and Engineering. 2022. DOI: https://doi.org/10.1109/TNSE.2022.3168533
20. Hassan S. R., Rehman A. U., Alsharabi N., Arain S., Quddus A., Hamam H. Design of load-aware resource allocation or heterogeneous fog computing systems. PeerJ Computer Science. 2024. Vol. 10. e1986. DOI: https://doi.org/10.7717/peerj-cs.1986
21. Heidari A., Jafari Navimipour N., Dag H., Unal M. Deepfake detection using deep learning methods: A systematic and comprehensive review. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery. 2024. Vol. 14. e1520. DOI: https://doi.org/10.1002/widm.1520
22. Heidari A., Jafari Navimipour N., Unal M., Zhang G. Machine learning applications in internet-of-drones: Systematic review, recent deployments, and open issues. ACM Computing Surveys. 2023. Vol. 55, No. 12. P. 1–45. DOI: https://doi.org/10.1145/3571728
23. Heidari A., Navimipour N. J., Unal M. A secure intrusion detection platform using blockchain and radial basis function neural networks for internet of drones. IEEE Internet of Things Journal. 2023. Vol. 10. P. 8445–8454. DOI: https://doi.org/10.1109/JIOT.2023.3251693

24. Henry A., Gautam S., Khanna S., Rabie K., Shongwe T., Bhattacharya P., Sharma B., Chowdhury S. Composition of hybrid deep learning model and feature optimization for intrusion detection system. Sensors. 2023. Vol. 23, No. 2. 890. DOI: https://doi.org/10.3390/s23020890
25. Hnamte V., Hussain J. Dcnnbilstm: An efficient hybrid deep learning-based intrusion detection system. Telematics and Informatics Reports. 2023. Vol. 10. 100053. DOI: https://doi.org/10.1016/j.teler.2023.100053
26. Hnamte V., Nhung-Nguyen H., Hussain J., Hwa-Kim Y. A novel two-stage deep learning model for network intrusion detection: Lstm-ae. IEEE Access. 2023. Vol. 11. P. 37131–37148. DOI: https://doi.org/10.1109/ACCESS.2023.3266979
27. Ibrahim M., Elhafiz R. Modeling an intrusion detection using recurrent neural networks. Journal of Engineering Research. 2023. Vol. 11. 100013. DOI: https://doi.org/10.1016/j.jer.2023.100013
28. Ieracitano C., Adeel A., Morabito F. C., Hussain A. A novel statistical analysis and autoencoder driven intelligent intrusion detection approach. Neurocomputing. 2020. Vol. 387. P. 51–62. DOI: https://doi.org/10.1016/j.neucom.2019.11.016
29. Injadat M., Moubayed A., Nassif A. B., Shami A. Multi-stage optimized machine learning framework for network intrusion detection. IEEE Transactions on Network and Service Management. 2020. Vol. 18, No. 2. P. 1803–1816. DOI: https://doi.org/10.1109/TNSM.2020.3014929
30. Javeed D., Gao T., Khan M. T. Sdn-enabled hybrid dl-driven framework for the detection of emerging cyber threats in iot. Electronics. 2021. Vol. 10. P. 1–16. DOI: https://doi.org/10.3390/electronics10080918
31. Junwon K., Jiho S., Ki-Woong P., Jung T. S. Improving Method of Anomaly Detection Performance for Industrial IoT Environment. Computers, Materials & Continua. 2022. Vol. 72, No. 3. P. 5377–5394. DOI: https://doi.org/10.32604/cmc.2022.026619
32. Kasongo S. M., Sun Y. A deep learning method with wrapper based feature extraction for wireless intrusion detection system. Computers & Security. 2020.

Vol. 92. 101752. DOI: https://doi.org/10.1016/j.cose.2019.101752

33. Khan M. A. HCRNNIDS: Hybrid Convolutional recurrent neural. Multidisciplinary Digital Publishing Institute. 2021.
34. Khan Z. I., Afzal M. M., Shamsi K. N. A comprehensive study on cic-ids2017 dataset for intrusion detection systems. International Research Journal of Advanced Engineering Hub. 2024. Vol. 2, No. 02. P. 254–260. DOI: https://doi.org/10.47857/irjaeh.2024.v02i02.0175
35. Kim T., Pak W. Early detection of network intrusions using a GAN-based one-class classifier. IEEE Access. 2022. Vol. 10. P. 119357–119367. DOI: https://doi.org/10.1109/ACCESS.2022.3221783
36. Kouicem D. E., Bouabdallah A., Lakhlef H. Internet of things security: A top-down survey. Computer Networks. 2018. Vol. 141. P. 199–221. DOI: https://doi.org/10.1016/j.comnet.2018.03.012
37. Kumar R., Malik A., Ranga V. An intellectual intrusion detection system using hybrid Hunger games Search and Remora optimization algorithm for IoT wireless networks. Knowledge-Based Systems. 2022. Vol. 256. 109762. DOI: https://doi.org/10.1016/j.knosys.2022.109762
38. Lilhore U. K., Manoharan P., Simaiya S., Alroobaea R., Alsafyani M., Baqasah A. M., Dalal S., Sharma A., Raahemifar K. Hidm: Hybrid intrusion detection model for industry 4.0 networks using an optimized cnn-lstm with transfer learning. Sensors. 2023. Vol. 23, No. 18. 7856. DOI: https://doi.org/10.3390/s23187856
39. Liu S., Lin G., Han Q. L., Wen S., Zhang J., Xiang Y. DeepBalance: Deep-learning and fuzzy oversampling for vulnerability detection. IEEE Transactions on Fuzzy Systems. 2020. Vol. 28. P. 1329–1343. DOI: https://doi.org/10.1109/TFUZZ.2019.2958558
40. Maseer Z. K., Yusof R., Bahaman N., Mostafa S. A., Foozy C. F. M. Benchmarking of machine learning for anomaly based intrusion detection systems in the cicids2017 dataset. IEEE Access. 2021. Vol. 9. P. 22351–22370. DOI: https://doi.org/10.1109/ACCESS.2021.3056614

41. Mebawondu J. O., Alowolodu O. D., Mebawondu J. O., Adetunmbi A. O. Network intrusion detection system using supervised learning paradigm. Scientific African. 2020. Vol. 9. e00497. DOI: https://doi.org/10.1016/j.sciaf.2020.e00497
42. Mehmood M., Javed T., Nebhen J., Abbas S., Abid R., Bojja G. R., Rizwan M. A hybrid approach for network intrusion detection. CMC-Computers Materials & Continua. 2022. Vol. 70. P. 91–107. DOI: https://doi.org/10.32604/cmc.2022.019050
43. Mohamad Noor M. binti, Hassan W. H. Current research on internet of things (IoT) security: A survey. Computer Networks. 2019. Vol. 148. P. 283–294. DOI: https://doi.org/10.1016/j.comnet.2018.11.025
44. Mohamed D., Ismael O. Enhancement of an iot hybrid intrusion detection system based on fog-to-cloud computing. Journal of Cloud Computing. 2023. Vol. 12, No. 1. 41. DOI: https://doi.org/10.1186/s13677-023-00408-3
45. Molina-Coronado B., Mori U., Mendiburu A., Miguel-Alonso J. Survey of network intrusion detection methods from the perspective of the knowledge discovery in databases process. IEEE Transactions on Network and Service Management. 2020. Vol. 17, No. 4. P. 2451–2479. DOI: https://doi.org/10.1109/TNSM.2020.3016246
46. Moustafa N., Choo K. K. R., Radwan I., Camtepe S. Outlier Dirichlet mixture mechanism: Adversarial statistical learning for Anomaly Detection in the fog. IEEE Transactions on Information Forensics and Security. 2019. Vol. 14. P. 1975–1987. DOI: https://doi.org/10.1109/TIFS.2018.2890808
47. Muhammad S. F., Sagheer A., Atta-ur-Rahman, Kiran S., Muhammad A. K., Amir M. A Fused Machine Learning Approach for Intrusion Detection System. Computers, Materials & Continua. 2022. Vol. 74, No. 2. P. 2607–2623. DOI: https://doi.org/10.32604/cmc.2023.032617
48. Mustapha A. DDoS attack detection using machine learning algorithms. Computers and Security. 2023. Vol. 127. 103117. DOI: https://doi.org/10.1016/j.cose.2023.103117

49. Naseer S., Saleem Y., Khalid S., Bashir M. K., Han J., Iqbal M. M., Han K. Enhanced network anomaly detection based on deep neural networks. IEEE Access. 2018. Vol. 6. P. 48231–48246. DOI: https://doi.org/10.1109/ACCESS.2018.2863036
50. Nazir A., He J., Zhu N., Qureshi S. S., Qureshi S. U., Ullah F., Wajahat A., Pathan M. S. A deep learning-based novel hybrid cnn-lstm architecture for efficient detection of threats in the iot ecosystem. Ain Shams Engineering Journal. 2024. Vol. 15. 102777. DOI: https://doi.org/10.1016/j.asej.2024.102777
51. Nedeljkovic D., Jakovljevic Z. Cnn based method for the development of cyber-attacks detection algorithms in industrial control systems. Computers & Security. 2022. Vol. 114. 102585. DOI: https://doi.org/10.1016/j.cose.2021.102585
52. Ramkumar D., Annadurai C., Nelson I. Iris-based continuous authentication in mobile ad hoc network. Concurrency and Computation: Practice and Experience. 2022. Vol. 34. P. 4–8. DOI: https://doi.org/10.1002/cpe.5542
53. Ramkumar D., Annadurai C., Nirmaladevi K. Continuous authentication consoles in mobile ad hoc network (MANET). Cluster Computing. 2019. Vol. 22. P. 7777–7786. DOI: https://doi.org/10.1007/s10586-017-1386-2
54. Roy S., Li J., Choi B. J., Bai Y. A lightweight supervised intrusion detection mechanism for IoT networks. Future Generation Computer Systems. 2022. Vol. 127. P. 276–285. DOI: https://doi.org/10.1016/j.future.2021.09.027
55. Saba T., Rehman A., Sadad T., Kolivand H., Bahaj S. A. Anomaly-based intrusion detection system for IoT networks through deep learning model. Computers and Electrical Engineering. 2022. Vol. 99. 107810. DOI: https://doi.org/10.1016/j.compeleceng.2022.107810
56. Saba T., Sadad T., Rehman A., Mehmood Z., Javaid Q. Intrusion detection system through Advance Machine Learning for the internet of things networks. IT Professional. 2021. Vol. 23. P. 58–64. DOI: https://doi.org/10.1109/MITP.2020.2992710
57. Saif S., Das P., Biswas S., Khari M., Shanmuganathan V. HIIDS: Hybrid intelligent intrusion detection system empowered with machine learning and

metaheuristic algorithms for application in IoT based healthcare. Microprocessors and Microsystems. 2022. 104622. DOI: https://doi.org/10.1016/j.micpro.2022.104622

58. Sattari F., Farooqi A. H., Qadir Z., Raza B., Nazari H., Almutiry M. A Hybrid Deep Learning Approach for Bottleneck Detection in IoT. IEEE Access. 2022. Vol. 10. P. 77039–77053. DOI: https://doi.org/10.1109/ACCESS.2022.3188635
59. Shahriar M. H., Haque N. I., Rahman M. A., Alonso M. G-ids: Generative adversarial networks assisted intrusion detection system. 2020 IEEE 44th Annual Computers, Software, and Applications Conference (COMPSAC). IEEE, 2020. P. 376–385. DOI: https://doi.org/10.1109/COMPSAC48688.2020.0-218
60. Sharif D. M. A new DDoS attacks intrusion detection model based on deep learning for cybersecurity. Computers and Security. 2023. Vol. 135. 103511. DOI: https://doi.org/10.1016/j.cose.2023.103511
61. Shi W. C., Sun H. M. Deepbot: a time-based botnet detection with deep learning. Soft Computing. 2020. Vol. 24. P. 16605–16616. DOI: https://doi.org/10.1007/s00500-020-04952-0
62. Sun P., Liu P., Li Q., Liu C., Lu X., Hao R., Chen J. Dl-ids: Extracting features using cnn-lstm hybrid network for intrusion detection system. Security and Communication Networks. 2020. Vol. 2020. P. 1–11. DOI: https://doi.org/10.1155/2020/8890306
63. Talukder M. A., Hasan K. F., Islam M. M., Uddin M. A., Akhter A., Yousuf M. A., Alharbi F., Moni M. A. A dependable hybrid machine learning model for network intrusion detection. Journal of Information Security and Applications. 2023. Vol. 72. 103405. DOI: https://doi.org/10.1016/j.jisa.2022.103405
64. Turukmane A. V., Devendiran R. M-MultiSVM: An efficient feature selection assisted Network Intrusion Detection System using machine learning. Computer Security. 2023. Vol. 137. 103587. DOI: https://doi.org/10.1016/j.cose.2023.103587
65. Vashishtha L. K., Singh A. P., Chatterjee K. Hidm: A hybrid intrusion detection model for cloud based systems. Wireless Personal Communications. 2023. Vol.

128. P. 2637–2666. DOI: https://doi.org/10.1007/s11277-022-10052-8

66. Wang C., Sun Y., Wang W., Liu H., Wang B. Hybrid intrusion detection system based on combination of random forest and autoencoder. Symmetry. 2023. Vol. 15, No. 3. 568. DOI: https://doi.org/10.3390/sym15030568

67. Wu Y., Wei D., Feng J. Network attacks detection methods based on deep learning techniques: A survey. Security and Communication Networks. 2020. DOI: https://doi.org/10.1155/2020/8872923

68. Yao R., Wang N., Liu Z., Chen P., Sheng X. Intrusion detection system in the advanced metering infrastructure: a cross-layer feature-fusion cnn-lstm-based approach. Sensors. 2021. Vol. 21, No. 2. 626. DOI: https://doi.org/10.3390/s21020626

69. Yilmaz I., Masum R., Siraj A. Addressing imbalanced data problem with generative adversarial network for intrusion detection. 2020 IEEE 21st International Conference on Information Reuse and Integration for Data Science (IRI). IEEE, 2020. P. 25–30. DOI: https://doi.org/10.1109/IRI49571.2020.00012

70. Zainudin A., Ahakonye L. A. C., Akter R., Kim D. S., Lee J. M. An efficient Hybrid-DNN for DDoS detection and classification in Software-defined IIoT networks. IEEE Internet of Things Journal. 2023. Vol. 10. P. 8491–8504. DOI: https://doi.org/10.1109/JIOT.2022.3196942

71. Zhang H., Li Y., Lv Z., Sangaiah A. K., Huang T. A real-time and ubiquitous network attack detection based on deep belief network and support vector machine. IEEE/CAA Journal of Automatica Sinica. 2020. Vol. 7, No. 3. P. 790–799. DOI: https://doi.org/10.1109/JAS.2020.1003099

105

Додаток А. Технічне завдання

Вінницький національний технічний університет

Факультет менеджменту та інформаційної безпеки

Кафедра менеджменту та безпеки інформаційних систем

ЗАТВЕРДЖУЮ

Голова секції "Управління
інформаційною
безпекою" кафедри МБІС
д.т.н., професор

_____ Юрій ЯРЕМЧУК
"14" вересня 2025 р.

ТЕХНІЧНЕ ЗАВДАННЯ

до магістерської кваліфікаційної роботи на тему:

Удосконалення системи виявлення мережевих атак шляхом застосування гібридного підходу машинного навчання з динамічним налаштуванням порогових значень

08-72.МКР.011.00.000.ТЗ

Керівник магістерської кваліфікаційної роботи
к.ф-м.н., доцент каф. МБІС
_____ Шиян А. А.

Вінниця – 2025 р.

1. Найменування та область застосування

Програмний комплекс гібридної системи виявлення мережевих атак «AEGIS IDS» з динамічним налаштуванням порогових значень. Область застосування: захист інформаційно-телекомунікаційних систем підприємств, організацій та установ від несанкціонованого доступу та мережевих вторгнень. Система може використовуватися в корпоративних мережах, центрах обробки даних та IoT-інфраструктурах.

2.Підстава для розробки

Розробка виконується на основі наказу ректора ВНТУ №313 від 24. 09. 2025 р.

3. Мета та призначення розробки

3.1 Мета розробки: підвищення ефективності виявлення мережевих атак шляхом розробки гібридної системи на основі ансамблевих методів машинного навчання та глибоких нейронних мереж з механізмом динамічного налаштування порогових значень.

3.2 Призначення: розроблений програмний засіб призначений для моніторингу мережевого трафіку в реальному часі, виявлення відомих атак та аномалій, автоматичної адаптації порогів детектування до змін профілю трафіку та сповіщення адміністраторів про інциденти безпеки.

4. Джерела розробки

4.1. Ahmad Z., Shahid Khan A. Network intrusion detection system: a systematic study of machine learning and deep learning approaches // Transactions on Emerging Telecommunications Technologies. – 2021. Vol. 32..

4.2. Ferrag M. A. et al. Edge-iiotset: A new comprehensive realistic cyber security dataset of iot and iiot applications // IEEE Access. – 2022. Vol. 10..

4.3. Moustafa N. et al. Outlier Dirichlet mixture mechanism: Adversarial statistical learning for Anomaly Detection in the fog // IEEE Transactions on Information Forensics and Security. – 2019. Vol. 14..

4.4. Injadat M. et al. Multi-stage optimized machine learning framework for network intrusion detection // IEEE Transactions on Network and Service Management. – 2020. Vol. 18..

5. Вимоги до програми

5.1 Вимоги до функціональних характеристик:

5.1.1. Програмний засіб повинен забезпечувати захоплення та попередню обробку мережевих пакетів у реальному часі.

5.1.2. Система має реалізовувати гібридний аналіз трафіку з використанням ансамблю Random Forest та CNN-1D для класифікації атак, а також автоенкодера для виявлення аномалій.

5.1.3. Програмний засіб повинен мати модуль динамічного налаштування порогових значень (Dynamic Threshold Tuner), що адаптується до змін трафіку.

5.1.4. Інтерфейс користувача (Web UI) має забезпечувати візуалізацію статистики загроз, керування сигнатурами та аналіз лог-файлів.

5.1.5. Система повинна підтримувати класифікацію загроз за рівнями: SAFE, LOW, MEDIUM, HIGH, CRITICAL.

5.2 Вимоги до надійності:

5.2.1. Програмний засіб повинен коректно обробляти помилки вхідних даних та забезпечувати безперервну роботу при високому навантаженні.

5.2.2. Забезпечення збереження журналів подій та результатів аналізу в базі даних SQLite (aegis_core.db).

5.2.3. Стійкість до концептуального дрейфу (degradation of performance) не більше 2% протягом місяця експлуатації.

5.3 Вимоги до складу і параметрів технічних засобів:

– Процесор – багатоядерний (аналог Intel Core i5/i7 або вище);

– Оперативна пам’ять – не менше 8 Гб (рекомендовано 16 Гб для навчання моделей);

– Дисковий простір – не менше 10 Гб вільного місця;

– Середовище функціонування – ОС сімейства Windows/Linux, інтерпретатор Python 3.11.

– Необхідні бібліотеки: PyTorch 2.0, Scikit-learn 1.3, Flask 2.3, Scapy 2.5

6. Вимоги до програмної документації

6.1. Інструкція користувача по роботі з веб-інтерфейсом системи AEGIS.

6.2. Вихідний код програми з коментарями.

7. Вимоги до технічного захисту інформації

7.1. Забезпечення цілісності бази даних сигнатур та логів.

7.2. Розмежування доступу до панелі адміністратора системи.

8. Техніко-економічні показники

8.1 Цінність результатів використання даного проекту повинна перевищувати витрати на його реалізацію.

8.2 Система має забезпечувати зниження рівня хибнопозитивних спрацювань порівняно з традиційними системами.

9. Стадії та етапи розробки

№ з/п	Назва етапів магістерської кваліфікаційної роботи	Початок	Закінчення
1	Визначення напрямку магістерської роботи, формулювання теми	24.09.2025	30.09.2025
2	Аналіз предметної області обраної теми	01.10.2025	15.10.2025
3	Розробка гібридного методу та архітектури системи	16.10.2025	31.10.2025
4	Програмна реалізація, навчання моделей та експериментальне дослідження	01.11.2025	20.11.2025
5	Економічне обґрунтування розробки	21.11.2025	30.11.2025
6	Оформлення пояснювальної записки, висновків та презентації	01.12.2025	07.12.2025
8	Захист магістерської кваліфікаційної роботи		

109

10. Порядок контролю та прийому

10.1 До приймання магістерської кваліфікаційної роботи надається:

– ПЗ до магістерської кваліфікаційної роботи;

– програмний додаток;

– презентація;

– відзив керівника роботи;

– відзив опонента

Технічне завдання до виконання прийняв Чуйко Н. В.

Додаток Б. Лістинг програми

```
# -*- coding: utf-8 -*-
"""
PROJECT AEGIS: ENTERPRISE NETWORK SECURITY MONITOR
Architecture: Flask (Backend) + SQLite (DB) + JS/Canvas (Frontend)
"""

import sqlite3
import datetime
import time
import random
import threading
import os
import logging
import sys
from flask import Flask, render_template_string, jsonify, Response, request

# --- CONFIGURATION ---
CONF = {
    'PORT': 5069,
    'DB': 'aegis_core.db',
    'REFRESH_RATE': 1.0,
    'MAX_LOGS': 300
}


# --- LOGGING SETUP ---
logging.basicConfig(level=logging.ERROR)  # Мінімум шуму в консолі
log = logging.getLogger('werkzeug')
log.setLevel(logging.ERROR)

app = Flask(__name__)

#
==========================================================================
====================
# [BACKEND] DATABASE LAYER
#
```

```
================================================================
====================
class Database:
    def __init__(self):
        self.path = CONF['DB']
        self.init_schema()

    def get_conn(self):
        conn = sqlite3.connect(self.path, check_same_thread=False)
        conn.row_factory = sqlite3.Row
        # Налаштування SQLite під одночасний запис/читання
        conn.execute("PRAGMA journal_mode=WAL;")
        conn.execute("PRAGMA synchronous=NORMAL;")
        return conn

    def init_schema(self):
        # НЕ видаляємо базу щоразу щоб не втрачати дані
        with self.get_conn() as conn:
            c = conn.cursor()
            c.execute("""
                CREATE TABLE IF NOT EXISTS traffic (
                    id INTEGER PRIMARY KEY AUTOINCREMENT,
                    timestamp TEXT,
                    src_ip TEXT,
                    dst_port INTEGER,
                    protocol TEXT,
                    flag TEXT,
                    size INTEGER,
                    payload_hex TEXT,
                    alert_level TEXT, -- LOW, MED, HIGH, CRIT
                    attack_type TEXT
                )
            """)
            c.execute("""
                CREATE TABLE IF NOT EXISTS stats (
                    key TEXT PRIMARY KEY,
                    value INTEGER
                )
```

```
        """)
        c.executemany(
            "INSERT OR IGNORE INTO stats (key, value) VALUES (?, 0)",
            [('packets',), ('threats',), ('bandwidth',)]
        )
        conn.commit()

db = Database()

# ================================================================
# [BACKEND] SIMULATION ENGINE (The "Brain")
# ================================================================
class ThreatEngine(threading.Thread):
    def __init__(self):
        super().__init__(daemon=True)
        self.running = False
        self._stop = False

    def stop(self):
        self._stop = True

    def generate_ip(self):
        return f"{random.randint(10, 220)}.{random.randint(0,
255)}.{random.randint(0, 255)}.{random.randint(1, 254)}"

    def generate_hex(self, size=32):
        return ' '.join(f'{random.randint(0, 255):02X}' for _ in range(size))

    def run(self):
        print(f"[*] AEGIS ENGINE THREAD ONLINE")
        while not self._stop:
            if not self.running:
                time.sleep(0.3)
                continue
```

```
        try:
            is_attack = random.random() > 0.85
            proto = random.choice(['TCP', 'UDP', 'ICMP', 'HTTP/1.1'])
            port = random.choice([80, 443, 22, 21, 3389, 53, 8080])
            size = random.randint(64, 1500)

            attack_type = "Normal Traffic"
            level = "LOW"
            flag = "ACK"

            if is_attack:
                atk_rnd = random.choice(['SYN Flood', 'SQL Injection', 'XSS', 'Brute
Force', 'Malware C&C'])
                attack_type = atk_rnd

                if atk_rnd == 'SYN Flood':
                    level = "CRIT"
                    flag = "SYN"
                    size = 64
                elif atk_rnd == 'SQL Injection':
                    level = "HIGH"
                    proto = "HTTP/1.1"
                elif atk_rnd == 'Brute Force':
                    level = "MED"
                    port = 22
                else:
                    level = "HIGH"

            conn = db.get_conn()
            cur = conn.cursor()

            cur.execute("""
                INSERT INTO traffic (
                    timestamp, src_ip, dst_port, protocol, flag,
                    size, payload_hex, alert_level, attack_type
                )
                VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?)
```

```
            """, (
                datetime.datetime.now().strftime("%H:%M:%S.%f")[:-3],
                self.generate_ip(),
                port,
                proto,
                flag,
                size,
                self.generate_hex(),
                level,
                attack_type
            ))

            # Оновлення статистики
            cur.execute("UPDATE stats SET value = value + 1 WHERE
key='packets'")
            cur.execute("UPDATE stats SET value = value + ? WHERE
key='bandwidth'", (size,))
            if is_attack:
                cur.execute("UPDATE stats SET value = value + 1 WHERE
key='threats'")

            # Rolling buffer щоб база не розросталася
            cur.execute("""
                DELETE FROM traffic
                WHERE id NOT IN (
                    SELECT id FROM traffic ORDER BY id DESC LIMIT ?
                )
            """, (CONF['MAX_LOGS'],))

            conn.commit()
            conn.close()

            # Інтервали між «пакетами» імітують реальні бурсти
            time.sleep(random.uniform(0.05, 0.5))

        except Exception as e:
            print(f"[ERR] Engine Error: {e}")
            time.sleep(0.5)
```

```
engine = ThreatEngine()
engine.start()

#
============================================================
====================
# [FRONTEND] INTERFACE TEMPLATE
#
============================================================
====================

UI_TEMPLATE = """
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>AEGIS IDS | Enterprise Security</title>
    <link href="https://cdnjs.cloudflare.com/ajax/libs/font-
awesome/6.4.0/css/all.min.css" rel="stylesheet">
    <script src="https://cdn.jsdelivr.net/npm/chart.js"></script>
    <style>
        :root {
            --bg-dark: #050505;
            --panel: #0a0f14;
            --border: #1a2634;
            --text-main: #a0aab5;
            --accent: #00f3ff;
            --danger: #ff0055;
            --warn: #ffcc00;
            --safe: #00ff99;
            --font-mono: 'Courier New', monospace;
        }
        * { box-sizing: border-box; }
        body {
            background: var(--bg-dark);
            color: var(--text-main);
            font-family: 'Segoe UI', sans-serif;
```

```
    margin: 0;
    padding: 0;
    overflow: hidden;
    height: 100vh;
    display: flex;
    flex-direction: column;
}
header {
    height: 60px;
    background: var(--panel);
    border-bottom: 1px solid var(--accent);
    display: flex;
    align-items: center;
    padding: 0 20px;
    justify-content: space-between;
    box-shadow: 0 0 15px rgba(0, 243, 255, 0.1);
}
.brand {
    font-family: var(--font-mono);
    font-size: 24px;
    font-weight: bold;
    color: var(--accent);
    letter-spacing: 3px;
    text-shadow: 0 0 10px var(--accent);
}
.status-bar {
    display: flex;
    gap: 20px;
    font-family: var(--font-mono);
    font-size: 12px;
}
.stat-item i { margin-right: 5px; }
.live-ind {
    width: 10px;
    height: 10px;
    background: var(--danger);
    border-radius: 50%;
    display: inline-block;
```

```
    animation: blink 1s infinite;
}
.grid-container {
    display: grid;
    grid-template-columns: 280px 1fr 350px;
    grid-template-rows: 240px 1fr;
    gap: 10px;
    padding: 10px;
    height: calc(100vh - 60px);
}
.panel {
    background: var(--panel);
    border: 1px solid var(--border);
    border-radius: 4px;
    padding: 15px;
    position: relative;
    overflow: hidden;
    display: flex;
    flex-direction: column;
}
.panel-title {
    font-family: var(--font-mono);
    color: var(--accent);
    font-size: 12px;
    text-transform: uppercase;
    margin-bottom: 10px;
    border-bottom: 1px solid var(--border);
    padding-bottom: 5px;
    display: flex;
    justify-content: space-between;
    align-items: center;
}
.p-controls { grid-column: 1 / 2; grid-row: 1 / 2; }
.btn {
    width: 100%;
    padding: 10px;
    margin-bottom: 8px;
    border: 1px solid var(--accent);
```

```
            background: transparent;
            color: var(--accent);
            font-family: var(--font-mono);
            cursor: pointer;
            font-weight: bold;
            transition: 0.2s;
            text-transform: uppercase;
        }
        .btn:hover {
            background: var(--accent);
            color: #000;
            box-shadow: 0 0 15px var(--accent);
        }
        .btn-stop { border-color: var(--danger); color: var(--danger); }
        .btn-stop:hover { background: var(--danger); color: #fff; box-shadow: 0 0
15px var(--danger); }
        .btn-secondary {
            border-color: #555;
            color: #888;
            font-size: 10px;
        }

        .p-metrics { grid-column: 2 / 3; grid-row: 1 / 2; display: flex; justify-content:
space-around; align-items: center; }
        .metric-box { text-align: center; }
        .metric-val { font-size: 32px; color: #fff; font-weight: bold; font-family: var(--
font-mono); }
        .metric-lbl { font-size: 10px; text-transform: uppercase; color: #666; }
        .text-red { color: var(--danger); text-shadow: 0 0 10px var(--danger); }

        .p-chart { grid-column: 3 / 4; grid-row: 1 / 2; }

        .p-logs { grid-column: 1 / 3; grid-row: 2 / 3; }
        .log-table { width: 100%; border-collapse: collapse; font-family: var(--font-
mono); font-size: 11px; }
        .log-table th {
            text-align: left;
            color: #666;
```

```
            padding: 5px;
            border-bottom: 1px solid var(--border);
            position: sticky;
            top: 0;
            background: var(--panel);
        }
        .log-table td {
            padding: 4px 5px;
            border-bottom: 1px solid #151b22;
            color: #888;
            white-space: nowrap;
        }
        .log-row:hover {
            background: #151b22;
            color: #fff;
            cursor: pointer;
        }
        .log-row.selected {
            background: #1e2835;
            color: #fff;
        }

        .alert-CRIT { color: var(--danger); font-weight: bold; }
        .alert-HIGH { color: #ff9900; }
        .alert-MED  { color: #ffff00; }
        .alert-LOW  { color: var(--safe); }

        .p-inspect { grid-column: 3 / 4; grid-row: 2 / 3; font-family: var(--font-mono);
font-size: 11px; }
        .hex-view { margin-top: 10px; color: #ccc; line-height: 1.4; word-break:
break-all; max-height: 70vh; overflow-y: auto; }
        .hex-highlight { color: var(--accent); }

        .scan-line {
            height: 2px;
            width: 100%;
            background: var(--accent);
            position: absolute;
```

```
            top: 0;
            left: 0;
            opacity: 0.3;
            animation: scan 3s linear infinite;
            pointer-events: none;
        }
        @keyframes blink { 0% { opacity: 1; } 50% { opacity: 0.3; } 100% { opacity:
1; } }
        @keyframes scan { 0% { top: 0%; } 100% { top: 100%; } }

        ::-webkit-scrollbar { width: 6px; }
        ::-webkit-scrollbar-track { background: #000; }
        ::-webkit-scrollbar-thumb { background: #333; }
        ::-webkit-scrollbar-thumb:hover { background: var(--accent); }

        .filter-bar {
            display: flex;
            gap: 8px;
            align-items: center;
            font-size: 10px;
        }
        .filter-bar select,
        .filter-bar input {
            background: #05080c;
            border: 1px solid #333;
            color: var(--text-main);
            font-size: 10px;
            padding: 3px 5px;
            border-radius: 3px;
            font-family: var(--font-mono);
        }
        .filter-bar label { color: #666; }
    </style>
</head>
<body>

<header>
    <div class="brand"><i class="fas fa-shield-virus"></i> AEGIS IDS</div>
```

```
    <div class="status-bar">
        <span class="stat-item"><i class="fas fa-server"></i> SERVER: <span
id="status-text" style="color:var(--danger)">OFFLINE</span></span>
        <span class="stat-item"><i class="fas fa-microchip"></i> CPU: <span
id="cpu-val">0%</span></span>
        <span class="stat-item"><i class="fas fa-network-wired"></i> <span id="bw-
val">0.0</span> KB/s</span>
    </div>
</header>

<div class="grid-container">
    <!-- CONTROLS -->
    <div class="panel p-controls">
        <div class="panel-title">SYSTEM CONTROL</div>
        <button class="btn" onclick="sendCommand('start')">INITIALIZE</button>
        <button class="btn btn-stop"
onclick="sendCommand('stop')">SHUTDOWN</button>
        <button class="btn btn-secondary" onclick="clearLogs()"><i class="fas fa-
trash"></i> CLEAR BUFFER</button>
        <button class="btn btn-secondary" onclick="window.open('/export')"><i
class="fas fa-download"></i> DUMP CSV</button>
    </div>

    <!-- METRICS -->
    <div class="panel p-metrics">
        <div class="metric-box">
            <div class="metric-val" id="m-packets">0</div>
            <div class="metric-lbl">TOTAL PACKETS</div>
        </div>
        <div class="metric-box">
            <div class="metric-val text-red" id="m-threats">0</div>
            <div class="metric-lbl">THREATS DETECTED</div>
        </div>
        <div class="metric-box">
            <div class="metric-val" id="m-level" style="color:var(--
safe)">NORMAL</div>
            <div class="metric-lbl">DEFCON LEVEL</div>
        </div>
```

```
    </div>

    <!-- THREAT INTENSITY CHART -->
    <div class="panel p-chart">
      <div class="panel-title">THREAT INTENSITY (PER SECOND)</div>
      <canvas id="miniChart"></canvas>
    </div>

    <!-- LOGS -->
    <div class="panel p-logs">
      <div class="scan-line"></div>
      <div class="panel-title">
        <span>TRAFFIC INTERCEPTOR</span>
        <span class="filter-bar">
          <label>LEVEL</label>
          <select id="f-level" onchange="updateData()">
            <option value="ALL">ALL</option>
            <option value="CRIT">CRIT</option>
            <option value="HIGH">HIGH</option>
            <option value="MED">MED</option>
            <option value="LOW">LOW</option>
          </select>
          <label>PROTO</label>
          <select id="f-proto" onchange="updateData()">
            <option value="ALL">ALL</option>
            <option value="TCP">TCP</option>
            <option value="UDP">UDP</option>
            <option value="ICMP">ICMP</option>
            <option value="HTTP/1.1">HTTP</option>
          </select>
          <input id="f-search" type="text" placeholder="IP / signature"
onkeyup="debouncedUpdate()" />
          <span><i class="fas fa-circle live-ind"></i> LIVE</span>
        </span>
      </div>
      <div style="overflow-y: auto; height: 100%;">
        <table class="log-table">
          <thead>
```

```
                <tr>
                    <th width="80">TIME</th>
                    <th width="120">SOURCE IP</th>
                    <th width="60">PORT</th>
                    <th width="60">PROTO</th>
                    <th>SIGNATURE</th>
                    <th width="60">LEVEL</th>
                </tr>
            </thead>
            <tbody id="log-body"></tbody>
        </table>
    </div>
</div>

<!-- INSPECTOR -->
<div class="panel p-inspect">
    <div class="panel-title">DEEP PACKET INSPECTION</div>
    <div id="inspector-content">
        <div style="color:#666; padding: 20px; text-align:center;">
            Select a packet from the log<br>to analyze payload.
        </div>
    </div>
</div>
</div>

<script>
    let refreshInterval = null;
    let chart = null;
    let lastThreatCount = 0;
    let selectedRowId = null;
    let searchTimeout = null;

    window.onload = function() {
        initChart();
    };

    function initChart() {
        const ctx = document.getElementById('miniChart').getContext('2d');
```

```
    chart = new Chart(ctx, {
        type: 'bar',
        data: {
            labels: Array(20).fill(''),
            datasets: [{
                data: Array(20).fill(0),
                backgroundColor: '#ff0055',
                barThickness: 5
            }]
        },
        options: {
            responsive: true,
            maintainAspectRatio: false,
            plugins: { legend: { display: false } },
            scales: {
                x: { display: false },
                y: { display: false, beginAtZero: true }
            },
            animation: false
        }
    });
}

function sendCommand(cmd) {
    fetch(`/api/cmd/${cmd}`)
        .then(r => r.json())
        .then(data => {
            const st = document.getElementById('status-text');
            if (data.status) {
                st.innerText = "ONLINE";
                st.style.color = "var(--safe)";
                if (!refreshInterval) {
                    updateData();
                    refreshInterval = setInterval(updateData, 1000);
                }
            } else {
                st.innerText = "OFFLINE";
                st.style.color = "var(--danger)";
```

```
                if (refreshInterval) {
                    clearInterval(refreshInterval);
                    refreshInterval = null;
                }
            }
        });
    }

    function debouncedUpdate() {
        if (searchTimeout) clearTimeout(searchTimeout);
        searchTimeout = setTimeout(updateData, 300);
    }

    function buildQueryParams() {
        const lvl = document.getElementById('f-level').value;
        const proto = document.getElementById('f-proto').value;
        const search = document.getElementById('f-search').value.trim();
        const params = new URLSearchParams();
        if (lvl && lvl !== 'ALL') params.append('lvl', lvl);
        if (proto && proto !== 'ALL') params.append('proto', proto);
        if (search.length > 0) params.append('search', search);
        return params.toString();
    }

    function updateData() {
        const q = buildQueryParams();
        const url = q ? `/api/data?${q}` : '/api/data';

        fetch(url)
            .then(r => r.json())
            .then(d => {
                document.getElementById('m-packets').innerText = d.stats.packets;
                document.getElementById('m-threats').innerText = d.stats.threats;
                document.getElementById('bw-val').innerText = (d.stats.bandwidth /
1024).toFixed(1);
                document.getElementById('cpu-val').innerText =
Math.floor(Math.random() * 30 + 10) + "%";
```

```
            const elLvl = document.getElementById('m-level');
            if (d.stats.threats > 80) {
               elLvl.innerText = "CRITICAL";
               elLvl.style.color = "var(--danger)";
            } else if (d.stats.threats > 40) {
               elLvl.innerText = "ELEVATED";
               elLvl.style.color = "var(--warn)";
            } else {
               elLvl.innerText = "NORMAL";
               elLvl.style.color = "var(--safe)";
            }

            const deltaThreats = Math.max(0, d.stats.threats - lastThreatCount);
            lastThreatCount = d.stats.threats;
            chart.data.datasets[0].data.shift();
            chart.data.datasets[0].data.push(deltaThreats);
            chart.data.datasets[0].backgroundColor = deltaThreats > 0 ? '#ff0055' :
'#00f3ff';
            chart.update();

            const tbody = document.getElementById('log-body');
            tbody.innerHTML = "";
            d.logs.forEach(l => {
               const tr = document.createElement('tr');
               tr.className = 'log-row';
               tr.dataset.id = l.id;
               if (selectedRowId && selectedRowId === l.id) {
                  tr.classList.add('selected');
               }
               tr.onclick = () => inspectPacket(l.id, tr);

               tr.innerHTML = `
                  <td>${l.time}</td>
                  <td>${l.src}</td>
                  <td>${l.dst}</td>
                  <td>${l.proto}</td>
                  <td>${l.type}</td>
                  <td class="alert-${l.lvl}">${l.lvl}</td>
```

```
            `;
            tbody.appendChild(tr);
        });
    });
}

function inspectPacket(id, rowElement) {
    selectedRowId = id;
    document.querySelectorAll('.log-row').forEach(r =>
r.classList.remove('selected'));
    if (rowElement) rowElement.classList.add('selected');

    fetch(`/api/packet/${id}`)
        .then(r => r.json())
        .then(pkt => {
            const el = document.getElementById('inspector-content');
            el.innerHTML = `
                <div style="margin-bottom:10px; border-bottom:1px solid #333;
padding-bottom:5px;">
                    <strong>HEADER INFO</strong><br>
                    SRC: <span style="color:var(--accent)">${pkt.src}</span> &gt;
DST PORT: ${pkt.dst}<br>
                    PROTO: ${pkt.proto} | LEN: ${pkt.size} | FLAG: ${pkt.flag}<br>
                    LEVEL: <span class="alert-${pkt.lvl}">${pkt.lvl}</span> |
SIGNATURE: ${pkt.type}
                </div>
                <strong>PAYLOAD HEX DUMP:</strong>
                <div class="hex-view">
                    ${formatHex(pkt.hex)}
                </div>
            `;
        });
}

function formatHex(hexStr) {
    const cleaned = hexStr.trim().split(/\\s+/);
    let out = "";
    for (let i = 0; i < cleaned.length; i++) {
```

```
            if (i % 16 === 0) out += "<br>";
            out += cleaned[i] + " ";
        }
        return out;
    }

    function clearLogs() {
        fetch('/api/clear', { method: 'POST' })
            .then(r => r.json())
            .then(() => {
                selectedRowId = null;
                document.getElementById('inspector-content').innerHTML = '<div
style="color:#666; padding: 20px; text-align:center;">Buffer cleared<br>Waiting
for new packets.</div>';
                updateData();
            });
    }
</script>
</body>
</html>
"""

#
============================================================
====================
# [BACKEND] ROUTES
#
============================================================
====================
@app.route('/')
def index():
    return render_template_string(UI_TEMPLATE)

@app.route('/api/cmd/<action>')
def command(action):
    if action == 'start':
        engine.running = True
    elif action == 'stop':
```

```
        engine.running = False
    return jsonify(status=engine.running)

@app.route('/api/data')
def get_data():
    lvl = request.args.get('lvl')
    proto = request.args.get('proto')
    search = request.args.get('search')

    conn = db.get_conn()
    cur = conn.cursor()

    stats = {}
    for row in cur.execute("SELECT key, value FROM stats"):
        stats[row['key']] = row['value']

    query = "SELECT id, timestamp, src_ip, dst_port, protocol, flag, size,
alert_level, attack_type FROM traffic"
    conds = []
    params = []

    if lvl and lvl.upper() != 'ALL':
        conds.append("alert_level = ?")
        params.append(lvl.upper())
    if proto and proto.upper() != 'ALL':
        conds.append("protocol = ?")
        params.append(proto)
    if search:
        conds.append("(src_ip LIKE ? OR attack_type LIKE ?)")
        like = f"%{search}%"
        params.extend([like, like])

    if conds:
        query += " WHERE " + " AND ".join(conds)

    query += " ORDER BY id DESC LIMIT 50"
    cur.execute(query, params)
```

```
    logs = []
    last_alert = False
    for row in cur.fetchall():
        if row['alert_level'] in ('HIGH', 'CRIT'):
            last_alert = True
        logs.append({
            'id': row['id'],
            'time': row['timestamp'],
            'src': row['src_ip'],
            'dst': row['dst_port'],
            'proto': row['protocol'],
            'type': row['attack_type'],
            'lvl': row['alert_level'],
            'flag': row['flag'],
            'size': row['size']
        })

    conn.close()
    return jsonify(stats=stats, logs=logs, last_alert=last_alert)

@app.route('/api/packet/<int:pid>')
def get_packet(pid):
    conn = db.get_conn()
    cur = conn.cursor()
    row = cur.execute("SELECT * FROM traffic WHERE id = ?", (pid,)).fetchone()
    conn.close()
    if not row:
        return jsonify(error="not found"), 404

    pkt = {
        'id': row['id'],
        'time': row['timestamp'],
        'src': row['src_ip'],
        'dst': row['dst_port'],
        'proto': row['protocol'],
        'type': row['attack_type'],
        'lvl': row['alert_level'],
        'hex': row['payload_hex'],
```

```
        'flag': row['flag'],
        'size': row['size']
    }
    return jsonify(pkt)

@app.route('/api/clear', methods=['POST'])
def clear_buffer():
    conn = db.get_conn()
    cur = conn.cursor()
    cur.execute("DELETE FROM traffic")
    cur.execute("UPDATE stats SET value = 0 WHERE key IN
('packets','threats','bandwidth')")
    conn.commit()
    conn.close()
    return jsonify(status="ok")

@app.route('/export')
def export_csv():
    def generate():
        yield "Time,Source,Port,Protocol,Type,Severity,Size,Payload\n"
        conn = db.get_conn()
        cur = conn.execute("SELECT * FROM traffic ORDER BY id DESC")
        for row in cur:
            yield
f"{row['timestamp']},{row['src_ip']},{row['dst_port']},{row['protocol']}," \

f"{row['attack_type']},{row['alert_level']},{row['size']},{row['payload_hex']}\n"
        conn.close()
    return Response(
        generate(),
        mimetype='text/csv',
        headers={"Content-Disposition": "attachment; filename=ids_dump.csv"}
    )

if __name__ == "__main__":
    print(f"\n>>> AEGIS IDS v2.1 STARTED")
    print(f">>> ACCESS: http://127.0.0.1:{CONF['PORT']}")
    print(f">>> Press Ctrl+C to stop\n")
```

```
try:
    app.run(port=CONF['PORT'], debug=False)
finally:
    engine.stop()
```

Додаток В. Ілюстративний матеріал (презентація)

Магістерська робота на тему:
Удосконалення системи виявлення мережевих атак шляхом застосування гібридного підходу машинного навчання з динамічним налаштуванням порогових значень

Виконав: ст. 2-го курсу, групи 1КІТС-24м Чуйко Н. В.
Керівник: к.ф-м.н., доц. каф. МБІС, Шиян А. А.

Мета і завдання дослідження

Метою кваліфікаційної роботи є підвищення ефективності виявлення мережевих атак шляхом розробки гібридної системи на основі ансамблевих методів машинного навчання та глибоких нейронних мереж з механізмом динамічного налаштування порогових значень.
Для досягнення поставленої мети необхідно вирішити такі завдання:

— провести аналіз сучасного стану та тенденцій розвитку систем виявлення мережевих вторгнень
— дослідити методи машинного навчання, що застосовуються для виявлення кібератак, та визначити можливості їх гібридизації;
— розробити удосконалений метод виявлення мережевих атак на основі гібридного підходу, що поєднує ансамблеві класифікатори, автокодувальники та згорткові нейронні мережі;
— розробити алгоритм динамічного налаштування порогових значень класифікації з урахуванням актуального стану мережевого середовища;
— здійснити програмну реалізацію розробленої гібридної системи виявлення мережевих атак;
— провести експериментальне дослідження та порівняльний аналіз ефективності розробленої системи на еталонних наборах даних;
— обґрунтувати економічну доцільність впровадження розробленої системи.

Наукова новизна одержаних результатів

— запропоновано гібридний метод виявлення мережевих атак, що поєднує ансамблевий класифікатор Random Forest, автокодувальник для детектування аномалій та одновимірну згорткову нейронну мережу CNN-1D для аналізу часових патернів трафіку, що на відміну від існуючих підходів забезпечує одночасне виявлення відомих та невідомих атак з точністю понад 98%;
— запропоновано алгоритм динамічного налаштування порогових значень класифікації, який на відміну від статичних порогів враховує ковзну статистику відхилень реконструкції автокодувальника та ймовірнісні оцінки ансамблевого класифікатора, що дозволяє знизити рівень хибнопозитивних спрацювань на 45–60% порівняно з фіксованими порогами;

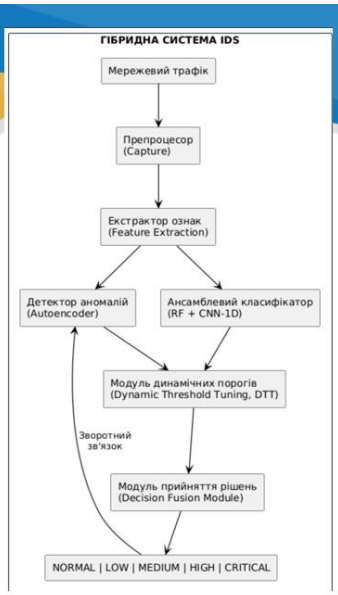

Рисунок — Концептуальна архітектура гібридної системи виявлення мережевих атак

Динамічне налаштування порогових значень

Динамічний поріг оновлюється згідно з правилом:

$$\theta(t) = \alpha \times \theta_{\text{opt}}(t) + (1 - \alpha) \times \theta(t-1)$$

де $\theta_{opt}(t)$ позначає оптимальний поріг на поточному кроці, а α є параметром згладжування зі значеннями від нуля до одиниці.

Програмна реалізація

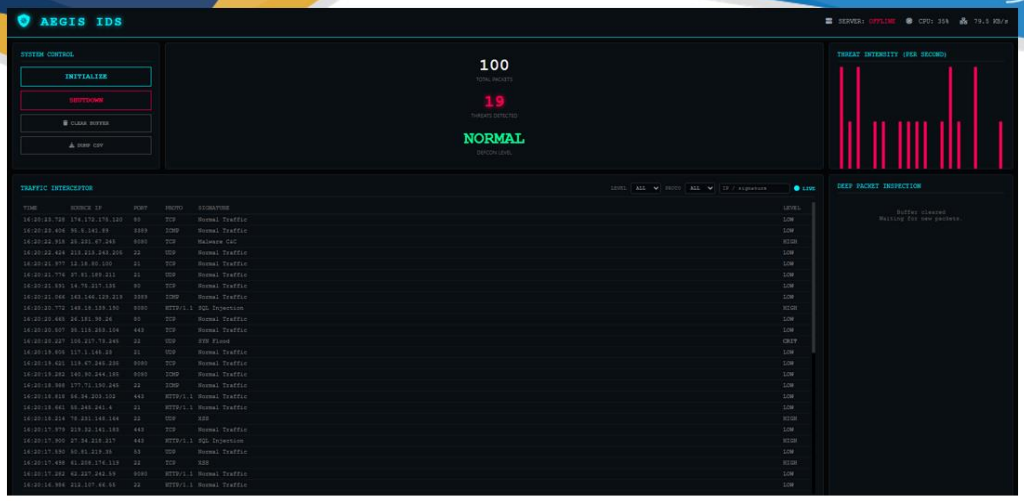

Рисунок — Інтерфейс системи AEGIS IDS

Експериментальне дослідження та порівняльний аналіз ефективності розробленої системи

Метод	Accuracy	Precision	Recall	F1-score	ROC-AUC	Час, мс
Snort IDS (сигнатурний)	0.8923	0.8756	0.8234	0.8487	0.9012	0.3
Random Forest	0.9523	0.9412	0.9378	0.9395	0.9734	1.1
CNN-1D	0.9634	0.9521	0.9489	0.9505	0.9812	1.8
Autoencoder (anomaly)	0.9156	0.8923	0.9567	0.9234	0.9623	1.2
Гібридний (фікс. пороги)	0.9756	0.9678	0.9623	0.9650	0.9889	2.1
AEGIS (динам. пороги)	0.9834	0.9789	0.9756	0.9772	0.9923	2.3

Таблиця — Порівняння ефективності методів виявлення мережевих атак на датасеті CIC-IDS2017

Порівняння фіксованих порогових значень з динамічними

День експлуатації	F1 (фікс. пороги)	F1 (динам. пороги)	Різниця
1	0.9650	0.9772	+1.22%
7	0.9423	0.9745	+3.22%
14	0.9134	0.9712	+5.78%
21	0.8756	0.9689	+9.33%
30	0.8234	0.9656	+14.22%

Таблиця — Порівняння стійкості до концептуального дрейфу

Висновки

У кваліфікаційній роботі вирішено актуальне наукове завдання удосконалення системи виявлення мережевих атак шляхом застосування гібридного підходу машинного навчання з динамічним налаштуванням порогових значень.

Детально описано реалізацію всіх модулів системи: препроцесора, екстрактора ознак, автоенкодера, ансамблевого класифікатора та модуля динамічних порогів. Представлено інтерфейси системи.

Експериментальне дослідження підтвердило ефективність розробленої системи: Accuracy=0.9834, F1-score=0.9772, ROC-AUC=0.9923. Система демонструє суттєву перевагу над базовими методами та результатами літературних джерел. Модуль динамічних порогів забезпечує стійкість до концептуального дрейфу: деградація F1-score після 30 днів становить лише 1.16% порівняно з 14.16% для фіксованих порогів.

Дякую за увагу!

138

Додаток Г. Протокол перевірки на плагіат

ПРОТОКОЛ ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ

Назва роботи: Удосконалення системи виявлення мережевих атак шляхом застосування гібридного підходу машинного навчання з динамічним налаштуванням порогових значень

Тип роботи: магістерська кваліфікаційна робота

Підрозділ: кафедра менеджменту та безпеки інформаційних систем факультет менеджменту та інформаційної безпеки гр.1КІТС-24м

Коефіцієнт подібності текстових запозичень, виявлених у роботі системою StrikePlagiarism (КП1) **1,57 %**

Висновок щодо перевірки кваліфікаційної роботи (відмітити потрібне)

- **Запозичення, виявлені у роботі, оформлені коректно і не містять ознак академічного плагіату, фабрикації, фальсифікації. Роботу прийняти до захисту**
- У роботі не виявлено ознак плагіату, фабрикації, фальсифікації, але надмірна кількість текстових запозичень та/або наявність типових розрахунків не дозволяють прийняти рішення про оригінальність та самостійність її виконання. Роботу направити на доопрацювання.
- У роботі виявлено ознаки академічного плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень. Робота до захисту не приймається.

Експертна комісія:

к.т.н., доцент, зав. каф. МБІС Карпінець В.В. _____

к.ф.-м.н., доцент каф. МБІС Шиян А.А. _____

Особа, відповідальна за перевірку Коваль Н.П. _____

З висновком експертної комісії ознайомлений(-на)

Керівник _____ доц. Шиян А.А.

Здобувач _____ Чуйко Н.В.