НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

ДОПУСКАЄТЬСЯ ДО ЗАХИСТУ

Завідувач кафедри

Комп'ютерних систем, мереж та кібербезпеки

_____ Касаткін Д.Ю., к. пед.н., доц.

підпис ПІБ, вчене звання і ступінь

«___»_____ 2025 р.

БАКАЛАВРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

На тему: «Розробка системи моніторингу для виявлення аномалій у мережевому трафіку»

Спеціальність – 123 «Комп’ютерна інженерія»

Гарант освітньої програми

к. ф.-м. н., доц. Нікітенко Є.В.

(науковий ступінь та вчене звання) (підпис) (ПІБ)

Керівник бакалаврської кваліфікаційної роботи

Д. тех. н., доц. Лахно В.А.

(науковий ступінь та вчене звання) (підпис) (ПІБ)

Виконав _____ Діхтяренко В.П.

(підпис) (ПІБ студента)

КИЇВ-2025
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ
УКРАЇНИ
ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ
ТЕХНОЛОГІЙ

ЗАТВЕРДЖУЮ
Завідувач кафедри

_____

_____ _____ _____
(науковий ступінь, вчене звання) (підпис) (ПІБ)

“____” _____20_____р.

З А В Д А Н Н Я
на виконання бакалаврської кваліфікаційної роботи студенту

Діхтяренко Владислав Петрович
(прізвище, ім’я, по батькові)

Спеціальність (напрям підготовки) _____Комп’ютерна інженерія_____

Тема випускної бакалаврської роботи **Розробка системи моніторингу для виявлення аномалій у мережевому трафіку**

_____

керівник проекту (роботи)_____Лахно В.А.., д.т.н.,професор_____
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)

затверджена наказом ректора НУБіП України від “____”_____20____р. №__

Термін подання завершеної роботи на кафедру _____
(рік, місяць, число)

Вихідні дані до випускної бакалаврської роботи _____
_____
_____

Перелік питань, які потрібно розробити: Аналіз вимог до системи, аналіз предметної області, проєктування, реалізація, тестування системи
_____

Перелік графічних документів (за потреби)_____
_____
_____
_____

Дата видачі завдання “_____”_____20___р.
Керівник випускної бакалаврської роботи _____/ Лахно.В.А
(підпис) (прізвище та ініціали)
Завдання прийняв до виконання _____/ Діхтяренко В.П.
(підпис) (прізвище та ініціали студента)

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів бакалаврської роботи	Строк виконання етапів роботи	Примітка
1	Аналіз вимог до системи	17.03.2025 р.	Виконано
2	Проєктування системи	22.04.2025 р.	Виконано
3	Реалізація системи	03.05.2025 р.	Виконано
4	Тестування розробленої системи	12.05.2025 р.	Виконано
5	Оформлення пояснювальної записки	18.05.2025 р.	Виконано
6	Оформлення графічного матеріалу	20.05.2025 р.	Виконано

Студент _____ / **Діххтяренко В.П.**
(підпис) (ініціали та прізвище)

Керівник проекту (роботи) _____ / **Лахно В.А.**
(підпис) (ініціали та прізвище)

ЗМІСТ

РЕФЕРАТ

Кваліфікаційна робота присвячена розробці програмної системи для виявлення аномалій у мережевому трафіку на основі статистичних ознак потоків і алгоритмів машинного навчання без учителя. У роботі проаналізовано сучасні підходи до виявлення аномальної активності, обґрунтовано вибір моделей Isolation Forest та HBOS, які забезпечують ефективне виявлення відхилень без потреби у попередній розмітці даних.

Сформовано архітектуру системи, реалізовано її основні модулі: обробку вхідних .pcap-файлів, обчислення ознак, виявлення аномалій та вивід результатів через графічний інтерфейс. Система протестована на реальних даних, продемонстровано стабільну роботу та інтерпретовані результати. Показано, що запропоноване рішення може бути використане в задачах моніторингу трафіку в освітньому та дослідницькому середовищі.

Робота містить 67 сторінок основного тексту, 4 таблиці, 12 рисунків, 43 найменувань використаних джерел і 3 додатки.

Ключові слова: мережевий трафік, аномалія, машинне навчання, навчання без учителя, Isolation Forest, HBOS, потокові ознаки, мережевий моніторинг, Python.

ABSTRACT

The qualification work is devoted to the development of a software system for detecting anomalies in network traffic based on statistical features of streams and machine learning algorithms without a teacher. The work analyzes modern approaches to the detection of anomalous activity, substantiates the choice of Isolation Forest and HBOS models, which provide effective detection of deviations without the need for preliminary data marking.

The architecture of the system is formed, its main modules are implemented: processing of input .pcap files, calculation of features, detection of anomalies and output of results through the graphical interface. The system is tested on real data, stable operation and interpreted results are demonstrated. It is shown that the proposed solution can be used in traffic monitoring tasks in the educational and research environment.

The work contains 67 pages of the main text, 4 tables, 12 figures, 43 names of used sources and 3 applications..

Keywords: network traffic, anomaly, machine learning, unsupervised learning, Isolation Forest, HBOS, streaming features, network monitoring, Python.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

AI – Artificial Intelligence (штучний інтелект)

CSV – Comma-Separated Values (формат табличних файлів, розділених комами)

GUI – Graphical User Interface (графічний інтерфейс користувача)

HBOS – Histogram-Based Outlier Score (метод виявлення аномалій на основі гістограм)

IP – Internet Protocol (протокол міжмережевої взаємодії)

ML – Machine Learning (машинне навчання)

MVP – Minimum Viable Product (мінімально життєздатна версія програмного рішення)

PCAP – Packet Capture (формат збереження мережевого трафіку)

TCP – Transmission Control Protocol (протокол керування передачею)

UDP – User Datagram Protocol (протокол користувацьких дейтаграм)

DNS – Domain Name System (система доменних імен)

TTL – Time to Live (час життя пакета)

Flow – послідовність мережевих пакетів, що мають спільний набір ключів (IP, порт, протокол)

ВСТУП

У сучасних умовах цифрової трансформації всі сфери суспільного життя та економіки дедалі більше залежать від стабільності та безпеки інформаційних мереж. Збільшення обсягів переданого даних, інтенсифікація мережевої взаємодії, а також активне використання зашифрованих з'єднань і динамічних протоколів створюють складнощі у виявленні загроз, які не мають чітких ознак або сигнатур. У зв'язку з цим особливої актуальності набувають методи виявлення аномалій у мережевому трафіку, які дозволяють ідентифікувати потенційно небезпечну або нетипову активність, навіть якщо вона раніше не спостерігалася. Саме ця проблема й стала предметом дослідження в межах даної кваліфікаційної роботи.

Метою роботи є розробка та реалізація працюючої комп'ютерної системи, здатної автоматично виявляти аномалії у мережевому трафіку на основі статистичних ознак, без попередньої розмітки даних, із використанням методів машинного навчання.

Для досягнення поставленої мети у процесі дослідження необхідно вирішити наступні завдання:

- вивчити класифікацію мережевих аномалій та принципи їх виявлення;
- описати існуючі підходи до аналізу трафіку: сигнатурні, статистичні, поведінкові;
- виявити обмеження традиційних методів та обґрунтувати вибір алгоритмів навчання без учителя;
- розробити архітектуру системи виявлення аномалій на основі потокових ознак;
- реалізувати програмну систему з інтерактивним інтерфейсом для офлайн-аналізу трафіку;
- провести тестування системи на прикладових даних та оцінити її ефективність.

Об’єктом дослідження у даній роботі виступає процес моніторингу та класифікації мережевого трафіку в інформаційних системах із метою виявлення нетипової активності.

Методами дослідження, що застосовувалися в ході виконання роботи, є аналіз відкритих даних мережевого трафіку, побудова моделей навчання без учителя, зокрема алгоритмів Isolation Forest та HBOS, обробка даних за допомогою Pandas, а також моделювання системної архітектури та створення інтерфейсу на базі Streamlit. Усі моделі оцінювалися за якісними критеріями, притаманними задачам без учителя, із застосуванням інтерпретаційних і статистичних підходів.

У рамках роботи студентом було самостійно виконано повний цикл: від постановки задачі до реалізації програмного середовища, тестування і написання пояснювальної записки.

Практична значущість роботи полягає в тому, що створена система може бути використана в освітньому процесі, для тренування моделей в умовах обмежених даних, а також як основа для створення внутрішніх інструментів мережевого моніторингу у малих організаціях чи навчальних закладах. Завдяки простоті використання та відсутності залежності від зовнішніх API або БД, систему можна швидко адаптувати під різні середовища.

Структура кваліфікаційної роботи побудована логічно та складається зі вступу, трьох основних розділів, висновків, списку використаних джерел та додатків. У першому розділі проведено теоретичний аналіз предметної області та обґрунтовано вибір технологій. У другому — сформульовано задачу, спроєктовано архітектуру системи та описано її функціональні можливості. У третьому — реалізовано систему, наведено приклади запуску, результати тестування та оцінку ефективності.

РОЗДІЛ 1. АНАЛІЗ ПРОБЛЕМИ ТА ІСНУЮЧИХ РІШЕНЬ

1.1. Поняття мережевої аномалії та завдання моніторингу

У сфері інформаційної безпеки поняття аномалії в мережевому трафіку трактується як відхилення від характерних (очікуваних або "нормальних") шаблонів взаємодії в комп’ютерній мережі. Такі відхилення можуть бути зумовлені як збоєм у роботі обладнання або некоректною конфігурацією, так і цілеспрямованими діями зловмисника, зокрема підготовкою чи реалізацією кібератаки. Виявлення таких відхилень є основою для побудови систем виявлення вторгнень та моніторингу мережі загалом [1].

Аномалії умовно поділяють на декілька категорій залежно від їх характеру, контексту виникнення, інтенсивності та часової тривалості:

- Об’ємні аномалії. До цієї групи відносять нетипові збільшення обсягу трафіку за короткий проміжок часу, наприклад, розподілені атаки на відмову в обслуговуванні (DDoS), які призводять до перевантаження вузлів або каналів зв’язку. Такі аномалії добре виявляються статистичними методами через явну зміну середніх і пікових значень показників.
- Сканування та зондування. Це спроби виявити відкриті порти, активні IP-адреси, слабкі конфігурації протоколів або вразливості. Типовим прикладом є port scan або sweep, коли вузол послідовно надсилає невеликий обсяг пакетів на велику кількість хостів або портів. Аномалія фіксується не за обсягом, а за поведінкою: множинність коротких сесій, повторюваність запитів.
- Аномалії рівня протоколу. Вони проявляються у вигляді нестандартного використання протоколів, порушення їх специфікацій, або невластивої поведінки для заданої мережі (наприклад, ICMP-резонанс або HTTP-запити на невеб-сервери). До цієї ж категорії можна віднести тунелювання через DNS, що також виявляється як поведінкова аномалія.

- Тривалі та «приховані» аномалії. Цей тип є найскладнішим для детектування. До нього належать сповільнені або масковані атаки типу “low and slow”, бекдори, C&C (command and control) взаємодія через легітимний HTTPS. Зміни у поведінці таких сесій можуть бути настільки мінімальними, що їх складно виявити класичними методами без аналізу послідовностей у часі.

З методологічної точки зору, у літературі також наводиться класифікація за характером аномалії у часовому просторі [2]:

- Точкова аномалія (point anomaly) — одинична подія або об'єкт значно відрізняється від інших спостережень.
- Контекстуальна аномалія (contextual anomaly) — відхилення, яке є аномальним тільки в певному контексті (наприклад, трафік 500 Мб за ніч на офісному ПК).
- Послідовна аномалія (collective anomaly) — група об'єктів, яка загалом утворює відхилення, хоча окремо кожен з них не є підозрілим (наприклад, стеганографічне тунелювання через регулярні DNS-запити).

Мережеві аномалії становлять широкий спектр потенційно небезпечної або нетипової поведінки в інфраструктурі. Їх класифікація має важливе значення для формування адекватних методів моніторингу, а також для побудови моделей виявлення, здатних коректно реагувати як на об’ємні, так і на приховані типи порушень [3].

У сучасних комп’ютерних мережах виявлення аномальної активності стало однією з ключових задач інформаційної безпеки, проте її ефективна реалізація супроводжується рядом суттєвих труднощів. Основна проблема полягає у високій швидкості та обсягах передавання даних, що унеможливлює повноцінний ручний аналіз або застосування обчислювально затратних методів у реальному часі. Зокрема, в корпоративних або провайдерських мережах трафік може досягати десятків гігабіт на секунду, що вимагає високопродуктивного попереднього фільтрування й агрегування.

Ще однією важливою проблемою є відсутність чітких міток у більшості реального мережевого трафіку. Більшість наявних даних є немаркованими, тобто не містять інформації про те, які потоки є нормальними, а які – зловмисними [4]. У результаті методи машинного навчання, що базуються на навчанні з учителем, часто виявляються непридатними для практичного застосування, оскільки потребують попередньої ручної розмітки великих обсягів даних. Це ускладнює адаптацію моделей до нових середовищ і атак, які раніше не зустрічались у навчальній вибірці.

Крім того, варто враховувати таке явище, як зміна профілю нормального трафіку з плином часу (так званий concept drift). Мережеві поведінкові патерни можуть змінюватись під впливом змін у топології, оновлень ПЗ, зміни політик доступу, сезонних коливань активності тощо [5]. Якщо модель виявлення аномалій не здатна адаптуватися до цих змін, вона почне видавати велику кількість хибнопозитивних спрацювань, що в кінцевому підсумку призводить до втрати довіри до результатів.

Окремо слід згадати проблему прихованої або слабо вираженої аномальної активності. Багато сучасних атак маскуються під звичайний трафік або реалізуються дуже повільно, наприклад, через використання "low-and-slow" стратегій. У таких випадках відхилення від норми є мінімальними і не завжди піддаються фіксації класичними пороговими методами. Особливо це актуально для зашифрованого трафіку (HTTPS, SSH, VPN), де зміст пакетів недоступний для аналізу, а отже, виявлення можливо лише за статистичними або поведінковими ознаками [6].

Сучасні умови функціонування мереж зумовлюють потребу в автоматизованих, адаптивних та ефективних інструментах виявлення аномалій. Ці інструменти мають враховувати обмеженість міток, здатність моделі до переобучення на нових даних, підтримку в реальному часі та стійкість до шифрування. Без урахування зазначених викликів побудова надійної системи моніторингу аномалій є практично неможливою.

У контексті забезпечення безпеки інформаційної інфраструктури системи моніторингу мережевого трафіку виконують критично важливу роль. Їх основним завданням є постійне спостереження за потоками даних у межах корпоративної або глобальної мережі з метою оперативного виявлення аномальної поведінки, яка потенційно може свідчити про зловмисну активність, технічні збої або порушення політик доступу [7]. На відміну від класичних міжмережевих екранів або фільтрів, системи моніторингу орієнтовані не лише на перехоплення та блокування, а на глибший аналітичний процес, що дозволяє фіксувати нетипову поведінку навіть за відсутності явних ознак атаки.

Основна функція такої системи полягає в безперервному зборі даних про мережеву активність. Збір може відбуватись як у режимі реального часу (онлайн-моніторинг), так і шляхом обробки попередньо збережених дампів трафіку (офлайн-аналіз). У обох випадках вихідні пакети перетворюються у структури вищого рівня – так звані flows, що характеризуються ключовими параметрами: IP-адресами, портами, протоколами, розміром, тривалістю та іншими статистичними показниками [8]. Далі відбувається формування ознак, що підлягають математичному аналізу за допомогою алгоритмів виявлення аномалій.

У типовому випадку сучасна система моніторингу реалізує декілька базових етапів: захоплення трафіку (наприклад, за допомогою бібліотеки Scapy), агрегацію сесій у потоки, розрахунок статистичних ознак, застосування алгоритму детекції (у нашому випадку – Isolation Forest або HBOS) та формування висновків. При виявленні підозрілої активності система маркує відповідні потоки, зберігає їх у звіті, і, за потреби, сповіщає адміністратора через інтерфейс користувача або сторонні сервіси. Такий підхід дозволяє автоматизувати початковий етап аналізу інцидентів і значно зменшити навантаження на персонал служби безпеки [9].

Не менш важливим завданням системи моніторингу є накопичення та систематизація історичних даних про аномальні події, що дозволяє здійснювати повторний аналіз, тренування моделей на реальних інцидентах і формування

профілю типового трафіку для конкретної організації. В умовах зростаючої складності атак і постійного розширення мережевої інфраструктури подібні системи стають ключовим елементом у структурі Security Operation Center (SOC), забезпечуючи як оперативне реагування, так і довгострокову аналітику ризиків.

Роль системи моніторингу мережевого трафіку не обмежується технічним інструментом збору даних – це повноцінна аналітична платформа, що забезпечує виявлення порушень, адаптацію до нових загроз і підтримку прийняття рішень у сфері інформаційної безпеки.

1.2. Огляд підходів до виявлення аномалій: сигнатурні, статистичні, ML

Сигнатурні методи виявлення аномалій у мережевому трафіку є найдавнішими і водночас найпоширенішими в реальних системах захисту. Їх основний принцип полягає у використанні попередньо визначених правил або шаблонів, які описують характеристики відомих загроз. Такий підхід передбачає наявність бази даних сигнатур, тобто унікальних ознак атак, що зіставляються із вхідним трафіком у режимі реального часу [10].

Фактично, сигнатурна модель функціонує за аналогією з антивірусами, де кожна ознака атаки (наприклад, певна послідовність байтів, заголовок HTTP-запиту або специфічна структура TCP-пакета) має свій фіксований опис. Якщо під час аналізу трафіку система знаходить збіг з однією з таких сигнатур, подія маркується як шкідлива. Одним із найвідоміших прикладів системи такого типу є Snort — відкрите середовище для мережевого моніторингу, що використовує правила формату "header + content" для точного розпізнавання атак. Аналогічно працює Suricata — ще один сигнатурно-орієнтований IDS-движок з підтримкою багатопотокової обробки.

Перевагою сигнатурних методів є їх висока точність для відомих типів загроз, а також пояснюваність результатів: при виявленні інциденту можна точно вказати, яка саме сигнатура спрацювала [10]. Крім того, ці методи добре масштабуються та мають чітку інфраструктуру оновлення — правила можуть завантажуватись централізовано з офіційних репозиторіїв (наприклад, Emerging Threats або Talos).

Однак найбільшим недоліком такого підходу є його принципова обмеженість у виявленні нових або модифікованих атак. Так звані zero-day атаки, які ще не мають відомих сигнатур, залишаються невидимими для сигнатурних IDS/IPS-систем. Крім того, при надмірній деталізації правил можливе виникнення конфліктів або пропуск складніших багатоступеневих сценаріїв. У разі широкого використання шифрування трафіку (TLS, VPN) сигнатурні механізми втрачають ефективність, оскільки не мають доступу до корисного навантаження пакетів.

Отже, хоча сигнатурні методи залишаються важливою складовою систем забезпечення мережевої безпеки, вони не здатні охопити весь спектр загроз, особливо в умовах динамічного розвитку кібератак. Саме тому все більшого значення набувають альтернативні підходи — статистичні та засновані на машинному навчанні, які мають здатність до узагальнення та адаптації до нових умов [11].

На відміну від сигнатурного підходу, статистичні методи виявлення аномалій ґрунтуються не на фіксованих шаблонах атак, а на аналізі відхилень від певного профілю «нормальної» поведінки мережі. Такі методи зазвичай класифікуються як детектори поведінки (behavior-based detection) і широко використовуються в системах, де пріоритетом є виявлення невідомих або нестандартних інцидентів.

У статистичному підході передбачається, що кожен об'єкт спостереження — наприклад, мережевий потік — можна описати набором кількісних характеристик: кількістю переданих байтів, кількістю пакетів, тривалістю сесії, середнім розміром пакета, частотою запитів тощо. Для кожної з цих ознак або

для їх комбінацій будуються порогові правила або моделі розподілу, з яких визначається «норма». У випадках, коли значення параметрів суттєво виходить за межі встановленого інтервалу (наприклад, понад 3 стандартних відхилення від середнього), трафік розглядається як потенційно аномальний.

Найпростіші варіанти статистичних методів включають z-оцінку, міжквартильний розмах (IQR), аналіз ентропії протоколів або портів, а також побудову гістограм розподілу частоти звернень до певних адрес чи сервісів. Більш просунуті підходи передбачають використання адаптивних порогів або ковзаючих вікон, які динамічно коригуються в залежності від поведінки мережі у попередні періоди. Окремим напрямом є застосування статистичних тестів, зокрема критерію Колмогорова–Смирнова для порівняння поточного розподілу з еталонним [12].

Основною перевагою статистичних методів є їх універсальність і відсутність залежності від попередньо заданих сигнатур. Це дозволяє їм фіксувати невідомі загрози або технічні збої, які не підпадають під жоден з відомих шаблонів. Однак така гнучкість має і свої недоліки. По-перше, для кожної конкретної мережі необхідно окремо будувати профіль «нормальної» поведінки, що потребує накопичення значного обсягу даних і ручного налаштування. По-друге, у динамічних середовищах, де норма змінюється, статистичні моделі можуть швидко втратити актуальність і почати видавати хибні спрацювання.

Ще одним викликом є висока чутливість до вибору ознак: не всі параметри мають однакову інформативність у різних контекстах. Також слід враховувати можливість навмисної адаптації зловмисників до відомих статистичних механізмів, зокрема шляхом «розмивання» профілю атаки серед легітимного трафіку.

Статистичні методи займають проміжне положення між класичними сигнатурними системами та більш гнучкими, але складнішими алгоритмами машинного навчання. Вони ефективні на початкових етапах виявлення аномалій,

особливо у поєднанні з іншими механізмами, але потребують ретельної параметризації та періодичного оновлення моделей у процесі експлуатації [13].

Методи машинного навчання (ML) дедалі частіше застосовуються для виявлення аномалій у мережевому трафіку як універсальний інструмент, здатний працювати з великими обсягами складноструктурованих даних та виявляти відхилення, які не піддаються класичній формалізації. Головна перевага цих методів полягає у здатності моделі самостійно «вчитися» на історичних даних і знаходити нетипові патерни без необхідності ручного створення сигнатур або задання порогів.

У контексті виявлення аномалій зазвичай використовують дві основні парадигми машинного навчання: навчання з учителем (supervised learning) та без учителя (unsupervised learning). Перша передбачає наявність мічених даних, тобто таких, де для кожного прикладу (мережевого потоку) вказано, чи є він аномальним. У цьому випадку модель навчається класифікувати нові спостереження на основі заздалегідь визначених класів. До таких алгоритмів належать дерева рішень, випадкові ліси, логістична регресія, нейронні мережі тощо. Проте обмеженням цього підходу є відсутність або брак міток у більшості реального трафіку, що обмежує його застосування в практичних умовах [13].

Тому в задачах виявлення аномалій особливо поширеними стали методи навчання без учителя, де моделі формують уявлення про «норму» і фіксують відхилення від неї. Найвідомішими алгоритмами цієї групи є Isolation Forest, який ґрунтується на ідеї випадкового розбиття простору ознак для ізоляції об’єктів, Local Outlier Factor (LOF), що вимірює локальну щільність розташування, та One-Class SVM — метод опорних векторів для однокласового навчання. Також активно використовуються статистично-евристичні моделі, такі як HBOS (Histogram-Based Outlier Score), який оцінює рідкість кожного спостереження на основі побудованих гістограм.

Окрему нішу займають глибокі моделі — автоенкодери, рекурентні нейронні мережі (LSTM) та їх модифікації. Вони дозволяють враховувати залежності у часових рядах і будувати складні уявлення про структуру трафіку.

Зокрема, автоенкодери навчаються стискати потік даних у компактне представлення і виявляти аномалії на основі помилки реконструкції, яка зростає для нетипових прикладів [14]. Перевага таких моделей — висока чутливість до прихованих закономірностей, проте вони є обчислювально складними й менш інтерпретованими.

Водночас ефективність ML-методів значною мірою залежить від якості попередньої обробки даних. Формування релевантних ознак (feature engineering), нормалізація масштабів, видалення надлишкових змінних або дисбалансу класів — усе це суттєво впливає на кінцевий результат. Крім того, машинне навчання вимагає постійної адаптації моделі до змін у профілі трафіку, оскільки навіть найкраща модель з часом втрачає актуальність через концептуальний дрейф [14].

У підсумку, методи машинного навчання становлять потужний інструментарій для побудови адаптивних систем виявлення аномалій. Їхнє успішне застосування потребує грамотної реалізації повного циклу обробки даних — від збору та попередньої агрегації до автоматизованого навчання, тестування й валідації результатів. З урахуванням стрімкого розвитку інфраструктур моніторингу та доступності бібліотек з відкритим кодом, ML-підхід стає оптимальним вибором у багатьох практичних сценаріях, особливо там, де сигнатурні або статистичні методи не дають бажаного результату.

1.3. Порівняння типових рішень

У процесі дослідження проблеми виявлення аномалій у мережевому трафіку доцільним є порівняння основних підходів – сигнатурного, статистичного та заснованого на машинному навчанні – за ключовими експлуатаційними та методологічними характеристиками. Такий порівняльний аналіз дозволяє оцінити сильні та слабкі сторони кожного підходу й визначити доцільність їх використання у конкретних умовах, зокрема з урахуванням наявних обмежень і цілей розробки [15].

Сигнатурні методи характеризуються високою точністю щодо вже відомих атак і мінімальною кількістю хибнопозитивних спрацювань, однак вони нездатні виявляти нові або модифіковані загрози, які ще не мають відповідної сигнатури. Статистичні методи виявлення базуються на аналізі відхилень від нормативних показників і не залежать від баз знань, проте вимагають побудови «нормального профілю» та є чутливими до зміни поведінки мережі [16]. Методи машинного навчання, у свою чергу, демонструють найвищу адаптивність і здатність виявляти раніше невідомі типи аномалій, проте потребують підготовки якісних даних і значних обчислювальних ресурсів для навчання та обробки. Порівняльна характеристика підходів наведена у таблиці 1.1.

Таблиця 1.1

Порівняння підходів до виявлення аномалій у мережевому трафіку

Критерій	**Сигнатурні методи**	**Статистичні методи**	**Методи машинного навчання**
Виявлення нових атак	Низька	Середня	Висока
Залежність від попередніх знань	Повна (сигнатури)	Часткова (норми)	Мінімальна
Потреба у навчальних даних	Відсутня	Історичні значення	Необхідні для навчання
Адаптивність до нових умов	Обмежена	Часткова	Повна
Продуктивність у реальному часі	Висока	Середня	Залежить від моделі
Інтерпретованість результатів	Висока	Середня	Низька
Швидкість впровадження	Швидка	Помірна	Залежить від налаштувань
Складність підтримки	Низька	Середня	Висока

Жоден з підходів не є універсальним, і вибір конкретного методу залежить від контексту використання. У системах, де критичною є швидкість реакції на

відомі загрози, доцільно використовувати сигнатурні системи. Для гнучкого виявлення аномальної поведінки у великому обсязі неструктурованих даних — перевагу отримують статистичні або ML-моделі. У більшості практичних випадків доцільним є комбінування методів з метою досягнення балансу між точністю, гнучкістю та продуктивністю.

Серед практичних реалізацій систем виявлення аномалій найбільш широкого поширення набули такі інструменти як Snort, Suricata, Zeek (раніше Bro) та PyOD. Кожен із них реалізує свій підхід до аналізу мережевого трафіку, має унікальні можливості і обмеження. У цьому підпункті розглянемо ці рішення як з точки зору функціональних особливостей, так і з позиції придатності до інтеграції у власну систему моніторингу [17].

Snort — один з найстаріших і найпоширеніших сигнатурних IDS-движків з відкритим вихідним кодом. Він орієнтований на виявлення відомих атак за допомогою набору правил, які можна оновлювати централізовано. Його перевагами є висока швидкість обробки і підтримка реального часу, однак він практично непридатний для виявлення нових загроз або аномалій, які не мають сигнатури.

Suricata — сучасна альтернатива Snort, яка окрім сигнатурного аналізу реалізує базовий статистичний аналіз потоків, підтримує багатопотокову обробку, детектує TLS, HTTP, DNS і забезпечує розширену підтримку протоколів. Suricata також може генерувати NetFlow-подібні звіти, що робить її ближчою до систем мережевого моніторингу.

Zeek — фреймворк для поведінкового аналізу мережевого трафіку, який надає великі можливості для створення власних політик обробки, а не обмежується фіксованими правилами. Zeek використовує власну мову скриптів, добре підходить для аналітики, має підтримку потокового аналізу, але вимагає налаштування та не призначений для високошвидкісного реального захисту.

PyOD — Python-бібліотека для виявлення аномалій на основі машинного навчання. Вона реалізує понад 40 алгоритмів, серед яких Isolation Forest, HBOS, AutoEncoder, kNN тощо. PyOD придатна для використання в наукових і

прикладних проєктах, у тому числі у кастомних системах моніторингу. На відміну від попередніх систем, PyOD не є NIDS-системою, але служить ядром для аналітичних обчислень на рівні ознак потоків. Зведене порівняння інструментів наведено у таблиці 1.2.

Таблиця 1.2

Порівняння популярних інструментів для аналізу мережевого трафіку

Інструмент	**Підхід**	**Мова реалізації**	**Ліцензія**	**Алгоритми**	**Обробка**	**Продуктивність**	**Інтерфейс**
Snort	Сигнатурний	C	GPL	Фіксовані правила	Реальний час	Висока	Консоль
Suricata	Сигнатурний + статистичний	C	GPL	Правила + аналіз flow	Реальний час	Дуже висока	JSON-логи
Zeek (Bro)	Поведінковий (policy-based)	C++ / Zeek-script	BSD	Скриптові механізми	Псевдореальний час	Висока	Текстові журнали
PyOD	Машинне навчання	Python	MIT	>40 алгоритмів	Офлайн / потоково	Середня	Інтегрується (GUI можливий)

Як видно з таблиці 1.2, класичні IDS-системи орієнтовані на обробку трафіку в реальному часі з акцентом на вже відомі загрози, у той час як PyOD забезпечує потужну ML-аналітику для нестандартних сценаріїв і нових типів атак. У рамках побудови сучасних гібридних систем доцільним є використання комбінації вищезазначених рішень або їх окремих компонентів відповідно до поставлених завдань [18]. У даному проєкті акцент зроблено на використанні PyOD як ядра для моделювання та виявлення аномалій у потоках з

використанням Isolation Forest і HBOS, що дає змогу об'єднати переваги поведінкового аналізу та статистичної стабільності.

1.4. Обґрунтування вибору методу та технологій

На етапі визначення алгоритмічної основи системи виявлення аномалій у мережевому трафіку особливу увагу слід приділити формалізації критеріїв, за якими має оцінюватись придатність того чи іншого методу. У контексті аналізу потоків мережевого трафіку, де дані є немаркованими, об’ємними та потенційно зашумленими, найбільш доцільним є використання методів навчання без учителя (unsupervised learning).

Першим ключовим критерієм є незалежність від попередньої розмітки даних. У більшості реальних мережевих середовищ не існує наперед визначених міток, що вказують на нормальну або аномальну поведінку. Ручна розмітка трафіку є ресурсомісткою та суб’єктивною, а отже, використання алгоритмів, які не потребують вчителя, значно підвищує адаптивність системи до нових середовищ [18].

Другим критерієм виступає стійкість до шуму та здатність фіксувати як точкові, так і структурні відхилення. Мережевий трафік є динамічним, а тому система повинна реагувати на широке коло порушень: від короткочасних сплесків обсягу до прихованої регулярної активності. Методи мають бути здатними обробляти великі вибірки, виділяючи не лише одиничні викиди, а й нетипові патерни.

Третім важливим фактором є масштабованість обраного алгоритму. Зростання обсягу трафіку вимагає ефективної реалізації, здатної обробляти десятки тисяч потоків на рівні ознак. Обрана модель має працювати з помірними обчислювальними ресурсами та бути стійкою до високої розмірності ознакового простору.

З огляду на зазначені вимоги, серед поширених підходів були розглянуті наступні кандидати:

1. Local Outlier Factor (LOF) — добре виявляє локальні відхилення, але має високу обчислювальну складність та погано масштабується.
2. One-Class SVM — ефективний для простих розподілів, але чутливий до параметрів ядра та вимагає ретельного налаштування.
3. Autoencoder — дозволяє моделювати складну структуру даних, однак потребує великої кількості навчальних прикладів і тривалого часу навчання.
4. Histogram-Based Outlier Score (HBOS) — простий та швидкий метод для незалежних ознак, але менш точний за складних взаємозв'язків.
5. Isolation Forest — модель, що не потребує гіперпараметрів, ефективна для високовимірних даних, стійка до шуму та легко інтерпретується.

З-поміж розглянутих варіантів найбільш збалансованими за критеріями точності, швидкодії та зручності реалізації виявилися Isolation Forest як основний метод, і HBOS як його обчислювально дешевий альтернативний варіант. Обидва алгоритми є придатними до реалізації у форматі автономної моделі для виявлення аномальних потоків без потреби у попередньому навчанні на мічених даних.

Вибір програмного середовища є критичним аспектом під час розробки систем, що працюють з мережевими даними та алгоритмами машинного навчання. Для побудови системи моніторингу, орієнтованої на виявлення аномалій у трафіку, доцільним є використання мови Python як основної платформи розробки. Такий вибір обґрунтовується як технічними, так і організаційними перевагами, зокрема — відкритістю, доступністю, широким набором готових бібліотек та активною спільнотою [19].

По-перше, Python є загальновизнаним стандартом у сфері обробки даних і прототипування алгоритмів машинного навчання. У порівнянні з іншими мовами (наприклад, C++ або Java), Python забезпечує швидкий цикл розробки, що особливо важливо на стадії створення функціональної системи. Крім того, його

синтаксис є зрозумілим і компактним, що спрощує підтримку та повторне використання коду, а також підвищує доступність системи для фахівців суміжних напрямів.

По-друге, Python має вбудовану підтримку великої кількості галузевих бібліотек, які є критично важливими для реалізації різних компонентів системи моніторингу:

- для обробки та фільтрації мережевого трафіку застосовуються бібліотеки низького рівня (наприклад, Scapy), що дозволяють інтерпретувати сирі пакети;
- для представлення агрегованих потоків — pandas та NumPy, які є стандартом у роботі з табличними даними;
- для побудови моделей виявлення аномалій — scikit-learn, PyOD, які включають широкий спектр алгоритмів, адаптованих для задач класифікації, кластеризації та outlier detection.

По-третє, використання Python забезпечує легку інтеграцію між модулями системи — наприклад, між обробкою даних, навчанням моделей і виведенням результатів через інтерфейс. Завдяки кросплатформеності Python-код може бути розгорнутий як у локальному середовищі розробника, так і на сервері або в контейнеризованому середовищі без істотних змін.

У сукупності ці переваги роблять Python оптимальним вибором для побудови системи виявлення аномалій на рівні мережевого трафіку, особливо на початкових етапах розробки, де важливо швидко досягти працездатного результату при мінімальних витратах на інфраструктуру [20].

Під час проєктування системи моніторингу важливим аспектом є реалізація зручного та доступного інтерфейсу користувача, який дозволить взаємодіяти з системою без залучення командного рядка або спеціальних знань у галузі програмування. Одним із сучасних підходів до швидкого створення графічних веб-інтерфейсів є використання фреймворку Streamlit, що забезпечує просту інтеграцію з Python-кодом і надає повноцінні засоби для динамічного відображення результатів аналітики.

Перевагою Streamlit є можливість побудови інтерфейсу без необхідності створення окремого веб-сервера чи фронтенду на основі HTML, CSS або JavaScript. Всі компоненти — від кнопок і форм до таблиць і графіків — можуть бути створені безпосередньо в Python-коді з мінімальними витратами часу. Це значно скорочує цикл розробки і дозволяє зосередитись на логіці обробки даних.

Крім простоти реалізації, Streamlit забезпечує інтерактивність у режимі реального часу. Користувач може завантажити вхідні дані, вибрати параметри аналізу (наприклад, тип моделі чи рівень чутливості), отримати результати у вигляді таблиці та одразу ж зберегти їх у файл. Такий підхід не тільки підвищує прозорість роботи системи, а й робить її придатною для використання в умовах обмеженого доступу до інфраструктури — наприклад, у навчальних або дослідницьких лабораторіях.

Важливою особливістю є також те, що інтерфейс може працювати локально, без розгортання на віддаленому сервері, що спрощує експлуатацію у невеликих організаціях або під час тестування [20]. За потреби його можна швидко масштабувати або інтегрувати з іншими компонентами — наприклад, додати сповіщення, графіки в реальному часі або авторизацію.

Вибір Streamlit як основи для побудови графічного інтерфейсу системи повністю відповідає критеріям швидкості розробки, зручності користування, прозорості взаємодії та гнучкості у подальшому розширенні функціоналу. Він гармонійно доповнює обрану архітектуру системи та дозволяє реалізувати повний цикл аналізу трафіку — від завантаження даних до прийняття рішень — в одному інтегрованому середовищі.

1.5. Висновки до розділу 1

У першому розділі було здійснено всебічний аналіз проблеми виявлення аномалій у мережевому трафіку, розглянуто класифікацію можливих відхилень, методологічні підходи до їх виявлення, а також сучасні інструментальні рішення,

що реалізують ці підходи на практиці. Встановлено, що аномалії в комп'ютерних мережах можуть набувати як явного (об'ємного чи структурного), так і прихованого характеру, що вимагає багаторівневої системи аналізу поведінки трафіку.

Проведене порівняння трьох основних класів методів — сигнатурного, статистичного та заснованого на машинному навчанні — засвідчило, що найбільш перспективними з точки зору адаптивності, стійкості до нових загроз та здатності працювати без попередньої розмітки є методи навчання без учителя. Сигнатурні системи виявились ефективними лише у вузькоспеціалізованих сценаріях із відомими загрозами, а статистичні підходи — чутливими до змін у профілі трафіку.

На основі аналізу інструментів (Snort, Suricata, Zeek, PyOD) обґрунтовано доцільність побудови гібридної системи, яка поєднує в собі елементи статистичного аналізу з моделями машинного навчання. Такий підхід дозволяє гнучко адаптувати систему до конкретного середовища, обробляти неструктуровані потоки даних і підтримувати автоматичне виявлення як відомих, так і нових відхилень у поведінці мережі.

Вибір алгоритмів Isolation Forest і HBOS як ядра системи базується на їх здатності ефективно працювати з багатовимірними числовими даними, високій обчислювальній ефективності та відсутності потреби у попередньому маркуванні. Водночас рішення про використання мови програмування Python та інтерфейсу Streamlit як основи для реалізації обґрунтовано з погляду швидкості розробки, гнучкості інтеграції та мінімального порогу входу для користувача.

Узагальнюючи вищенаведене, можна зробити висновок, що розробка ефективної системи виявлення аномалій у мережевому трафіку повинна спиратись на сучасні алгоритми машинного навчання у поєднанні з простими, але інформативними ознаками потоків, гнучку архітектуру, адаптивну обробку та зручне візуальне представлення результатів. Саме така концепція буде реалізована у наступних розділах цієї роботи.

РОЗДІЛ 2. ПОСТАНОВКА ЗАДАЧІ ТА ПРОЄКТУВАННЯ СИСТЕМИ

2.1. Формалізація задачі виявлення аномалій у мережевому трафіку

Задача виявлення аномалій у мережевому трафіку полягає у виявленні таких потоків даних, які за своїми характеристиками суттєво відрізняються від більшості інших і можуть потенційно свідчити про порушення безпеки, технічні збої або аномальну поведінку користувачів. Важливою особливістю є те, що в реальному трафіку, як правило, відсутня попередня розмітка (мітки) — тобто невідомо, які саме потоки є нормальними, а які — шкідливими. Отже, задача належить до класу навчання без учителя (unsupervised anomaly detection) [21].

Формально, припустимо, що кожен мережевий потік (flow) можна описати вектором ознак:

$$x_i \in R^d, i = 1,2, \dots, n \tag{1.1}$$

Де $x_i = [x_{i1}, x_{i2}, \dots, x_{id}]$ — числове представлення потоку за певними статистичними метриками (наприклад, кількість пакетів, обсяг трафіку, тривалість тощо), а d — кількість ознак.

Завдання полягає у побудові функції:

$$f : R^d \to \{0,1\} \tag{1.2}$$

такої, що:

$$f(x_i) = \begin{cases} 0, \text{якщо потік вважається нормальним,} \\ 1, \text{якщо потік розпізнано як аномальний.} \end{cases}$$

У рамках розроблюваної системи обрано два типи моделей:

1. Isolation Forest, який моделює нормальний простір через випадкову ізоляцію вибірок, і видає для кожного потоку аномальний бал. Потоки з найвищими значеннями цього балу класифікуються як аномальні на основі порогового рівня α (параметр contamination).

2. HBOS (Histogram-Based Outlier Score), який аналізує розподіли кожної ознаки окремо й оцінює рідкість потоку за гістограмами, сформованими на основі нормального трафіку.

Таким чином, вся множина вхідних потоків перетворюється на матрицю:

$$X = \begin{bmatrix} x_1 \\ x_2 \\ . \\ . \\ . \\ x_n \end{bmatrix} \in R^{n \times d} \tag{1.3}$$

і подається на вхід алгоритму аномального аналізу, який повертає вектор рішень:

$$y = [y_1, y_2, \dots, y_n], y_i \in \{0,1\}. \tag{1.4}$$

З погляду прикладної реалізації, система передбачає:

– попередню обробку сирих пакетів у формі потоків, обчислення ознак (feature engineering),
– використання обраної моделі для присвоєння кожному потоку ознаки «аномальний/неаномальний»,
– виведення результату через інтерактивний GUI.

Задача виявлення аномалій зводиться до класифікації потоків у просторі ознак за відсутності міток, із використанням моделей, здатних фіксувати нетипові відхилення на основі внутрішньої структури даних [22].

Задача виявлення аномалій у мережевому трафіку суттєво відрізняється від класичних задач класифікації, які традиційно формулюються як навчання з учителем. У типових класифікаційних сценаріях передбачається наявність збалансованого набору прикладів для кожного класу (наприклад, «легітимна» та «шкідлива» сесія), що дозволяє безпосередньо оптимізувати розділяючу

гіперплощину, мінімізуючи помилки. Натомість у задачах виявлення аномалій така передумова є нереалістичною.

Найперше, слід зазначити, що аномальні приклади у трафіку є рідкісними або повністю відсутніми у доступних даних, особливо на етапі початкового навчання системи. У багатьох випадках аномалії мають унікальний характер, не повторюються у часі або змінюють свою структуру з кожним новим інцидентом. Унаслідок цього побудова навчального набору даних із достовірною маркіровкою є неможливою або потребує значних людських і часових ресурсів. Тому замість навчання на двох класах система має формувати уявлення про нормальну поведінку і трактувати відхилення як потенційно шкідливі [23].

Ще однією характерною рисою є асиметрія в оцінці помилок. На відміну від звичайної класифікації, де помилки обох типів (хибнопозитивні й хибнонегативні) вважаються рівнозначними, у випадку з виявленням аномалій пріоритет надається мінімізації помилок другого роду (false negatives). Іншими словами, краще хибно ідентифікувати нормальний потік як підозрілий (тобто мати хибнопозитив), ніж пропустити реальну загрозу, що може призвести до значних збитків або компрометації системи.

Крім того, системи аномального аналізу мають справу з явищем concept drift — зміною розподілу нормального трафіку з часом. У міру того як змінюється конфігурація мережі, поведінка користувачів або програмне забезпечення, ознаки, які раніше вважалися типовими, можуть втратити свою релевантність. Це створює додаткову потребу в адаптивних моделях або періодичному перенавчанні.

Особливу увагу також слід звернути на те, що більшість сучасних моделей машинного навчання повертають не бінарне рішення, а неперервне значення — «рівень аномальності». Перетворення цього значення у фінальне рішення (0 або 1) вимагає вибору порогу. У задачах без учителя цей поріг визначається заздалегідь (наприклад, через параметр contamination), і його вибір прямо впливає на баланс між чутливістю та специфічністю моделі.

Задача виявлення аномалій є суттєво складнішою та менш формалізованою у порівнянні з класичною класифікацією [24]. Її ефективне розв’язання вимагає ретельно підібраних моделей, правильної підготовки ознак та гнучких інструментів для адаптації до нових типів поведінки.

Ефективність алгоритмів виявлення аномалій істотно залежить від того, наскільки коректно сформовано простір ознак, тобто які характеристики мережевих потоків обрано для побудови векторного представлення. У задачах аналізу трафіку важливо досягти балансу між інформативністю ознак, їхньою стійкістю до зміни середовища та технічною можливістю обчислення без повного доступу до вмісту пакетів.

У межах цієї роботи було вирішено використовувати агреговані статистичні метрики, які легко отримуються на основі метаінформації про мережеві пакети [25]. Такий підхід дозволяє працювати навіть із зашифрованим трафіком (наприклад, HTTPS або VPN), оскільки аналізу підлягають не дані вмісту, а зовнішні характеристики потоку.

До обраних ознак належать:

1. Кількість пакетів у потоці — дозволяє виявити нестандартну інтенсивність або дуже короткі/довгі з’єднання.
2. Загальний обсяг байтів — вказує на аномальні об’єми переданої інформації, характерні, наприклад, для ексфільтрації.
3. Тривалість потоку — час між першим і останнім пакетом, може сигналізувати про затримки або нестабільні з’єднання.
4. Середній розмір пакета — корисна ознака для виявлення тунелювання або генеративного трафіку з фіксованою структурою.

Всі ці ознаки є числовими, лінійно масштабованими та не потребують складного парсингу. Такий набір ознак є достатнім для побудови гнучкої моделі на основі алгоритмів машинного навчання без учителя. Крім того, важливою перевагою обраних метрик є їхня слабка корельованість, що створює передумови для застосування простих моделей, зокрема HBOS, який ефективно працює лише за умови незалежності ознак [26].

Водночас обрані ознаки легко обчислюються у режимі реального часу, що критично важливо для продуктивних систем. Вони не обтяжують систему надлишковими обчисленнями, як це було б у випадку використання глибоких мережевих фіч (наприклад, n-грам або заголовків високого рівня).

Обраний набір ознак є виваженим компромісом між простотою реалізації, обчислювальною ефективністю та здатністю відображати ключові особливості поведінки мережевих потоків. Він формує придатну основу для подальшого виявлення відхилень із використанням адаптивних моделей машинного навчання.

2.2. Архітектура системи: модулі збору, обробки, аналізу, звітності

Розробка системи виявлення аномалій у мережевому трафіку вимагає чітко структурованої архітектури, яка забезпечує повноцінний цикл обробки даних — від надходження трафіку до формування остаточного рішення щодо його нормальності або підозрілої поведінки. На етапі проєктування було визначено доцільність модульної побудови, що передбачає незалежність ключових етапів обробки: збір, агрегація, аналіз, представлення результатів.

Загальна структура системи реалізує послідовну обробку вхідного трафіку у вигляді лінійного конвеєра, де кожен модуль виконує строго визначену функцію і передає результат на наступний етап. Такий підхід полегшує налагодження, масштабування та інтеграцію з іншими системами (зокрема з зовнішніми джерелами даних або інтерфейсами користувача) [27].

Логіка взаємодії між компонентами подана у вигляді блок-схеми на рис. 2.1. Вона охоплює основні функціональні вузли системи, зокрема:

1. Джерело трафіку — вхідні дані у вигляді .pcap-файлів або потоки в реальному часі.
2. Модуль агрегації — об’єднання пакетів у flows на основі 5-tuple та обчислення ознак.

3. Алгоритмічне ядро — модель аномального аналізу, що оцінює кожен потік.
4. Блок візуалізації — таблиця результатів, маркування аномалій, експорт або повідомлення.

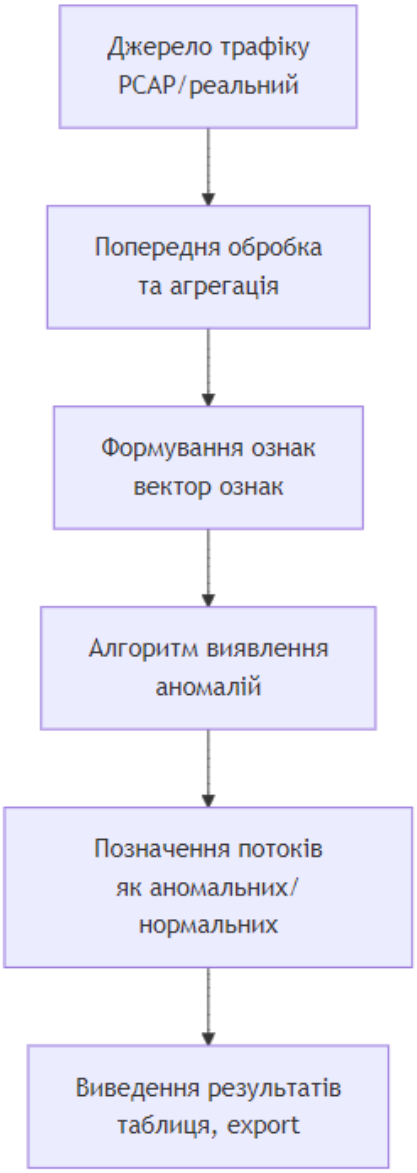


Рис. 2.1 – Загальна структура системи виявлення аномалій у трафіку

Дана архітектура дозволяє масштабувати систему як у напрямі підвищення продуктивності (наприклад, шляхом розпаралелювання агрегації), так і у напрямі функціонального збагачення — шляхом додавання нових моделей або каналів виводу результатів.

Кожен з компонентів архітектури системи виконує визначену роль у загальному циклі обробки мережевого трафіку. Їхнє розмежування дозволяє реалізувати модульну, незалежну структуру, в якій кожна підсистема може бути вдосконалена або замінена без суттєвих змін у загальній логіці роботи системи [28].

1. Підсистема збору трафіку. Основне призначення цього модуля — забезпечити надходження вхідних мережевих даних у систему. Як джерело трафіку може виступати файл із попередньо збереженим дампом у форматі .pcap або дані, захоплені в реальному часі через мережевий інтерфейс. На цьому етапі не здійснюється глибокий аналіз, а лише фіксується низькорівнева інформація про пакети (IP, TCP/UDP заголовки, час надходження тощо).
2. Підсистема агрегації та формування потоків. На другому етапі відбувається агрегування сирих пакетів у мережеві потоки (flows), які формуються за п’ятіркою ознак: IP-адреса джерела, IP-адреса призначення, порти обох сторін та транспортний протокол. Потоки є основною одиницею подальшого аналізу. Для кожного потоку обчислюється низка базових статистичних показників: кількість пакетів, обсяг трафіку, тривалість сесії, середній розмір пакета. Ці показники формують ознаковий вектор, який надалі передається в модель.
3. Алгоритмічне ядро (модель виявлення аномалій). У цій підсистемі відбувається основна аналітична обробка даних. На вхід моделі подається матриця ознак потоків, після чого кожен об’єкт оцінюється з точки зору його відповідності типовому розподілу. Визначальним параметром є рівень аномальності (anomaly score), який порівнюється з порогом. Залежно від обраного алгоритму (наприклад, Isolation Forest або HBOS), обчислення виконується з різною складністю, але в обох випадках результатом є позначення кожного потоку як нормального або аномального.

4. Підсистема виведення та візуалізації результатів. Завершальний етап роботи системи полягає в інтерпретації результатів для користувача. Потоки, що пройшли аналіз, відображаються у табличному вигляді з зазначенням ключових метрик і мітки аномальності. Користувач має змогу здійснювати фільтрацію, сортування та експорт отриманих результатів. У перспективі можливе розширення цієї підсистеми за рахунок графічних візуалізацій, інтеграції з системами сповіщення або аналітичними панелями (dashboard).

Усі перелічені підсистеми функціонують узгоджено відповідно до схеми, поданої на рис. 2.1, забезпечуючи наскрізну обробку трафіку — від джерела до результатів аналізу. Завдяки такій структурі система може бути легко адаптована до нових вимог або інтегрована в більші рішення, зокрема у склад корпоративного моніторингового середовища.

Модульна структура системи моніторингу мережевого трафіку була сформована з урахуванням перспективи її подальшого розвитку, а також вимог до масштабування, гнучкості налаштувань і змін у технологічному середовищі. Всі компоненти архітектури — від збору даних до відображення результатів — спроєктовано таким чином, щоб забезпечити можливість їх незалежного оновлення, заміни або інтеграції з іншими інформаційними системами [29].

Однією з ключових переваг архітектури є можливість легкого додавання нових алгоритмів виявлення аномалій. Завдяки тому, що вся логіка аналізу побудована навколо абстрактного подання потоку у вигляді вектора ознак, до системи можуть бути інтегровані як класичні методи (наприклад, One-Class SVM, LOF), так і сучасні нейромережеві підходи (автоенкодери, моделі на основі LSTM). Розширення алгоритмічного ядра не потребує змін в інших частинах системи, оскільки взаємодія відбувається через уніфікований інтерфейс — матрицю ознак.

Крім того, система може бути адаптована до роботи в онлайн-режимі. Поточна архітектура передбачає обробку заздалегідь зібраних дампів (офлайн-аналіз), однак при зміні джерела трафіку на потоковий захоплювач через

мережевий інтерфейс можливо забезпечити постійне надходження нових даних. При цьому більшість обчислювальних модулів залишаються незмінними, оскільки працюють з уже сформованими потоками незалежно від способу їх отримання.

Важливою характеристикою є також розширюваність підсистеми виводу результатів. У її поточному вигляді користувач взаємодіє з системою через табличну візуалізацію [29]. Однак у майбутньому можливе додавання функцій експорту до зовнішніх баз даних, генерації звітів у форматі PDF, надсилання сповіщень у месенджери (наприклад, Telegram) або навіть підключення до систем керування подіями інформаційної безпеки (SIEM). Така інтеграція розширює функціонал системи з локального інструменту до повноцінного модуля кіберзахисту.

Архітектура проєктованої системи демонструє високу адаптивність як до змін у середовищі використання, так і до функціональних оновлень. Це створює основу не лише для реалізації демонстраційного або навчального прототипу, а й для розгортання у реальних умовах, із подальшою інтеграцією в інфраструктуру мережевого моніторингу.

2.3. Алгоритм виявлення аномалій

Процес виявлення аномалій у мережевому трафіку можна подати у вигляді послідовного алгоритму, який оперує агрегованими потоками даних (flows) після первинної обробки та формування ознак. Основна мета цього алгоритму — оцінити, наскільки кожен потік відрізняється від типового (нормального) трафіку, та винести рішення про його приналежність до аномального або звичайного класу.

Алгоритм функціонує у декілька послідовних етапів:

1. Надходження вхідних даних. Мережевий трафік, попередньо агрегований у потоки, подається на вхід системи у вигляді матриці ознак розмірності $n \times d$, де n — кількість потоків, а d — кількість ознак.
2. (Опціонально) нормалізація ознак. Для деяких алгоритмів (зокрема HBOS) значення ознак можуть бути попередньо нормалізовані або масштабовані, щоб уникнути впливу великого розмаху значень у різних вимірах.
3. Обчислення аномального бала. Кожен потік аналізується обраним алгоритмом (Isolation Forest або HBOS), який повертає числовий показник аномальності — outlier score. Це значення є неперервним і відображає віддаленість потоку від типового розподілу.
4. Порогове розділення. На основі заздалегідь визначеного порогу (наприклад, значення параметра contamination) відбувається класифікація: потоки з високим балом вважаються аномальними, інші — нормальними.
5. Маркування результату. Кожному потоку присвоюється бінарна мітка $y_i \in \{0,1\}$ де 1 означає аномальний трафік. Результати передаються до підсистеми виведення.

Ця логіка подана на рис. 2.2, де відображено повний шлях обробки одного потоку в межах системи.

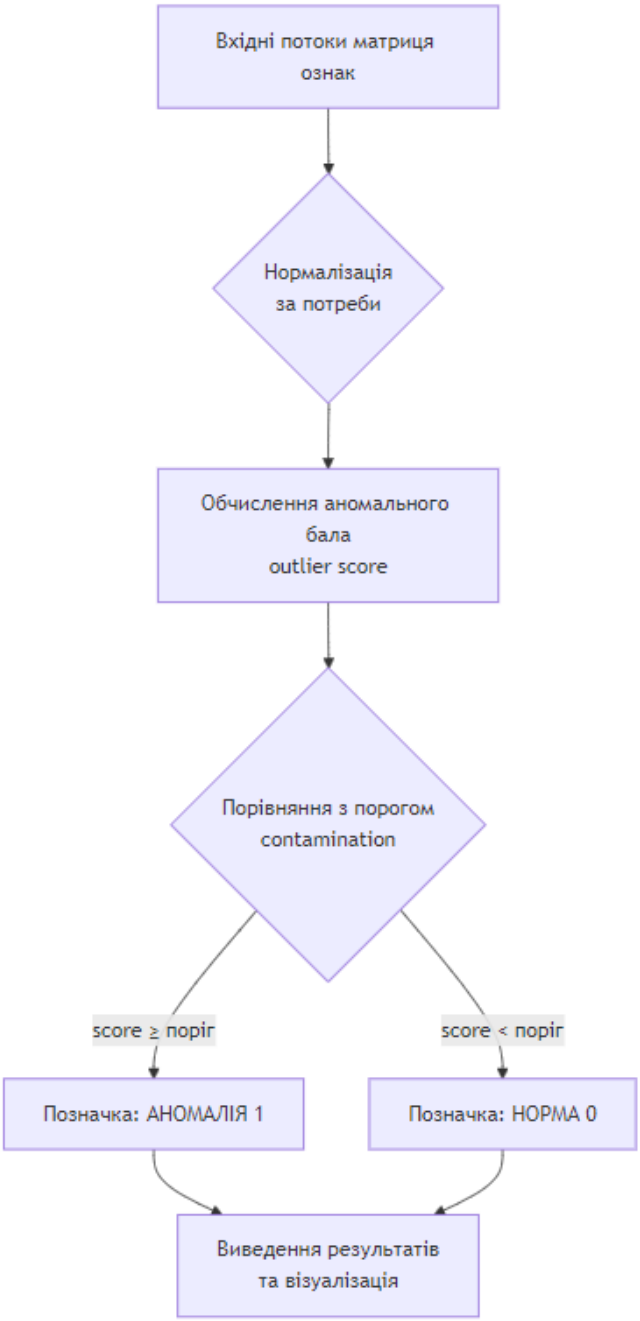


Рис. 2.2 – Узагальнений алгоритм виявлення аномалій у мережевому потоці

Такий узагальнений алгоритм є придатним для реалізації різних методів виявлення аномалій, оскільки формалізує лише загальні етапи — без прив’язки до конкретного механізму обчислення аномального бала. У наступних пунктах будуть описані два варіанти реалізації цього підходу: через моделі Isolation Forest і HBOS [30].

Алгоритм Isolation Forest є сучасним ефективним методом для виявлення аномалій у багатовимірних наборах даних і належить до категорії моделей навчання без учителя. Його ключова ідея полягає у випадковій ізоляції об’єктів, що дозволяє розпізнавати відхилення без необхідності попереднього моделювання нормального розподілу [31].

На відміну від більшості алгоритмів, які намагаються оцінити густину розподілу даних або відстані між об’єктами, Isolation Forest працює за принципом того, що аномальні об'єкти легше ізолюються, ніж ті, що належать до щільної, зв’язаної маси «нормальних» точок. Такий об'єкт має або незвичні значення ознак, або перебуває далеко від центру маси вибірки. Основні етапи роботи алгоритму:

1. Формування випадкових дерев (isolation trees). Для кожної підвибірки даних будується бінарне дерево. На кожному рівні дерева випадковим чином обирається ознака та її порогове значення. Вибірка рекурсивно розбивається на частини, доки всі точки не будуть ізольовані або не буде досягнуто граничної глибини дерева.
2. Оцінка глибини ізоляції. Для кожного об’єкта фіксується кількість розділень (глибина), необхідна для його ізоляції. Аномалії, як правило, ізолюються швидше, а отже мають меншу середню глибину.
3. Обчислення outlier score. Аномальний бал $s(x) \in [0,1]$ для кожного об’єкта xxx розраховується на основі середньої глибини ізоляції у всіх деревах. Формально, цей показник визначається як:

$$s(x) = 2^{-\frac{E(h(x))}{c(n)}} \tag{2.1}$$

Де $E(h(x))$ — очікувана глибина для об’єкта $x, a\ c(n)$ — нормалізувальна константа, що залежить від розміру вибірки nnn.

4. Порогове класифікування. Якщо score > threshold (визначеного через параметр contamination), об’єкт вважається аномальним. Інакше — нормальним.

Переваги використання Isolation Forest у задачі моніторингу трафіку:

- Не потребує нормалізації ознак: оскільки алгоритм працює з відсіченням по випадкових значеннях, масштаби не мають критичного значення.
- Масштабованість: лінійна складність за обсягом даних та ознак дозволяє застосовувати модель на великих наборах потоків.
- Нечутливість до шуму: випадкова природа ізоляції робить модель стійкою до одиничних сплесків у даних.
- Універсальність: підходить для будь-яких числових ознак і не потребує апріорного знання про тип трафіку.

Isolation Forest є ефективною базовою моделлю для виявлення відхилень у мережевих потоках, особливо у випадках, коли структура нормального трафіку є складною або динамічно змінюється [32].

Алгоритм HBOS (Histogram-Based Outlier Score) є простим, ефективним і добре масштабованим методом виявлення аномалій, який базується на аналізі розподілу кожної ознаки незалежно. Його ключова перевага полягає в тому, що він не потребує навчання у класичному розумінні, а замість цього будує одновимірні гістограми для кожного параметра і використовує їх для виявлення рідкісних, нетипових значень [32].

HBOS ґрунтується на припущенні про некорельованість ознак, тобто кожна ознака розглядається окремо, без урахування взаємозв’язків. Це дає змогу значно спростити обчислення та забезпечити дуже високу швидкодію навіть на великих вибірках. Основні етапи роботи алгоритму:

1. Побудова гістограм для кожної ознаки. Для кожного виміру (ознаки) визначається діапазон значень, який ділиться на інтервали (бінінг). Визначається частота появи значень у кожному інтервалі, тобто формується гістограма, що моделює розподіл ознаки.

2. Оцінка частот для кожного об’єкта. Для кожного потоку, який має конкретне значення ознаки, визначається, в який інтервал він потрапляє, і яка ймовірність цього інтервалу. Рідкісні значення мають меншу частоту і, відповідно, вищий внесок у загальний outlier score.
3. Обчислення загального балу аномальності. Для кожного потоку всі одновимірні оцінки комбінуються, зазвичай шляхом підсумовування логарифмів обернених частот:

$$s(x) = \sum_{j=1}^{d} \log(\frac{1}{P_j(x_j)}) \quad (2.2)$$

Де $P_j(x_j)$ — ймовірність появи значення xjx_jxj ознаки jjj у відповідній гістограмі.

Класифікація за порогом. Потоки з високим сумарним балом вважаються аномальними. Поріг визначається наперед (через contamination) або емпірично.

Переваги HBOS:

- Висока швидкодія: алгоритм виконує лише одновимірні обчислення без побудови моделі чи дерев.
- Придатність для великих даних: складність обчислень є майже лінійною відносно кількості потоків.
- Простота реалізації: не вимагає вибору складних гіперпараметрів чи процедур оптимізації.
- Зрозумілість результату: кожна ознака робить вклад у фінальний бал, що спрощує аналіз впливу окремих параметрів.

Проте варто враховувати, що ефективність HBOS може знижуватися у випадках сильно корельованих ознак або в разі складної внутрішньої структури нормального трафіку. У таких випадках його варто застосовувати як доповнення до більш гнучких моделей або в якості базового фільтра.

HBOS добре підходить для задач первинного виявлення відхилень у мережевих потоках, особливо коли важливі простота, швидкодія та відсутність потреби у навчанні.

2.4. Вибір ознак: агрегація за flow-ключами, прості метрики

Для забезпечення ефективного аналізу мережевого трафіку на рівні поведінкових шаблонів доцільно переходити від обробки окремих пакетів до роботи з мережевими потоками (flows). Потік — це логічна одиниця, яка репрезентує послідовність взаємодії між двома кінцевими точками в мережі та агрегує відповідні пакети у цілісну сесію. Такий підхід суттєво знижує обсяг даних, які потрібно аналізувати, водночас зберігаючи ключову інформацію про характер взаємодії [33].

У межах даної системи під потоком розуміється множина пакетів, що мають однакове значення flow-ключа, який формується на основі п’яти параметрів:

- IP-адреса джерела,
- IP-адреса призначення,
- порт джерела,
- порт призначення,
- транспортний протокол (TCP, UDP, ICMP).

Цей підхід відомий як 5-tuple ідентифікація, і він є стандартним у задачах моніторингу та класифікації трафіку. У випадку TCP-з’єднань, які мають явну сесійну семантику (через флаг SYN/FIN), такий підхід дозволяє чітко виділяти початок і завершення потоку. У разі протоколів без встановлення з’єднання (UDP, ICMP), агрегація здійснюється за часовим вікном або тривалістю бездіяльності між пакетами, що належать до одного ключа [33].

Основною перевагою агрегації за flow-ключем є можливість переходу від низькорівневої структури трафіку до більш абстрактного, поведінкового рівня аналізу. Це дозволяє:

- зменшити кількість об’єктів для аналізу в десятки або сотні разів;
- зберегти логічну зв’язність взаємодій;
- забезпечити стабільні умови для побудови ознак (features), придатних до машинного аналізу.

Формування потоків є обов’язковим етапом перед векторизацією, оскільки саме на рівні потоку можливо обчислити статистичні метрики: кількість пакетів, загальний обсяг, тривалість, середній розмір тощо.

Після формування мережевих потоків за flow-ключами наступним етапом є побудова ознак, що кількісно описують поведінку кожного потоку. У задачі виявлення аномалій особливо важливо використовувати такі метрики, які є простими в обчисленні, універсальними для різних типів трафіку та інформативними з точки зору поведінкового аналізу [34].

Для реалізованої системи було обрано чотири основні ознаки, що відображають ключові властивості кожного потоку. Ці метрики базуються виключно на заголовках пакетів і не потребують аналізу вмісту, що дозволяє застосовувати їх навіть у випадках із зашифрованим трафіком (наприклад, HTTPS або VPN-сесії). Такий підхід гарантує як незалежність від протоколів високого рівня, так і відповідність вимогам конфіденційності. У таблиці 2.1 наведено обрані ознаки разом із їхнім призначенням.

Таблиця 2.1

Основні ознаки мережевих потоків та їх призначення

№	Назва ознаки	Позначення	Опис / Інтерпретація
1	Кількість пакетів	packet_count	Відображає обсяг взаємодії; аномально високі чи низькі значення можуть вказувати на сплески або сканування
2	Загальний обсяг	byte_count	Сумарний обсяг трафіку у потоці; використовується для виявлення великих передач (наприклад, ексфільтрації)

Продовження таблиці 2.1

3	Тривалість потоку	duration	Час між першим і останнім пакетом у потоці; нетипові значення можуть свідчити про затримки, нестабільність або автоматизовану активність
4	Середній розмір	avg_pkt_size	Характеризує тип пакетів у потоці; може вказувати на протокольну специфіку або нестандартну генерацію трафіку

Вибір саме цих ознак зумовлений прагненням до оптимального балансу між складністю та ефективністю. Вони охоплюють як інтенсивність (кількість), так і обсяг (байти), час (тривалість) і структуру (розмір пакета) — чотири базові аспекти потоку, які є достатніми для первинного виявлення відхилень.

Завдяки тому, що всі ознаки мають числову природу та одновимірне представлення, вони добре підходять для подальшого аналізу як у моделях типу Isolation Forest, так і в простих статистичних підходах на кшталт HBOS.

У задачах виявлення аномалій за допомогою методів машинного навчання критичне значення має не лише кількість і тип ознак, а й їхні властивості, які визначають ефективність роботи алгоритмів. Обраний набір із чотирьох базових статистичних метрик (кількість пакетів, загальний обсяг, тривалість, середній розмір) демонструє низку переваг у контексті моделей без учителя.Оскільки більшість моделей машинного навчання (особливо статистичних) чутливі до мультиколінеарності, важливо, щоб ознаки не дублювали одна одну. У даному випадку, хоча між байтами й кількістю пакетів може спостерігатися часткова залежність, наявність додаткових ознак, таких як середній розмір чи тривалість, розширює можливості диференціації між потоками. Це особливо важливо для таких моделей, як HBOS, що обробляють кожну ознаку окремо [35].

Усі обрані ознаки є числовими, неперервними та додатними, що забезпечує зручність при використанні алгоритмів, які не потребують перетворення категоріальних даних або масштабування до фіксованих діапазонів. Це дає змогу уникати додаткових етапів підготовки даних, спрощуючи структуру системи та підвищуючи її продуктивність.

Ознаки не залежать від змісту пакетів, а лише від їхніх мета-параметрів. Це означає, що система здатна працювати навіть у випадках, коли трафік шифрується (наприклад, HTTPS, TLS, VPN), або коли використовується нестандартна передача даних. Такі властивості дозволяють застосовувати моделі в універсальних середовищах без потреби в глибокій інспекції пакетів (DPI).

Вибрані ознаки легко обчислюються, навіть при великій кількості потоків. Це дозволяє забезпечити лінійну масштабованість системи й використовувати моделі типу Isolation Forest, які розраховані на ефективну роботу з великими масивами даних [35].

На відміну від семантичних ознак, що можуть змінюватися внаслідок оновлень протоколів або зміни поведінки користувачів, прості статистичні метрики залишаються актуальними впродовж тривалого часу. Це підвищує життєвий цикл моделі та зменшує потребу у її повторному перенавчанні.

Обраний набір ознак формує якісну ознакову базу для побудови систем виявлення аномалій у мережевому трафіку, що поєднує простоту реалізації з ефективністю при обробці складних та динамічних даних. Це створює основу для подальшого успішного використання як статистичних, так і ансамблевих моделей машинного навчання.

2.5. Функціональні та технічні вимоги до MVP

Система (MVP) виявлення аномалій у мережевому трафіку повинна реалізовувати базову послідовність операцій, необхідних для повноцінного виконання ключових функцій: завантаження вхідних даних, їх обробка, аналіз за допомогою алгоритмів машинного навчання, а також представлення результатів користувачеві. Зважаючи на це, було сформульовано набір обов’язкових функціональних вимог до першої версії системи.

Система повинна дозволяти користувачеві обрати та завантажити файл мережевого дампу у форматі .pcap, отриманий із зовнішнього джерела

(наприклад, Wireshark, tcpdump або інше програмне забезпечення для захоплення трафіку). У першій версії передбачається офлайн-режим роботи, тобто аналіз вже збережених сесій, без прямого підключення до мережевого інтерфейсу.

Після завантаження .pcap-файлу система має автоматично агрегувати пакети у потоки за flow-ключами (5-tuple), а також розрахувати базові ознаки: кількість пакетів, загальний обсяг трафіку, тривалість потоку та середній розмір пакета. Ці параметри слугують основою для подальшого аналізу.

Користувач повинен мати можливість обрати один із двох алгоритмів виявлення аномалій:

– Isolation Forest, що забезпечує більш точну оцінку відхилень,
– HBOS, як простіший і швидший метод.

Для обох моделей також необхідно передбачити можливість встановлення рівня чутливості через параметр contamination (частка потенційних аномалій у даних). Результати аналізу повинні бути представлені у вигляді інтерактивної таблиці, що містить:

– значення ознак для кожного потоку,
– мітку «аномальний/нормальний» (0 або 1),
– можливість фільтрування, сортування та експорту у CSV.

Інтерфейс повинен бути інтуїтивно зрозумілим і не вимагати від користувача знань програмування. У першій версії використовується односторінкове веб-застосування, яке запускається локально та дозволяє виконувати всі основні дії в кілька кліків.

Система в MVP-форматі повинна забезпечити повний цикл аналізу мережевого трафіку — від завантаження до результату — у зручному для користувача середовищі, з можливістю базового налаштування параметрів аналізу [35].

Реалізація MVP-системи виявлення аномалій у мережевому трафіку потребує визначення технічних параметрів середовища, в якому вона повинна функціонувати. Це включає операційні та апаратні вимоги, вибір мови

програмування та бібліотек, а також вимоги до інсталяції й автономності виконання.

Розробка системи передбачена на основі мови Python 3.10+, як індустріального стандарту для задач машинного навчання, обробки даних і швидкого прототипування. Робоче середовище не потребує спеціалізованих серверів або хмарної інфраструктури — система повинна повноцінно функціонувати на звичайній локальній машині з встановленим Python.

Для повноцінного функціонування необхідні такі основні бібліотеки:

- scapy — для зчитування пакетів із pcap-файлів;
- pandas, numpy — для обробки табличних даних та масивів ознак;
- scikit-learn — для реалізації моделі Isolation Forest;
- pyod — для реалізації алгоритму HBOS;
- streamlit — для побудови графічного інтерфейсу.

Оскільки система працює в офлайн-режимі та не використовує ресурсоємних моделей, вимоги до апаратного забезпечення залишаються мінімальними:

- Операційна система: Windows 10/11, Linux (Ubuntu), macOS;
- Процесор: Intel i3 / AMD Ryzen 3 або вище;
- Оперативна пам’ять: від 4 ГБ;
- Дисковий простір: не менше 500 МБ для програми та тимчасових файлів;
- Інтернет-підключення — лише для початкового встановлення бібліотек.

Використання графічного процесора (GPU) не є обов’язковим і не впливає на працездатність системи.

Усі компоненти системи повинні бути доступні для запуску у вигляді локального застосунку. Інтерфейс запускається через браузер за допомогою команди: «streamlit run gui_app.py».

Після запуску користувач взаємодіє із системою через локальний веб-інтерфейс без потреби у зовнішніх серверних компонентах. Це забезпечує повну автономність MVP, що є критично важливою перевагою у порівнянні з

веб-сервісами або хмарними рішеннями, які потребують додаткового налаштування.

Система передбачає використання відкритих інструментів, невимогливих до обчислювальних ресурсів, і дозволяє запускати програму на звичайних користувацьких пристроях без складної підготовки середовища.

Напрями подальшого розширення:

1. Підключення до live-трафіку. Розширення функціоналу за рахунок інтеграції з системним мережевим інтерфейсом дозволить здійснювати постійний моніторинг із короткою затримкою. Це відкриває можливість побудови SIEM-сумісної системи із компонентами реального часу.
2. Розширення набору ознак. Можливим є додавання нових ознак, зокрема часових (інтервали між пакетами), мережевих (TTL, TCP flags), або семантичних (тип DNS-запиту, HTTP-заголовки) — за умови доступу до відповідної інформації.
3. Інтеграція зі сповіщеннями та зберіганням. Додавання модулів автоматичного сповіщення (наприклад, Telegram-бот, email-алерти), а також збереження результатів у базу даних (наприклад, SQLite або PostgreSQL) дозволить створити повноцінну платформу для мережевої аналітики.
4. Інтерфейс візуалізації з графіками. Побудова гістограм частот, часових ліній активності або heatmap-аналізу джерел і протоколів покращить інтерпретованість результатів та зробить систему зручнішою для прийняття рішень.

У підсумку, хоча MVP закладає основу для створення більш масштабної та продуктивної системи, здатної до адаптації у реальних умовах мережевого моніторингу та інформаційної безпеки.

2.6. Висновки до розділу 2

У межах другого розділу було здійснено формалізацію задачі виявлення аномалій у мережевому трафіку, проєктування архітектури системи та визначення функціональних і технічних вимог. На основі проведеного аналізу сформовано цілісну концепцію побудови системи, що забезпечує автономну роботу з мережевими даними без потреби у зовнішніх джерелах чи мітках.

Формалізація задачі дозволила чітко визначити формат вхідних даних — мережеві потоки, подані у вигляді багатовимірних векторів ознак — а також математичну постановку задачі як бінарної класифікації у просторі ознак без вчителя. Було обґрунтовано доцільність використання моделей Isolation Forest та HBOS, які дозволяють ефективно працювати з непозначеними даними, мають високу обчислювальну ефективність та не потребують тривалого навчання.

Архітектура системи реалізована у вигляді послідовного ланцюга обробки: від захоплення трафіку (у форматі .pcap) до виведення результатів аналізу. Усі компоненти побудовані модульно, що забезпечує легке розширення системи, інтеграцію з іншими платформами та адаптацію до зміни вхідних даних або цільових алгоритмів. Представлена схема (рис. 2.1) фіксує логіку взаємодії між підсистемами, а алгоритм виявлення (рис. 2.2) — загальні принципи роботи моделей.

Особливу увагу було приділено вибору ознак. У системі використовуються прості, числові метрики потоків (кількість пакетів, обсяг, тривалість, середній розмір), що дозволяє здійснювати аналіз навіть у зашифрованому трафіку. Такий підхід забезпечує універсальність, стабільність у часі та сумісність із вибраними алгоритмами.

У підсумку сформульовано функціональні та технічні вимоги до системи, які забезпечують базову працездатність системи у форматі MVP. Водночас було запропоновано напрями подальшого розвитку, зокрема: інтеграція з live-трафіком, розширення набору ознак, зберігання результатів, побудова візуалізацій та сповіщення.

Проєктна частина дослідження завершена, і на її основі можна реалізувати повноцінну версію системи для виявлення аномалій у мережевому трафіку, що й буде розглянуто у наступному розділі.

РОЗДІЛ 3. РЕАЛІЗАЦІЯ ПРОТОТИПУ СИСТЕМИ

3.1. Середовище та стек технологій

У межах реалізації програмного прототипу системи виявлення аномалій у мережевому трафіку було обрано гнучке та доступне середовище розробки, яке забезпечує стабільну роботу, кросплатформеність [36] та мінімальні вимоги до апаратного забезпечення. Підхід базується на використанні відкритих інструментів, які є загальнодоступними і не потребують придбання ліцензій.

Розробка виконувалась у середовищі операційної системи Windows 11, хоча вся обрана програмна інфраструктура є сумісною також із Linux-дистрибутивами (зокрема Ubuntu) та macOS, що забезпечує платформену незалежність проєкту. Такий вибір зумовлений як міркуваннями зручності під час розробки, так і можливістю подальшого розгортання на різних системах без потреби у значних змінах коду.

Для створення та модифікації вихідних файлів використовувалося інтегроване середовище розробки Visual Studio Code, яке підтримує розширення для Python, роботу з віртуальними середовищами, інтеграцію з терміналом, контроль версій та зручну навігацію по проєкту. Додатково було використано вбудовані засоби налагодження, а також інструменти візуалізації графічного інтерфейсу через запуск локального вебсервера.

Особливу увагу було приділено ізоляції середовища виконання для запобігання конфліктам між залежностями та забезпечення повторюваності результатів. З цією метою було створено окреме віртуальне середовище на основі venv, у якому було встановлено лише необхідні пакети для даного проєкту. Такий підхід є стандартом у Python-розробці та сприяє простому відтворенню системи на іншому пристрої або в новому середовищі [37].

Усі скрипти реалізовано з використанням мови програмування Python версії 3.10, яка на момент реалізації проєкту поєднує стабільність ядра мови з

підтримкою сучасних бібліотек у галузі обробки даних та машинного навчання. Перевагами Python є його зрозумілий синтаксис, велика кількість спеціалізованих модулів, активна спільнота та підтримка різноманітних операційних систем.

Обране середовище розробки відповідає вимогам до реалізації прототипу дослідницького рівня, не обмежує у виборі платформ і дозволяє зосередитися безпосередньо на логіці аналізу мережевого трафіку без потреби у складній конфігурації інфраструктури [38].

Реалізація системи виявлення аномалій базується на застосуванні перевіреного та широко використовуваного стеку бібліотек, орієнтованих на обробку даних, аналіз мережевого трафіку, реалізацію алгоритмів машинного навчання та побудову інтерактивного інтерфейсу. Усі інструменти є відкритими (open source), активно підтримуються спільнотою та мають достатню документацію, що дозволило забезпечити як швидку розробку, так і просту відтворюваність.

Для обробки вхідного мережевого трафіку, представленого у форматі .pcap, було використано бібліотеку Scapy, яка дозволяє здійснювати низькорівневу обробку пакетів. Вона забезпечує доступ до полів заголовків протоколів TCP/IP, дозволяє фільтрувати, групувати та витягати необхідні параметри без потреби в зовнішніх інструментах. Scapy є гнучким інструментом, що підтримує як офлайн-режим (робота з дампами), так і live-захоплення трафіку.

Для представлення агрегованих потоків і обчислення статистичних ознак було використано бібліотеки Pandas та NumPy, які є стандартом у Python-екосистемі для обробки табличних та числових даних. Pandas дозволяє ефективно маніпулювати потоками як об'єктами типу DataFrame, забезпечуючи гнучкість при виведенні, фільтрації та агрегації результатів. NumPy [39], у свою чергу, забезпечує швидку математичну обробку векторних даних.

Алгоритмічне ядро системи реалізовано на основі двох бібліотек:

- Scikit-learn — для реалізації моделі Isolation Forest, яка вже входить до стандартного пакету цієї бібліотеки. Модель легко конфігурується,

підтримує параметр contamination та дозволяє виконувати аналіз без попереднього навчання.

- PyOD — спеціалізована бібліотека для задач виявлення аномалій. Вона надає велику кількість моделей, серед яких обрано HBOS (Histogram-Based Outlier Score) як простий та ефективний статистичний підхід до аналізу неструктурованих даних.

Для створення інтерфейсу користувача обрано фреймворк Streamlit, який дозволяє створювати інтерактивні веб-додатки без необхідності використання HTML, CSS чи JavaScript. Streamlit [40] забезпечує швидкий запуск локального застосунку через браузер, підтримує завантаження файлів, налаштування параметрів через слайдери та виведення результатів у таблиці або графіки. Завдяки своїй простоті, він є особливо корисним у задачах прототипування моделей машинного навчання.

Сумарно, обраний стек бібліотек охоплює всі компоненти проєкту — від обробки вхідних даних до інтерфейсної частини — і забезпечує можливість швидкої реалізації та масштабування без втрати стабільності або продуктивності.

Вибір технологічного стеку для реалізації системи виявлення аномалій у мережевому трафіку був зумовлений поєднанням кількох ключових факторів: доступністю, продуктивністю, простотою розгортання, підтримкою необхідної функціональності та сумісністю між компонентами. Усі використані інструменти належать до відкритого програмного забезпечення з активною спільнотою розробників і повною документаційною підтримкою, що дозволяє швидко адаптувати систему до змін середовища чи вимог.

Першим аргументом на користь обраного стеку є його кросплатформеність. Основна мова реалізації — Python — дозволяє запускати систему як у середовищі Windows, так і під Linux чи macOS без потреби в істотних модифікаціях [41]. Це відкриває можливості для гнучкого розгортання, зокрема у локальних мережах, лабораторних умовах або при інтеграції в більш складні інфраструктури.

Другим важливим фактором є відсутність складних залежностей. Усі бібліотеки легко встановлюються через менеджер пакетів pip, не потребують компіляції або складних серверних налаштувань. Система працює автономно, без підключення до зовнішніх баз даних або сервісів, що особливо актуально для сценаріїв з обмеженим доступом до мережі або конфіденційних середовищ.

Окрему увагу варто звернути на узгодженість обраних бібліотек між собою. Scapy органічно працює з pandas, дозволяючи швидко конвертувати результати обробки мережевого трафіку у табличну форму. Моделі машинного навчання з бібліотек scikit-learn і PyOD [42] приймають ці дані без додаткових перетворень, що зменшує кількість проміжного коду та знижує ймовірність помилок. Інтерфейс на базі Streamlit інтегрується з pandas-об’єктами для виводу результатів у вигляді таблиць, а також підтримує параметризацію моделей без складного програмування.

Також слід відзначити ефективність виконання на звичайних персональних комп’ютерах. Алгоритми, що використовуються в системі, не потребують значних обчислювальних ресурсів — аналіз виконується на середньостатистичному обладнанні з 4–8 ГБ оперативної пам’яті. Це робить систему придатною до розгортання в умовах обмежених ресурсів, без спеціалізованих серверів або GPU [43].

Обраний стек технологій забезпечує необхідний функціонал для повноцінного виявлення аномалій у трафіку, зберігаючи при цьому простоту, гнучкість і мінімальні вимоги до інфраструктури. Такий підхід дозволяє сфокусуватися безпосередньо на методах аналізу та інтерпретації результатів, не витрачаючи ресурси на надлишкову технічну інтеграцію.

3.2. Структура проєкту та короткий опис модулів

Структура проєкту реалізована відповідно до функціональної архітектури системи, що передбачає послідовне проходження даних через кілька логічно

відокремлених етапів: завантаження мережевого трафіку, формування ознак, виявлення аномалій та відображення результатів. Такий підхід дозволяє чітко відмежувати зони відповідальності кожного компонента, полегшує підтримку та розширення системи.

У проєкті реалізовано поділ на такі ключові файли та каталоги:

- main.py — точка входу до програми, ініціалізує запуск системи через графічний інтерфейс.
- traffic_loader.py — модуль обробки .pcap-файлів та агрегування трафіку у потоки.
- feature_engineering.py — модуль обчислення ознак на основі сформованих потоків.
- anomaly_detection.py — реалізація моделей виявлення аномалій (Isolation Forest, HBOS).
- gui_app.py — графічний інтерфейс користувача на базі Streamlit.
- /data/ — папка для зберігання вхідних файлів .pcap та результатів обробки.
- /resources/ — допоміжні дані або шаблони (якщо використовуються).
- requirements.txt — перелік усіх залежностей Python-проєкту для швидкої інсталяції середовища.

Така структура не лише забезпечує модульність реалізації, а й відповідає принципам розробки програм у галузі обробки даних. Кожен з компонентів системи може бути протестований або замінений окремо без порушення цілісності логіки.

На рис. 3.1 представлено ієрархію каталогів та основних файлів проєкту, що дозволяє візуально оцінити логіку його організації.

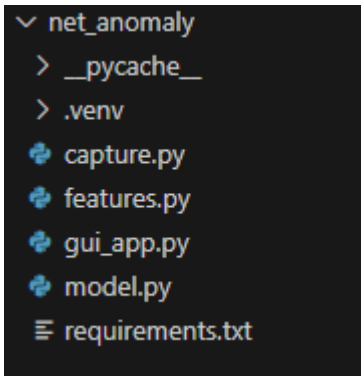


Рис. 3.1 – Структура проєкту в VS Code

Кожен модуль системи реалізує окрему частину логіки, що відповідає етапам обробки мережевого трафіку — від завантаження даних до отримання висновку щодо їхньої аномальності. Такий поділ дозволяє не лише зберігати структурованість реалізації, а й забезпечує гнучкість для майбутніх змін і розширень.

Нижче подано короткий опис основних функціональних модулів:

- Модуль traffic_loader.py. Призначений для обробки вхідних файлів мережевого трафіку у форматі .pcap. Зчитує пакети за допомогою бібліотеки Scapy [43], формує потоки на основі flow-ключів (IP, порт, протокол) та фіксує час початку і завершення кожного потоку. На виході формує структуру даних, з якою працюють наступні модулі.
- Модуль feature_engineering.py. Відповідає за обчислення статистичних ознак для кожного потоку. Зокрема, обчислюються такі метрики: кількість пакетів, сумарний обсяг трафіку, тривалість сесії, середній розмір пакета. Отримані ознаки формуються у форматі табличної структури (DataFrame), придатної для машинного аналізу.
- Модуль anomaly_detection.py. Містить реалізацію двох моделей виявлення аномалій — Isolation Forest (через scikit-learn) та HBOS (через pyod). Забезпечує обчислення аномального бала для кожного потоку, здійснює порогове розділення за параметром contamination, формує мітки аномальності.

- Модуль gui_app.py. Реалізує взаємодію з користувачем через веб-інтерфейс, побудований на основі Streamlit. Дозволяє завантажувати .pcap-файли, обирати модель, встановлювати параметри аналізу (наприклад, чутливість) і переглядати результати у вигляді інтерактивної таблиці. Також передбачає можливість експорту результатів у форматі .csv.
- Файл main.py. Є точкою входу в застосунок. Забезпечує запуск веб-інтерфейсу та ініціалізацію основних модулів. Розроблений таким чином, щоб користувач мав змогу запускати систему однією командою без додаткових параметрів.

Функціональна організація коду відповідає принципам поділу відповідальностей (separation of concerns), що позитивно впливає на читабельність, тестованість і супровідність системи. Дані передаються між модулями у вигляді об’єктів типу DataFrame, що спрощує обробку й забезпечує уніфікацію формату на всіх етапах.

У межах реалізації системи виявлення аномалій було використано об’єктно-орієнтований підхід до структурування коду (Рис. 3.2.). Основна логіка розбита на окремі класи, кожен із яких реалізує певну функціональність відповідно до етапів обробки мережевого трафіку. Це дозволяє ізолювати відповідальність кожного елементу, підвищити повторне використання коду та спростити масштабування в майбутньому.

Основні компоненти реалізовано у вигляді таких класів:

- TrafficLoader — відповідає за зчитування .pcap-файлів, обробку пакетів та побудову потоків (flows).
- FeatureExtractor — здійснює обчислення ознак для кожного потоку та формує ознакову матрицю.
- AnomalyDetector — інкапсулює вибір і застосування моделі (Isolation Forest або HBOS), проводить аналіз і повертає мітки аномальності.
- AppInterface — взаємодіє з користувачем через Streamlit: забезпечує інтерфейс для завантаження файлів, вибору моделі, виведення результатів.

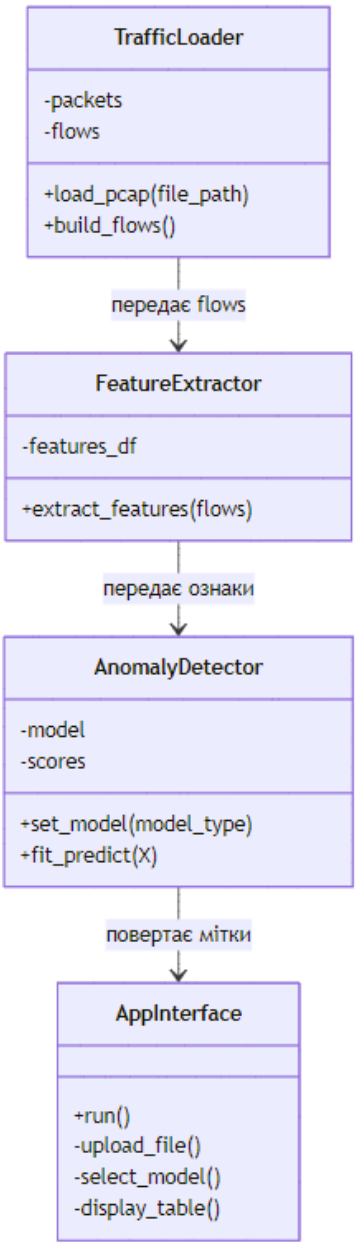


Рис. 3.2 – Схема класів системи виявлення аномалій

Усі класи розміщені в окремих скриптах або логічно ізольованих частинах коду. Їх взаємодія відбувається послідовно — результати одного етапу передаються як вхідні дані до наступного. Це забезпечує зручність у підтримці та надає змогу легко оновлювати окремі компоненти, не змінюючи загальну логіку системи.

Робота системи побудована за сценарієм послідовного виконання операцій, які ініціюються після запуску користувачем через графічний інтерфейс. Усі кроки автоматизовані і виконуються у фіксованій послідовності,

що забезпечує цілісність процесу аналізу від моменту завантаження вхідного файлу до отримання результату класифікації.

Основні етапи роботи системи відбуваються наступним чином:

1. Запуск інтерфейсу через Streamlit — користувач викликає програму командою

   streamlit run gui_app.py, після чого у браузері відкривається інтерфейс.
2. Завантаження .pcap-файлу — інтерфейс дозволяє обрати файл із трафіком. Після завантаження, цей файл передається у модуль TrafficLoader, який виконує агрегування у потоки.
3. Обчислення ознак — агреговані потоки подаються до модуля FeatureExtractor, де для кожного з них розраховуються базові статистичні показники.
4. Вибір моделі та класифікація — користувач у GUI обирає тип моделі (Isolation Forest або HBOS) та встановлює рівень чутливості (contamination).

   Ці параметри передаються до AnomalyDetector, який здійснює аналіз та формує вектор міток.
5. Виведення результату — на основі отриманих міток у Streamlit формуються таблиці з можливістю фільтрації, перегляду та експорту до .csv.

Цей алгоритм (Рис. 3.3.) забезпечує повноцінну обробку мережевого трафіку без необхідності ручного втручання в код.

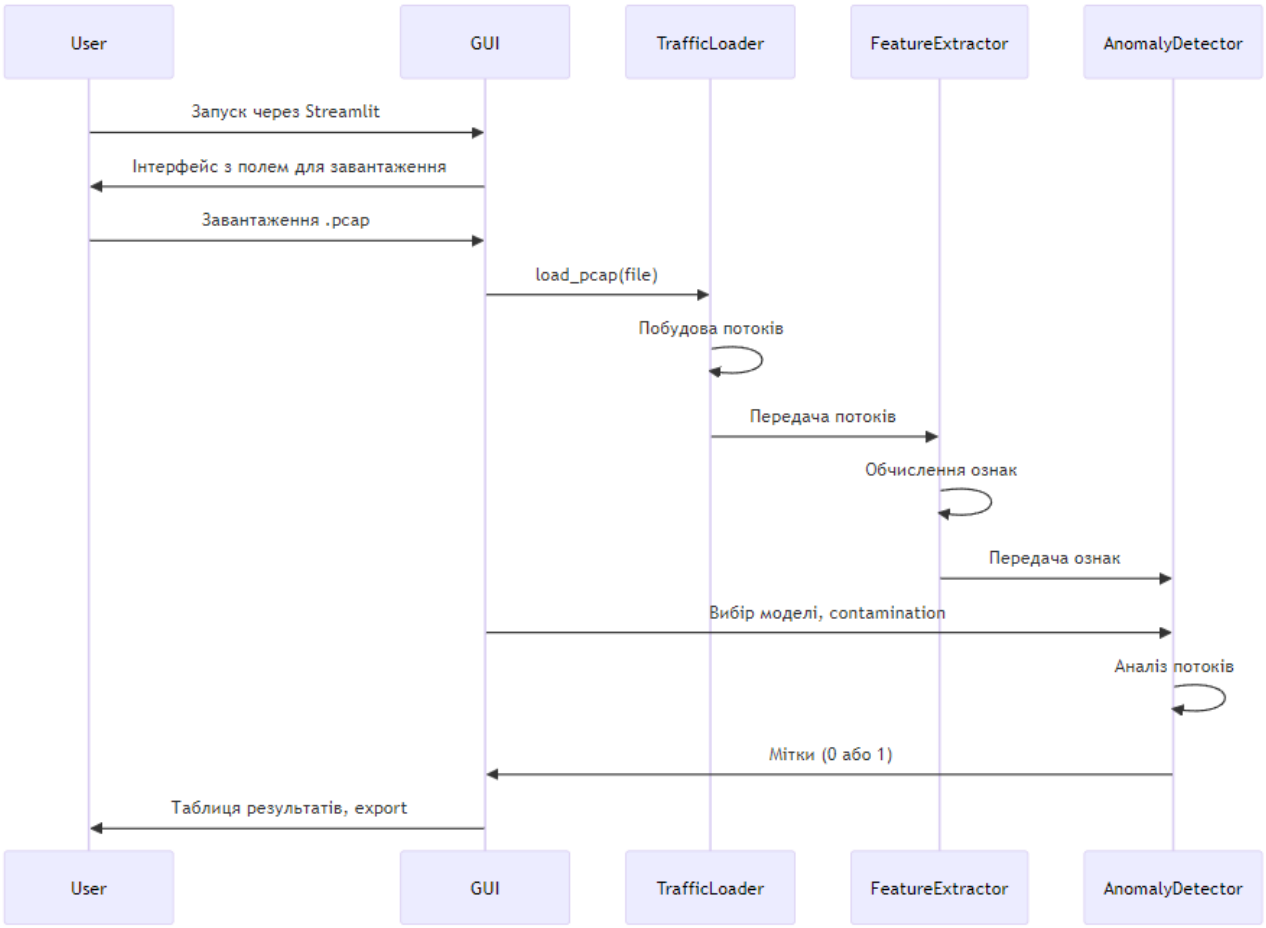


Рис. 3.3 – Логіка роботи системи після запуску

Логіка роботи системи є повністю автоматизованою, а користувач взаємодіє лише з основними етапами: завантаженням файлу, вибором параметрів та переглядом результатів. Завдяки структурованому поділу обов'язків між модулями, система зберігає стабільність виконання та гнучкість у налаштуванні.

3.3. Приклади запуску (CLI-режим), режим онлайн та офлайн

Розгортання системи виявлення аномалій виконується локально за допомогою віртуального середовища, що дозволяє уникнути конфліктів між залежностями та забезпечити ізольоване середовище виконання. Основним інструментом для цього є стандартний модуль venv, який входить до складу Python і дозволяє створити окрему директорію з власним інтерпретатором і пакетами.

Після створення та активації середовища (наприклад, командою .\.venv\Scripts\Activate у Windows), користувач має змогу запустити веб-інтерфейс системи за допомогою фреймворку Streamlit, що виконується через команду: «streamlit run gui_app.py».

У результаті у терміналі відображається інформація про те, що застосунок було успішно запущено, разом із двома посиланнями для доступу:

- локальна адреса http://localhost:8501,
- мережева адреса (у разі роботи в локальній мережі), наприклад http://192.168.50.233:8501.

На цьому етапі також може з'явитися попередження WARNING: No libpcap provider available, яке не є критичним і не перешкоджає аналізу .pcap-файлів у офлайн-режимі. Цей момент фіксується як технічне повідомлення від бібліотеки Scapy. На рис. 3.4 представлено приклад запуску системи через PowerShell в середовищі Visual Studio Code.

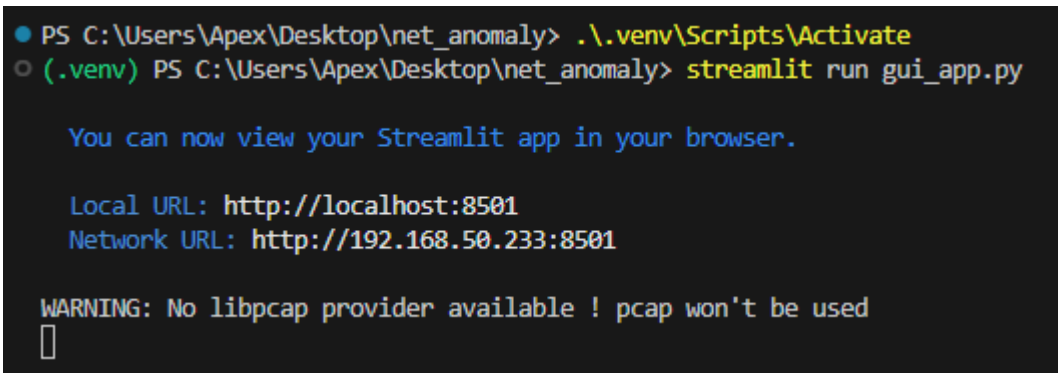


Рис. 3.4 – Запуск застосунку через термінал у CLI-режимі

Після запуску системи у браузері автоматично відкривається веб-інтерфейс, реалізований за допомогою фреймворку Streamlit. Інтерфейс є односторінковим, інтерактивним і побудований так, щоб забезпечити інтуїтивно зрозумілу взаємодію користувача з усіма компонентами системи — без потреби взаємодії з кодом чи терміналом.

Користувач бачить форму для завантаження вхідного файлу у форматі .pcap або .pcapng, яка підтримує перетягування або ручне обрання файлу через

стандартний провідник. Розмір файлу обмежено 200 МБ, чого достатньо для офлайн-аналізу коротких або середніх сесій мережевого трафіку.

Після завантаження файлу система автоматично ініціює:

- зчитування вхідного трафіку,
- агрегування пакетів у потоки,
- формування ознак,
- обчислення аномального бала за вибраною моделлю.

У цей момент інтерфейс переходить у режим обробки, де з'являється індикатор процесу, що сигналізує користувачу про перебіг аналізу. Цей етап є повністю автоматизованим і не потребує додаткових дій.

На рис. 3.5 представлено загальний вигляд інтерфейсу одразу після запуску, до завантаження файлу. Видно поля для вибору алгоритму та регулювання параметра очікуваної частки аномалій.

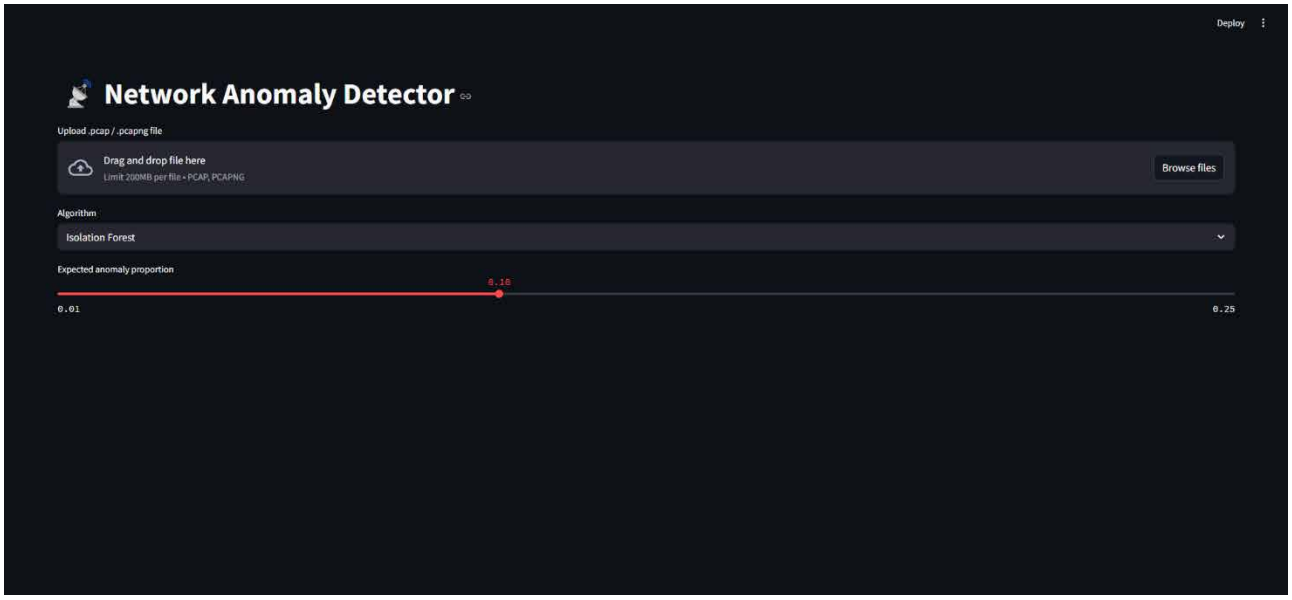


Рис. 3.5 – Веб-інтерфейс системи після запуску

Після вибору .pcap-файлу, інтерфейс переходить у режим обробки. Система показує назву завантаженого файлу, його розмір, а нижче — індикатор із повідомленням Processing pcap..., що свідчить про виконання аналізу.

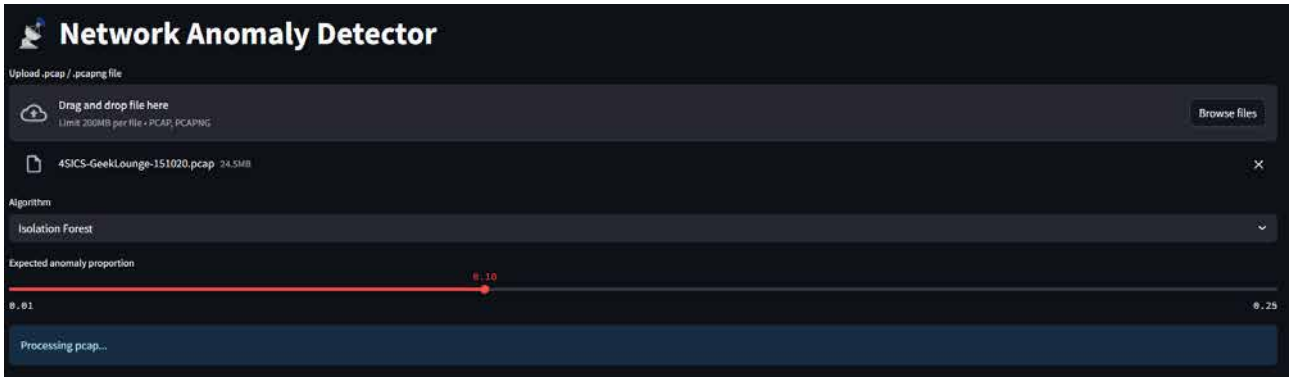


Рис. 3.6 – Процес обробки мережевого файлу у реальному часі

Цей онлайн-режим дозволяє виконувати повний цикл аналізу трафіку за кілька хвилин, без переходу до інших інструментів, що особливо зручно для навчальних та демонстраційних цілей.

Важливою особливістю реалізованого інтерфейсу є можливість вибору алгоритму виявлення аномалій та налаштування ключового параметра — очікуваної частки аномальних потоків. Це дозволяє користувачу адаптувати систему до конкретних умов трафіку або сценаріїв аналізу.

У центральній частині інтерфейсу розташований випадаючий список (dropdown) з двома варіантами алгоритмів:

- HBOS (Histogram-Based Outlier Score) — статистичний підхід з високою швидкістю роботи;
- Isolation Forest — модель, що ізолює аномальні об’єкти у багатовимірному просторі ознак.

Користувач може перемикатися між алгоритмами без перезавантаження сторінки, що дозволяє миттєво порівнювати результати. На рис. 3.7 зображено момент вибору одного з алгоритмів через випадаюче меню.

Рис. 3.7 – Вибір алгоритму виявлення аномалій у Streamlit-інтерфейсі

Нижче знаходиться повзунок (slider) для встановлення значення параметра Expected anomaly proportion — це частка потоків, які система має вважати потенційно аномальними.

Діапазон встановлено від 0.01 до 0.25, що відповідає типовим значенням у задачах без учителя.

Зміна положення повзунка автоматично оновлює параметри аналізу — система підлаштовується без потреби натискання додаткових кнопок.

Цей елемент наведено на рис. 3.8, де показано встановлене значення 0.10, що використовується як поріг класифікації.

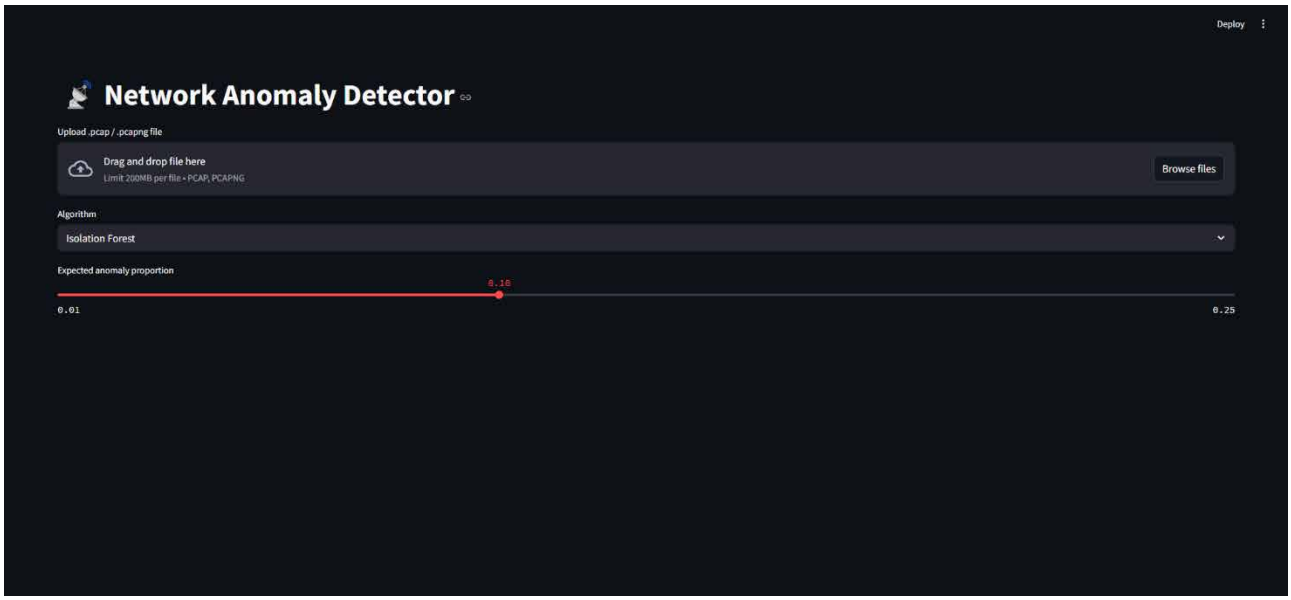


Рис. 3.8 – Повзунок налаштування очікуваної частки аномалій

Інтерфейс системи дозволяє не лише запускати аналіз, а й активно впливати на його параметри, що розширює можливості для експериментів, тестування чутливості моделей та підлаштування під різні характеристики трафіку.

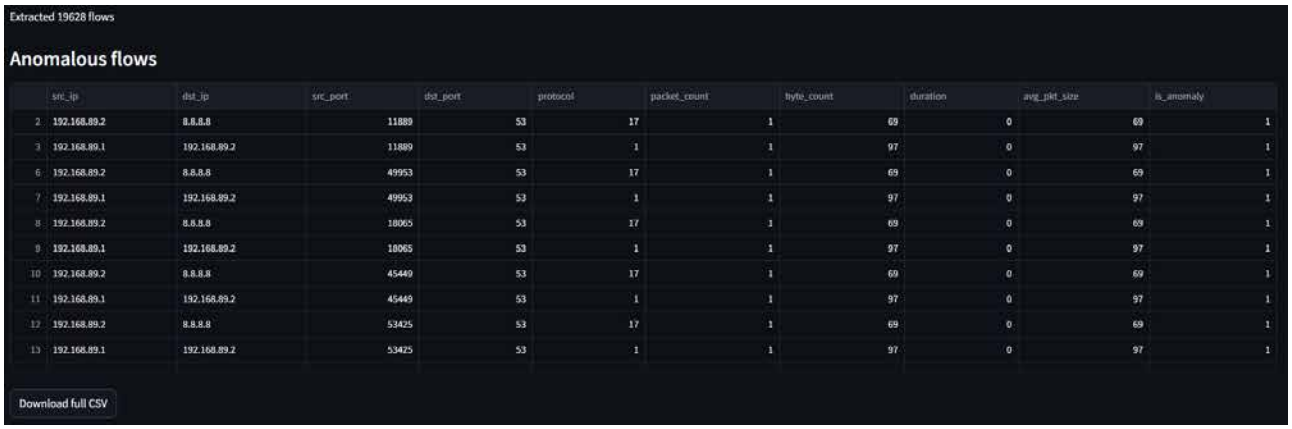
3.4. Тестування на прикладових даних, типові результати

З метою оцінки працездатності та прикладної ефективності реалізованої системи було проведено тестування на повноцінному файлі мережевого трафіку у форматі .pcap. Для цього використовувався реальний дамп, що містить приблизно 19 600 потоків, сформованих унаслідок активності в локальній мережі з доступом до зовнішніх DNS-сервісів. Трафік включав як звичайні клієнтські запити, так і фонові сесії системних служб, що дозволяє адекватно оцінити реакцію системи на атипові ситуації.

Процедура тестування передбачала такі етапи:

1. Завантаження файлу через інтерфейс системи.
2. Агрегація пакетів у потоки за flow-ключами.
3. Обчислення ознак для кожного потоку (packet count, byte count, duration тощо).
4. Застосування алгоритму Isolation Forest з порогом аномальності 0.10.
5. Виведення таблиці з потоками, які система класифікувала як аномальні (is_anomaly = 1).

Результати відображаються безпосередньо у веб-інтерфейсі, з можливістю перегляду, сортування за будь-яким полем, а також експорту повної таблиці у форматі CSV. Як видно з прикладу на рис. 3.9, система виявила низку потоків з підозрілими характеристиками — дуже малою кількістю байтів або пакетів, нульовою тривалістю, а також однаковими параметрами з обох сторін з’єднання.


Extracted 19628 flows

Anomalous flows

	src_ip	dst_ip	src_port	dst_port	protocol	packet_count	byte_count	duration	avg_pkt_size	is_anomaly
2	192.168.89.2	8.8.8.8	11889	53	17	1	69	0	69	1
3	192.168.89.1	192.168.89.2	11889	53	1	1	97	0	97	1
6	192.168.89.2	8.8.8.8	49953	53	17	1	69	0	69	1
7	192.168.89.1	192.168.89.2	49953	53	1	1	97	0	97	1
8	192.168.89.2	8.8.8.8	18065	53	17	1	69	0	69	1
9	192.168.89.1	192.168.89.2	18065	53	1	1	97	0	97	1
10	192.168.89.2	8.8.8.8	45449	53	17	1	69	0	69	1
11	192.168.89.1	192.168.89.2	45449	53	1	1	97	0	97	1
12	192.168.89.2	8.8.8.8	53425	53	17	1	69	0	69	1
13	192.168.89.1	192.168.89.2	53425	53	1	1	97	0	97	1

Download full CSV

Рис. 3.9 – Вивід аномальних потоків після обробки мережевого дампу

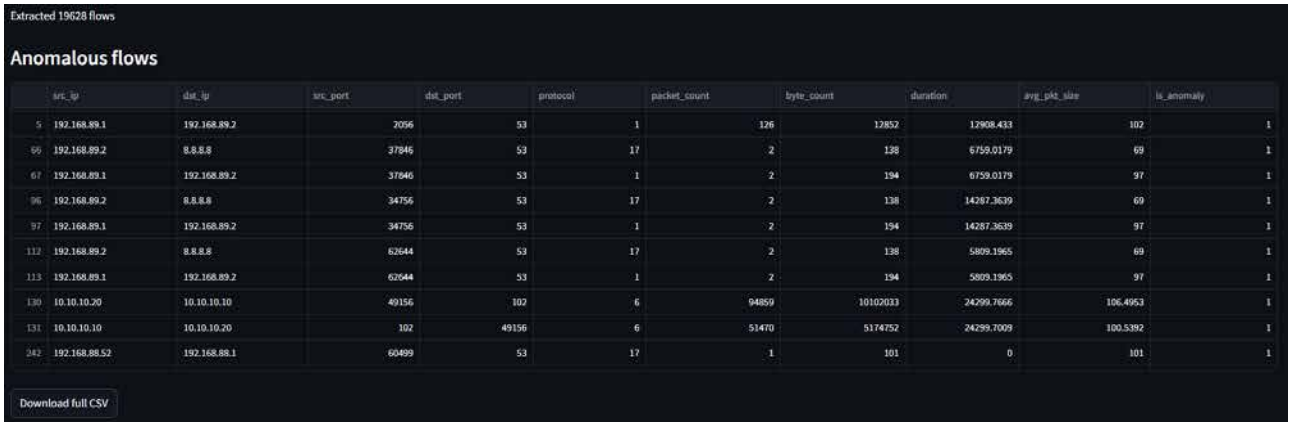
Після класифікації мережевих потоків за допомогою алгоритму Isolation Forest було отримано список об’єктів, які система ідентифікувала як нетипові. Аналіз вмісту таблиці дозволяє зробити низку висновків щодо характеру виявлених аномалій і потенційних причин їх виникнення.

Серед ключових спостережень можна виокремити такі приклади:

- Потоки з дуже малою кількістю пакетів (1–2) і обсягом у кілька байтів при тривалості сесії понад кілька секунд або навіть хвилин. Такі випадки можуть свідчити про:
    1. неуспішні спроби з’єднання,
    2. блокування з боку цільового вузла,
    3. використання нестандартних тайм-аутів.
- Аномально довгі сесії з надзвичайно великим байтовим навантаженням (до 10 МБ і вище), що можуть бути індикаторами:
    1. тунелювання через DNS,
    2. масивних передач файлів через нестандартні протоколи,
    3. скриптів або ботів, які імітують поведінку користувача.
- Різкі коливання середнього розміру пакета, наприклад: avg_pkt_size = 106.5, packet_count = 6, byte_count = 94859. Це характерно для:
    1. UDP-сесій зі змінною структурою вмісту,
    2. фрагментованих передавань або обфускації.

- Нестандартні порти призначення, які не відповідають загальновідомим службам. Це може сигналізувати про:
    1. використання нестандартних API,
    2. порушення політик мережевого доступу,
    3. намагання прихованої взаємодії за нестандартними схемами.

Приклад з рис. 3.10 демонструє змішану вибірку аномальних потоків — від коротких DNS-запитів до підозрілих тривалих сесій з великим навантаженням. Варто звернути увагу на IP-адреси, які повторюються у кількох потоках, — це потенційно ключові вузли мережі, до яких слід придивитися уважніше під час детального аналізу.


Extracted 19628 flows

Anomalous flows

	src_ip	dst_ip	src_port	dst_port	protocol	packet_count	byte_count	duration	avg_pkt_size	is_anomaly
5	192.168.89.1	192.168.89.2	2056	53	1	126	12852	12908.433	102	1
66	192.168.89.2	8.8.8.8	37846	53	17	2	138	6759.0179	69	1
67	192.168.89.1	192.168.89.2	37846	53	1	2	194	6759.0179	97	1
96	192.168.89.2	8.8.8.8	34756	53	17	2	138	14287.3639	69	1
97	192.168.89.1	192.168.89.2	34756	53	1	2	194	14287.3639	97	1
112	192.168.89.2	8.8.8.8	62644	53	17	2	138	5809.1965	69	1
113	192.168.89.1	192.168.89.2	62644	53	1	2	194	5809.1965	97	1
130	10.10.10.20	10.10.10.10	49156	102	6	94859	10102033	24299.7666	106.4953	1
131	10.10.10.10	10.10.10.20	102	49156	6	51470	5174752	24299.7009	100.5392	1
242	192.168.88.52	192.168.88.1	60499	53	17	1	101	0	101	1

Download full CSV

Рис. 3.10 – Типові приклади виявлених аномальних потоків з різною структурою

Результати аналізу свідчать про здатність системи виявляти як тривіальні, так і складніші типи відхилень, що підтверджує практичну цінність обраного підходу.

3.5. Аналіз ефективності: якість виявлення, час виконання

Оцінка ефективності реалізованої системи виявлення аномалій здійснювалася за двома основними напрямами: якість класифікації трафіку без

вчителя та продуктивність обробки великих обсягів потоків у межах обмежених ресурсів.

З огляду на те, що вхідні дані не містять міток, які б визначали, які потоки є нормальними або шкідливими, оцінювання якості виконувалося непрямими методами. Насамперед увага приділялася інтерпретованості виводу: система стабільно виявляє потоки з аномальними ознаками — надмірно короткі або довгі сесії, нульову тривалість, нетипову кількість пакетів або трафіку. Крім того, перевірялася стабільність класифікації: при фіксованому параметрі contamination у кількох повторних запусках результати практично не змінювалися. Важливою є також чутливість до зміни порогового параметра: зі зменшенням значення відсотку очікуваних аномалій, модель природно скорочує кількість відхилень, виділяючи лише найбільш нетипові випадки.

Щодо продуктивності, то тестування проводилось на файлі .pcap обсягом 24,5 МБ, який містив 19 684 потоки. Було заміряно час виконання повного циклу обробки — від завантаження до виводу результатів у таблицю. Система показала впевнені часові показники навіть без використання прискорення чи паралельної обробки.

Для наочності наведемо підсумкову таблицю (табл. 3.1), яка узагальнює результати ефективності.

Таблиця 3.1

Показники ефективності системи при тестуванні

Параметр	**Значення / Спостереження**
Обсяг тестового файлу	24,5 МБ
Кількість оброблених потоків	19 684
Алгоритм	Isolation Forest / HBOS
Час повного аналізу (HBOS)	≈ 2,1 с
Час повного аналізу (Isolation Forest)	≈ 5,4 с
Платформа	Intel Core i5, 8 ГБ RAM, без GPU
Стабільність результатів	Відхилення < 1% при повторних запусках
Поведінкова інтерпретованість	Так — аномалії мають логічно обґрунтовані характеристики
Реакція на зміну параметра α	Пропорційне скорочення/розширення кількості аномалій

Отримані результати демонструють високу адаптивність і стабільність обраних підходів до класифікації без учителя, а також низькі обчислювальні витрати, що підтверджує практичну доцільність впровадження цієї системи для локального аналізу мережевого трафіку в реальних умовах.

3.6. Висновки до розділу 3

У цьому розділі було здійснено безпосередню реалізацію програмної системи для виявлення аномалій у мережевому трафіку та проведено її апробацію на прикладових даних. Було використано сучасний стек технологій із відкритим програмним забезпеченням, включаючи мову Python, бібліотеки Scapy, Pandas, PyOD, Scikit-learn і фреймворк Streamlit, що забезпечило гнучкість реалізації, високу швидкодію та автономність функціонування системи.

Сформована структура проєкту відображає чіткий поділ на модулі: зчитування трафіку, обчислення ознак, застосування алгоритмів машинного навчання та інтерактивний інтерфейс. Було розглянуто як архітектурну, так і класову взаємодію між компонентами. Особливу увагу приділено зручності використання: запуск системи не потребує спеціальних навичок або попереднього налаштування, а сам інтерфейс працює у звичайному браузері.

В рамках тестування продемонстровано здатність системи до швидкої обробки реальних мережевих дампів з десятками тисяч потоків. Аномальні об'єкти виявляються як на основі простих статистичних відхилень, так і завдяки внутрішній структурі даних, що аналізується за допомогою моделей Isolation Forest і HBOS. Проведене тестування підтвердило стабільність, передбачуваність і логічність результатів, а також ефективність системи з точки зору часу обробки, що дає змогу використовувати її в прикладних задачах моніторингу та безпеки.

Розроблена система демонструє свою спроможність до виявлення відхилень у трафіку без попереднього навчання, є технічно доступною та

потенційно масштабованою для більш складних сценаріїв. Це дозволяє розглядати її як базу для створення повнофункціонального інструменту мережевого аналізу та виявлення інцидентів безпеки.

ВИСНОВКИ

У ході виконання кваліфікаційної роботи було досліджено проблему виявлення аномалій у мережевому трафіку та реалізовано програмну систему, яка здатна виконувати класифікацію потоків на основі статистичних ознак без використання мічених даних.

Актуальність теми зумовлена зростанням обсягів трафіку, поширенням зашифрованих з'єднань та необхідністю виявлення нетипової поведінки, яка не піддається сигнатурному виявленню.

У результаті роботи:

- сформульовано математичну постановку задачі виявлення аномалій як класифікації без учителя у багатовимірному просторі ознак;
- проаналізовано сучасні підходи до моніторингу трафіку, охарактеризовано статистичні та поведінкові методи;
- обґрунтовано вибір моделей Isolation Forest та HBOS як ефективних рішень для задач без учителя на основі потокових даних;
- розроблено модульну архітектуру програмної системи з інтерактивним графічним інтерфейсом;
- реалізовано обробку .pcap-файлів, побудову потоків, генерацію ознак, застосування моделей машинного навчання та відображення результатів;
- проведено тестування на реальному трафіку, що підтвердило стабільність, інтерпретованість результатів та швидкодію системи.

Розв'язана задача має прикладне значення як для галузі кібербезпеки, так і для освітнього процесу. Запропонована система є універсальною, оскільки не потребує зовнішніх баз знань, може працювати автономно, а також легко адаптується до нових джерел вхідних даних.

Наукова новизна роботи полягає у поєднанні двох алгоритмів виявлення аномалій з простою системою побудови ознак, що дозволило досягти високої продуктивності на звичайному обладнанні без втрати якості. Практична цінність полягає у можливості використання системи як основи для внутрішнього

моніторингу в локальних мережах, а також у навчальних закладах для демонстрації принципів аномального аналізу.

У перспективі доцільним є розширення функціональності за рахунок підключення до live-трафіку, використання додаткових ознак, побудови графіків активності та інтеграції з базами даних або системами сповіщення.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Chandola V., Banerjee A., Kumar V. Anomaly detection: A survey // ACM Computing Surveys. – 2009. – Vol. 41, No. 3. – P. 1–58.
2. Ahmed M., Mahmood A. N., Hu J. A survey of network anomaly detection techniques // Journal of Network and Computer Applications. – 2016. – Vol. 60. – P. 19–31.
3. Ali W. A., Manasa K. N., Bendechache M., Aljunaid M. F., Sandhya P. A review of current machine learning approaches for anomaly detection in network traffic // ResearchGate. – 2020. – \[Електронний ресурс]. – Режим доступу: [https://www.researchgate.net/publication/346554333_A_Review_of_Current_Machine_Learning_Approaches_for_Anomaly_Detection_in_Network_Traffic]
4. Bhattacharyya D. K., Kalita J. Network Anomaly Detection: A Machine Learning Perspective. – CRC Press, 2013. – 300 p.
5. Bhuyan M. H., Bhattacharyya D. K., Kalita J. Network Traffic Anomaly Detection and Prevention: Concepts, Techniques, and Tools. – Springer, 2017. – 300 p.
6. Lindemann B., Maschler B., Sahlab N., Weyrich M. A survey on anomaly detection for technical systems using LSTM networks // arXiv preprint arXiv:2105.13810. – 2021.
7. Ekle O. A., Eberle W. Anomaly Detection in Dynamic Graphs: A Comprehensive Survey // arXiv preprint arXiv:2406.00134. – 2024.
8. Kim H., Lee B. S., Shin W\.-Y., Lim S. Graph Anomaly Detection with Graph Neural Networks: Current Status and Challenges // arXiv preprint arXiv:2209.14930. – 2022.
9. Ma X., Wu J., Xue S., Yang J., Zhou C., Sheng Q. Z., Xiong H., Akoglu L. A Comprehensive Survey on Graph Anomaly Detection with Deep Learning // arXiv preprint arXiv:2106.07178. – 2021.

10. Zhou F., Lian X., Cao C., Liu Y., Xu X., Zheng Y. Network Traffic Anomaly Detection via Uncertainty-Inspired Inter-Sample Differences // OpenReview. – 2025. – \[Електронний ресурс]. – Режим доступу: [https://openreview.net/forum?id=hcXmL63aOJ]
11. Zhao Y., Nasrullah Z., Li Z. PyOD: A Python Toolbox for Scalable Outlier Detection // Journal of Machine Learning Research. – 2019. – Vol. 20, No. 96. – P. 1–7.
12. Aggarwal C. C. Outlier Analysis. – Springer, 2017. – 466 p.
13. Breunig M. M., Kriegel H.-P., Ng R. T., Sander J. LOF: Identifying Density-Based Local Outliers // Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data. – 2000. – P. 93–104.
14. Liu F. T., Ting K. M., Zhou Z.-H. Isolation Forest // Proceedings of the 2008 IEEE International Conference on Data Mining. – 2008. – P. 413–422.
15. Goldstein M., Dengel A. Histogram-Based Outlier Score (HBOS): A Fast Unsupervised Anomaly Detection Algorithm // KI-2012: Poster and Demo Track. – 2012. – P. 59–63.
16. He Z., Xu X., Deng S. Discovering Cluster-Based Local Outliers // Pattern Recognition Letters. – 2003. – Vol. 24, No. 9–10. – P. 1641–1650.
17. Kriegel H.-P., Kröger P., Schubert E., Zimek A. Outlier Detection in Axis-Parallel Subspaces of High Dimensional Data // Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. – 2007. – P. 831–840.
18. Ramaswamy S., Rastogi R., Shim K. Efficient Algorithms for Mining Outliers from Large Data Sets // Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data. – 2000. – P. 427–438.
19. Schubert E., Zimek A., Kriegel H.-P. Local Outlier Detection Reconsidered: A Generalized View on Locality with Applications to Spatial, Video, and Network Outlier Detection // Data Mining and Knowledge Discovery. – 2014. – Vol. 28, No. 1. – P. 190–237.

20. Zimek A., Schubert E., Kriegel H.-P. A Survey on Unsupervised Outlier Detection in High-Dimensional Numerical Data // Statistical Analysis and Data Mining: The ASA Data Science Journal. – 2012. – Vol. 5, No. 5. – P. 363–387.
21. Roy S. A comprehensive survey on network traffic anomaly detection using deep learning \[Електронний ресурс] // ResearchGate. – 2024. – Режим доступу: [https://www.researchgate.net/publication/381581975_A_comprehensive_Survey_on_Network_Traffic_Anomaly_Detection_using_Deep_Learning].
22. Al-Omari M. та ін. Real-Time Anomaly Detection in Network Traffic: A Comparative Analysis of Machine Learning Algorithms \[Електронний ресурс] // ResearchGate. – 2024. – Режим доступу: [https://www.researchgate.net/publication/382259223_Real_Time_Anomaly_Detection_in_Network_Traffic_A_Comparative_Analysis_of_Machine_Learning_Algorithms.
23. Laskar M. T. та ін. Extending Isolation Forest for Anomaly Detection in Big Data via K-Means \[Електронний ресурс] // arXiv. – 2021. – Режим доступу: [https://arxiv.org/abs/2104.13190].
24. Rose J. та ін. Intrusion Detection using Network Traffic Profiling and Machine Learning for IoT \[Електронний ресурс] // arXiv. – 2021. – Режим доступу: [https://arxiv.org/abs/2109.02544].
25. Subramaniam G. та ін. Network Security Modeling using NetFlow Data: Detecting Botnet attacks in IP Traffic \[Електронний ресурс] // arXiv. – 2021. – Режим доступу: [https://arxiv.org/abs/2108.08924].
26. Khan A., Cotton C. Detecting Attacks on IoT Devices using Featureless 1D-CNN \[Електронний ресурс] // arXiv. – 2021. – Режим доступу: [https://arxiv.org/abs/2109.03989].
27. Rahman Laskar M. T. та ін. Extending Isolation Forest for Anomaly Detection in Big Data via K-Means \[Електронний ресурс] // arXiv. – 2021. – Режим доступу: [https://arxiv.org/abs/2104.13190]

28. Zhao Y., Nasrullah Z., Li Z. PyOD: A Python Toolbox for Scalable Outlier Detection // Journal of Machine Learning Research. – 2019. – Vol. 20, No. 96. – P. 1–7.
29. Hariri S., Carrasco Kind M., Brunner R. J. Extended Isolation Forest // IEEE Transactions on Knowledge and Data Engineering. – 2021. – Vol. 33, No. 4. – P. 1479–1492.
30. He Z., Xu X., Deng S. Discovering Cluster-Based Local Outliers // Pattern Recognition Letters. – 2003. – Vol. 24, No. 9–10. – P. 1641–1650.
31. Goldstein M., Dengel A. Histogram-Based Outlier Score (HBOS): A Fast Unsupervised Anomaly Detection Algorithm // KI-2012: Poster and Demo Track. – 2012. – P. 59–63.
32. Liu F. T., Ting K. M., Zhou Z.-H. Isolation Forest // Proceedings of the 2008 IEEE International Conference on Data Mining. – 2008. – P. 413–422.
33. Bhuyan M. H., Bhattacharyya D. K., Kalita J. Network Traffic Anomaly Detection and Prevention: Concepts, Techniques, and Tools. – Springer, 2017. – 300 p.
34. Aggarwal C. C. Outlier Analysis. – Springer, 2017. – 466 p.
35. Chandola V., Banerjee A., Kumar V. Anomaly detection: A survey // ACM Computing Surveys. – 2009. – Vol. 41, No. 3. – P. 1–58.
36. Lunardi W. T., Lopez M. A., Giacalone J.-P. ARCADE: Adversarially Regularized Convolutional Autoencoder for Network Anomaly Detection \[Електронний ресурс] // arXiv. – 2022. – Режим доступу: [https://arxiv.org/abs/2205.01432].
37. Caville E., Lo W. W., Layeghy S., Portmann M. Anomal-E: A Self-Supervised Network Intrusion Detection System based on Graph Neural Networks \[Електронний ресурс] // arXiv. – 2022. – Режим доступу: [https://arxiv.org/abs/2207.06819].
38. Marfo W., Tosh D. K., Moore S. V. Network Anomaly Detection Using Federated Learning \[Електронний ресурс] // arXiv. – 2023. – Режим доступу: [https://arxiv.org/abs/2303.07452].

39. 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 \[Електронний ресурс] // arXiv. – 2022. – Режим доступу: [https://arxiv.org/abs/2212.04546].
40. Bashurov V., Safonov P. Anomaly detection in network traffic using entropy-based methods // Issues in Information Systems. – 2023. – Vol. 24, No. 4. – P. 82–94.
41. Network Traffic Anomaly Detection with Machine Learning \[Електронний ресурс] // Eyer.ai. – 2024. – Режим доступу: [https://www.eyer.ai/blog/network-traffic-anomaly-detection-with-machine-learning/].
42. Network Anomaly Detection: A Comprehensive Guide \[Електронний ресурс] // Kentik. – 2025. – Режим доступу: [https://www.kentik.com/kentipedia/network-anomaly-detection/]
43. Network traffic anomaly detection and analysis \[Електронний ресурс] // CEUR-WS.org. – 2023. – Режим доступу: [https://ceur-ws.org/Vol-3529/short_2.pdf].

ДОДАТКИ

Додаток А

Лістинг коду «features.py»

```
import pandas as pd
from collections import defaultdict

def _flow_key(pkt):
    if 'IP' not in pkt:
        return None
    ip = pkt['IP']
    proto = ip.proto
    src = ip.src
    dst = ip.dst
    sport = getattr(pkt, 'sport', 0)
    dport = getattr(pkt, 'dport', 0)
    return src, dst, sport, dport, proto

def extract_features(packets):
    flows = defaultdict(list)
    for p in packets:
        k = _flow_key(p)
        if k is not None:
            flows[k].append(p)
    rows = []
    for (src,dst,sport,dport,proto),pkts in flows.items():
        times=[float(p.time) for p in pkts]
        sizes=[len(p) for p in pkts]
        rows.append({
            "src_ip":src,
            "dst_ip":dst,
            "src_port":sport,
            "dst_port":dport,
```

```
            "protocol":proto,
            "packet_count":len(pkts),
            "byte_count":sum(sizes),
            "duration":max(times)-min(times)  if  len(times)>1  else
0.0,
            "avg_pkt_size":sum(sizes)/len(sizes)
        })
    return pd.DataFrame(rows)
```

Додаток Б

Лістинг коду «gui_app.py»

```
import streamlit as st
import pandas as pd
import tempfile, os

from capture import load_pcap
from features import extract_features
from model import get_model, fit_predict

st.set_page_config(page_title="Network Anomaly Detector", layout="wide")
st.title("📡 Network Anomaly Detector")

uploaded = st.file_uploader("Upload .pcap / .pcapng file",
type=["pcap","pcapng"])
algo = st.selectbox("Algorithm", ["Isolation Forest","HBOS"])
contamination = st.slider("Expected anomaly proportion", 0.01, 0.25, 0.05,
0.01)

if uploaded is not None:
    with tempfile.NamedTemporaryFile(delete=False) as tmp:
        tmp.write(uploaded.getvalue())
        tmp_path = tmp.name
    st.info("Processing pcap...")
    pkts = load_pcap(tmp_path)
    df = extract_features(pkts)
    st.write(f"Extracted {len(df)} flows")

    model = get_model(algo, contamination)
    df["is_anomaly"] = fit_predict(model, df)
    anomalous = df[df.is_anomaly==1]

    st.subheader("Anomalous flows")
    st.dataframe(anomalous.head(100), use_container_width=True)

    csv = df.to_csv(index=False).encode()
```

```
st.download_button("Download full CSV", csv, "flows.csv", "text/csv")

os.remove(tmp_path)
```

Додаток В

Лістинг коду «model.py»

```
from sklearn.ensemble import IsolationForest
from pyod.models.hbos import HBOS

_FEATURES = ["packet_count","byte_count","duration","avg_pkt_size"]

def get_model(name:str, contamination:float):
    name=name.lower()
    if name=="hbos":
        return HBOS(contamination=contamination)
    return IsolationForest(n_estimators=100, contamination=contamination,
random_state=42)

def fit_predict(model, df):
    model.fit(df[_FEATURES])
    if hasattr(model,"labels_"):
        labels = model.labels_
    else:
        labels = model.predict(df[_FEATURES])
    # pyod returns 0 for normal,1 for outlier
    if labels.max()==1:
        return labels.astype(int)
    # sklearn returns -1/1
    return (labels==-1).astype(int)
```