

Міністерство освіти і науки України
Державний університет «Житомирська політехніка»
Факультет інформаційно-комп'ютерних технологій
Кафедра комп'ютерної інженерії та кібербезпеки

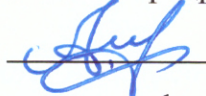
ПОЯСНЮВАЛЬНА ЗАПИСКА

до випускної кваліфікаційної роботи магістра
на тему: «Методи та засоби інтелектуального аналізу
мережного трафіку»

Виконала студентка 2-го курсу групи КБм-24-1
спеціальності 125 «Кібербезпека та захист інформації»

 Маргарита КИРИЄНКО

Керівник доцент кафедри КІ та КБ, к.т.н.

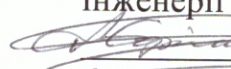
 Олексій ШЕЛУХА

Рецензент доцент кафедри КІ та КБ, к.т.н., доцент

 Юрій БРОДСЬКИЙ

Житомир – 2025

Державний університет «Житомирська політехніка»
Факультет інформаційно-комп'ютерних технологій
Кафедра комп'ютерної інженерії та кібербезпеки
Спеціальність 125 «Кібербезпека та захист інформації»
Освітня програма «Кібербезпека»

ЗАТВЕРДЖУЮ
Завідувач
кафедри комп'ютерної
інженерії та кібербезпеки
 Андрій Єфіменко
6 жовтня 2025 р.

ЗАВДАННЯ

на випускню кваліфікаційну роботу магістра

Студентка: Кириєнко Маргарита Юріївна

Тема роботи: «Методи та засоби інтелектуального аналізу мережного трафіку»
затверджена Наказом університету від 6 жовтня 2025 р. № 566/с

Термін здачі студентом закінченої роботи 8 грудня 2025 р.

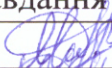
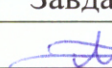
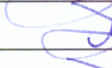
Вихідні дані роботи (зазначається предмет і об'єкт дослідження):

- процес аналізу мережевого трафіку в комп'ютерних мережах;
- методи та моделі інтелектуального виявлення аномалій і кібератак, що поєднують статистичний аналіз, rule-based логіку та елементи машинного навчання.

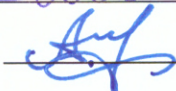
Обладнання та програмне забезпечення:

1. VirtualBox
2. Visual Studio Code
3. Python 3.12, бібліотеки: pyzabbix, pandas, numpy, scikit-learn
4. Telegram Bot API
5. Zabbix
6. OS Ubuntu / Windows 10

Консультанти з випускної кваліфікаційної роботи із зазначенням розділів, що їх стосуються

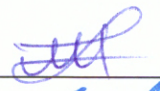
Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв
1	Маєвський О.В.	 06.10.25	 06.10.25
2	Маєвський О.В.	 06.10.25	 06.10.25
3	Маєвський О.В.	 06.10.25	 06.10.25

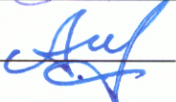
Студент  Маргарита КИРИЄНКО

Керівник  Олексій ШЕЛУХА

Календарний план

№ з/п	Назви етапів випускної кваліфікаційної роботи	Термін виконання етапів роботи	Примітка
1	Аналіз завдання та формування структури роботи	03.10.2025	Виконано
2	Аналіз літературних джерел	21.10.2025	Виконано
3	Аналіз методів інтелектуального аналізу та вибір інструментів	14.11.2025	Виконано
4	Проектування архітектури моделі	29.11.2025	Виконано
5	Визначення критеріїв аномалій	04.12.2025	Виконано
6	Розгортання інфраструктури моніторингу	05.12.2025	Виконано
7	Реалізація Python-модуля та rule-based аналізу	07.12.2025	Виконано
8	Інтеграція нейронної моделі (перцептрона)	09.12.2025	Виконано
9	Налаштування Telegram-сповіщень	10.12.2025	Виконано
10	Тестування роботи моделі та аналіз результатів	11.12.2025	Виконано
11	Оформлення пояснювальної записки	11.12.2025	Виконано
12	Оформлення презентації	14.12.2025	Виконано
13	Захист	18.12.2025	Виконано

Студент  Маргарита КИРИШЕНКО

Керівник  Олексій ШЕЛУХА

АНОТАЦІЯ

Кваліфікаційна робота на тему «Методи та засоби інтелектуального аналізу мережного трафіку» складається з переліку умовних скорочень, вступу, трьох розділів, висновків, списку використаної літератури та додатків.

Текстова частина викладена на 81 сторінці. Пояснювальна записка містить 40 рисунків, 2 таблиці та 13 сторінок додатків. Список використаних джерел містить 48 найменувань на 4 сторінках. Загальний обсяг роботи становить 95 сторінок.

Метою даної роботи є розробка моделі інтелектуального аналізу мережного трафіку для виявлення аномалій та ознак кібератак у режимі реального часу.

У першому розділі наведено огляд систем моніторингу мережі, методів інтелектуального аналізу даних та основних типів кібератак, що проявляються у мережевому трафіку.

У другому розділі описано проектування моделі: структуру системи, вибір інформативних ознак, поєднання rule-based підходу з одношаровим перцептроном та формування загальної логіки оцінювання ризику.

У третьому розділі реалізовано практичну частину моделі з використанням Zabbix і Python, виконано збір даних та навчання перцептрона, протестовано систему на змодельованих мережевих атаках і оцінено її ефективність у реальних умовах.

У додатках наведено лістинг програмного коду, використаного для реалізації моделі інтелектуального аналізу та автоматичного сповіщення.

Ключові слова: МЕРЕЖЕВИЙ ТРАФІК, ІНТЕЛЕКТУАЛЬНИЙ АНАЛІЗ, МОНІТОРИНГ, RULE-BASED МОДЕЛЬ, ПЕРЦЕПТРОН, АНОМАЛІЇ, ZABBIX, PYTHON, TELEGRAM-БОТ.

ANNOTATION

The qualification work titled “Methods and tools of intelligent network traffic analysis” consists of a list of abbreviations, an introduction, three chapters, conclusions, a list of references, and appendices.

The textual part includes 81 pages. The explanatory note contains 40 illustrations, 2 tables, and 13 pages of appendices. The list of references consists of 48 titles on 4 pages. The total volume of the work is 95 pages.

The purpose of this work is to develop a model of intelligent network traffic analysis for detecting anomalies and signs of cyberattacks in real time.

The first chapter provides an overview of network monitoring systems, methods of intelligent data analysis, and the main types of cyberattacks that manifest themselves in network traffic.

The second chapter presents the design of the model, including the system structure, the selection of informative features, the combination of the rule-based approach with a single-layer perceptron, and the formation of the overall risk assessment logic.

The third chapter describes the practical implementation of the model using Zabbix and Python, the data collection and perceptron training process, as well as the testing of the system under simulated network attacks and the evaluation of its effectiveness in real-world conditions.

The appendices provide the source code listings used to implement the intelligent analysis model and the automated notification system.

Keywords: NETWORK TRAFFIC, INTELLIGENT ANALYSIS, MONITORING, RULE-BASED MODEL, PERCEPTRON, ANOMALIES, ZABBIX, PYTHON, TELEGRAM BOT.

ЗМІСТ

ВСТУП	9
РОЗДІЛ 1. ОГЛЯД СИСТЕМ АНАЛІЗУ МЕРЕЖЕВОГО ТРАФІКУ ТА МЕТОДІВ ВИЯВЛЕННЯ КІБЕРАТАК	10
1.1. Аналіз існуючих рішень для моніторингу мережевого трафіку.....	10
1.2. Методи інтелектуального аналізу даних у мережевій безпеці	16
1.3. Сучасні кібератаки та характерні ознаки їх виявлення у мережевому трафіку	22
1.4. Постановка завдання дослідження	28
РОЗДІЛ 2. ПРОЄКТУВАННЯ МОДЕЛІ ІНТЕЛЕКТУАЛЬНОГО АНАЛІЗУ МЕРЕЖЕВОГО ТРАФІКУ	31
2.1. Вибір технологій та методів для побудови моделі інтелектуального аналізу	31
2.2. Проектування архітектури моделі інтелектуального аналізу мережевого трафіку.....	34
2.3. Визначення параметрів і критеріїв виявлення аномалій	40
РОЗДІЛ 3. РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ МОДЕЛІ ІНТЕЛЕКТУАЛЬНОГО АНАЛІЗУ МЕРЕЖЕВОГО ТРАФІКУ	46
3.1. Налаштування та розгортання інфраструктури збору мережевих даних Zabbix.....	46
3.2. Реалізація аналітичної підсистеми моделі інтелектуального аналізу	59
3.3. Тестування моделі інтелектуального аналізу мережевого трафіку	67
ВИСНОВКИ.....	77
СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ	78
ДОДАТКИ.....	82

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

AI – Artificial Intelligence

API – Application Programming Interface

CPU – Central Processing Unit

DoS – Denial of Service

DDoS – Distributed Denial of Service

DPI – Deep Packet Inspection

ICMP – Internet Control Message Protocol

IDS/IPS – Intrusion Detection System / Intrusion Prevention System

IBM – International Business Machines Corporation

IT – Information Technology

ML – Machine Learning

NMS – Network Monitoring System

OSI – Open Systems Interconnection

PRTG – Paessler Router Traffic Grapher

QoS – Quality of Service

SIEM – Security Information and Event Management

SNMP – Simple Network Management Protocol

SQL – Structured Query Language

SYN – Synchronize

TCP – Transmission Control Protocol

ПЗ – програмне забезпечення

ПК – персональний комп'ютер

ЦП – центральний процесор

ВСТУП

Актуальність теми. У сучасних умовах стрімкого розвитку цифрових технологій та зростання обсягу мережевого трафіку питання кібербезпеки набуває особливої важливості, адже кількість і складність кібератак постійно зростають, створюючи ризики для інформаційних систем та критичної інфраструктури. Традиційні засоби моніторингу часто не забезпечують своєчасного виявлення нових або прихованих атак, що зумовлює потребу у більш інтелектуальних методах аналізу трафіку. Водночас складні моделі машинного та глибинного навчання потребують значних ресурсів і витрат, тому у практичних середовищах актуальними є простіші підходи, які поєднують статистичний аналіз, rule-based логіку та елементи машинного навчання й здатні забезпечити ефективне виявлення аномалій без суттєвих інфраструктурних вимог.

Метою кваліфікаційної роботи є розробка моделі інтелектуального аналізу мережевого трафіку для виявлення аномалій та ознак кібератак у режимі реального часу.

Визначена мета обумовлює наступні завдання:

- проаналізувати сучасні підходи до моніторингу мережевого трафіку та методи інтелектуального виявлення кібератак;
- спроектувати архітектуру моделі інтелектуального аналізу, визначивши її основні компоненти та параметри;
- реалізувати прототип системи, що поєднує статистичні методи, rule-based аналіз і одношаровий перцептрон для оцінювання ризику;
- забезпечити інтеграцію моделі з платформою Zabbix та впровадити механізм автоматичного сповіщення про виявлені аномалії.

Об’єктом дослідження є процес аналізу мережевого трафіку в комп’ютерних мережах.

Предметом дослідження є методи та моделі інтелектуального виявлення аномалій у мережевому трафіку, що поєднують статистичні підходи, rule-based логіку та елементи машинного навчання.

РОЗДІЛ 1. ОГЛЯД СИСТЕМ АНАЛІЗУ МЕРЕЖЕВОГО ТРАФІКУ ТА МЕТОДІВ ВИЯВЛЕННЯ КІБЕРАТАК

1.1. Аналіз існуючих рішень для моніторингу мережевого трафіку

Системи моніторингу мережі відіграють ключову роль у забезпеченні стабільної та безпечної роботи сучасних інформаційних систем. Вони включають програмні та апаратні засоби, які відстежують різні параметри мережі, такі як трафік, пропускна здатність, доступність сервісів та безпека. Мережеві адміністратори використовують ці системи для швидкого виявлення проблем, запобігання вузьким місцям у трафіку та захисту від потенційних загроз.

Ефективність моніторингу залежить від застосовуваних методів аналізу трафіку, які можуть варіюватися залежно від цілей та вимог організації. Серед них можна виділити:

- Визначення джерел даних для аналізу.
- Виявлення пристроїв і додатків у мережі.
- Аналіз і оптимізацію пропускної здатності.
- Використання спеціалізованого програмного забезпечення для моніторингу та аналізу трафіку.

Застосування цих методів дозволяє значно покращити контроль за мережевою інфраструктурою та забезпечити її стабільність. Основні переваги моніторингу мережевого трафіку включають:

- Оптимізацію продуктивності серверів і мережевих пристроїв.
- Виявлення додатків та пристроїв, що споживають найбільше ресурсів.
- Підвищення рівня безпеки завдяки виявленню аномалій.
- Зниження простоїв та економію ресурсів організації.
- Зручну візуалізацію трафіку та доступ до історичних даних.

Для досягнення цих переваг використовуються різні типи систем моніторингу, кожен з яких має свої особливості та сфери застосування. Сучасні рішення для моніторингу мережевого трафіку можна розділити на кілька основних категорій:

1. Системи аналізу логів та подій (SIEM – Security Information and Event Management). SIEM-системи не тільки аналізують події в реальному часі, але й виконують кореляцію даних з різних джерел, що дозволяє ефективно виявляти складні загрози та аномалії, які можуть бути пропущені при аналізі окремих подій. Такі платформи забезпечують інтеграцію з різними типами обладнання та програмного забезпечення, дозволяючи централізовано зберігати та обробляти лог-файли. Наприклад, Splunk дозволяє проводити глибокий аналіз з великим обсягом даних [1], а IBM QRadar забезпечує кореляцію подій для виявлення складних атак, таких як інсайдерські загрози або складні багатофазні атаки [2].

2. Системи аналізу поведінки трафіку (NBAD – Network Behavior Anomaly Detection). NBAD-системи використовують різні алгоритми машинного навчання, які дозволяють їм адаптуватися до зміни нормальної поведінки мережі. Це дає змогу виявляти нові загрози, навіть якщо вони не були визначені заздалегідь. Наприклад, Darktrace застосовує "Enterprise Immune System" для побудови моделей нормальної поведінки кожного пристрою в мережі та виявлення аномалій в реальному часі [3]. Cisco Stealthwatch також використовує технології машинного навчання для виявлення підозрілих аномалій, які можуть вказувати на кібератаки або витoki даних [4].

3. Класичні системи мережевого моніторингу (NMS – Network Monitoring Systems). Крім базового моніторингу, NMS-системи можуть надавати детальну інформацію про стан обладнання, наявність помилок або відмов в мережі, а також аналізувати якість передачі даних по каналах зв'язку. Zabbix, наприклад, дозволяє створювати складні шаблони моніторингу для різних типів пристроїв і протоколів, надаючи адміністраторам точну інформацію про можливі несправності [5]. PRTG Network Monitor використовує вбудовані сенсори для моніторингу таких параметрів, як пропускна здатність, затримка або втрата пакетів, що дає змогу швидко виявляти і вирішувати проблеми з мережею [6].

4. Платформи візуалізації мережевого трафіку. Інструменти для візуалізації, такі як Grafana та Kibana, є потужними засобами для створення

інтуїтивно зрозумілих та настроюваних дашбордів, які дозволяють візуально відстежувати стан мережі в реальному часі. Grafana інтегрується з різними джерелами даних, такими як Prometheus або InfluxDB, що дозволяє здійснювати гнучкий моніторинг і аналіз. Kibana, в свою чергу, є частиною Elastic Stack і спеціалізується на візуалізації логів та даних, що зберігаються в Elasticsearch, що дозволяє отримати глибокий аналіз історичних даних і виявляти потенційні загрози або тренди.

Згідно з цими категоріями, існує багато програмних рішень для моніторингу мережевого трафіку, кожне з яких має свої особливості та сфери застосування. Далі проведемо огляд найбільш поширених та ефективних з них:

1. Zabbix – це система з відкритим кодом для моніторингу продуктивності та доступності серверів, мережі, а також додатків. Вона дозволяє здійснювати моніторинг у реальному часі за допомогою SNMP, агентського та без-агентського моніторингу. Крім того, Zabbix надає можливість аналізувати продуктивність IT-інфраструктури, проводити оповіщення про проблеми та генерувати звіти, що робить її ефективним інструментом для виявлення несправностей і оптимізації роботи мереж [7].

2. Grafana є потужним інструментом для візуалізації даних, який дає змогу створювати інтуїтивно зрозумілі та динамічні панелі управління (дашборди), що дозволяють аналізувати метрики мережевого трафіку та інші показники в реальному часі [8]. Вона інтегрується з багатьма джерелами даних, такими як Prometheus, InfluxDB, Elasticsearch, що дозволяє аналізувати метрики трафіку, стан серверів та інфраструктури в цілому.

3. Splunk – це SIEM-платформа, яка використовує великі обсяги даних для збору, зберігання та аналізу інформації з різних джерел у реальному часі. Вона дозволяє здійснювати кореляцію подій та забезпечує глибокий аналіз для виявлення безпекових загроз, здійснення аудитів і оперативної аналітики [9]. Splunk часто використовують для моніторингу безпеки, а також для керування інцидентами в IT-інфраструктурі.

4. IBM QRadar – це комплексна платформа для аналізу безпеки та управління подіями. Вона здійснює кореляцію даних з різних джерел для ефективного виявлення та дослідження загроз, а також автоматизує процеси управління безпекою, включаючи реагування на інциденти. QRadar підтримує інтерфейси для інтеграції з різними системами, що дозволяє застосовувати масштабовані рішення для виявлення складних атак.

5. Darktrace – це інноваційна NBAD-система, що застосовує штучний інтелект для автоматичного виявлення аномалій у мережевому трафіку. Вона здатна будувати моделі нормальної поведінки для кожного пристрою в мережі та виявляти підозрілі аномалії, які можуть вказувати на атаки або внутрішні загрози. Darktrace використовує «Enterprise Immune System», що дозволяє своєчасно виявляти навіть нові типи атак, не відомі раніше.

6. Cisco Stealthwatch – це інструмент для аналізу та моніторингу мережевого трафіку, що використовує передові методи поведінкового аналізу і машинного навчання для виявлення підозрілих аномалій. Він дозволяє виявляти загрози в реальному часі та отримувати цінну інформацію для розслідувань безпеки. Stealthwatch також здатен аналізувати мережевий трафік на рівні додатків і користувачів для забезпечення більш точного виявлення внутрішніх загроз.

7. Nagios – це одна з найстаріших і найпопулярніших систем для моніторингу мережі та сервісів [10]. Вона використовує простий, але потужний підхід до контролю стану мережевих пристроїв, серверів і додатків. Nagios є гнучким інструментом, що дозволяє налаштовувати сповіщення, використовувати плагіни та інтегрувати різноманітні інструменти для розширення можливостей системи.

8. PRTG Network Monitor – це комплексна система для моніторингу, яка дозволяє здійснювати збір даних через SNMP, NetFlow, WMI та інші протоколи. Вона забезпечує детальний моніторинг мережевих пристроїв, сервісів та додатків, включаючи функції для виявлення проблем із пропускнуою здатністю, доступністю та затримками в мережі [11].

9. Elastic Stack (ELK) – це набір інструментів для збору, зберігання, аналізу та візуалізації логів і мережевого трафіку [12]. Включає Elasticsearch, Logstash і Kibana, що дозволяють обробляти та аналізувати величезні обсяги даних. ELK Stack є надзвичайно популярним для збору великих обсягів логів і здійснення швидкого пошуку через великі бази даних, що допомагає виявляти тренди та загрози в інфраструктурі.

10. Kibana – це платформа для візуалізації та аналітики даних, що є частиною Elastic Stack. Вона дозволяє створювати інтерактивні та динамічні дашборди для глибокого аналізу мережевого трафіку та інших великих наборів даних. Kibana застосовується в поєднанні з Elasticsearch для виявлення аномалій і аналізу логів, надаючи користувачам можливість вивчати дані в реальному часі та знаходити потенційні проблеми [13].

Таблиця 1.1

Порівняння систем моніторингу мережевого трафіку

Система	Тип системи	Візуалізація	Виявлення загроз	Підтримка машинного навчання	Відкритий код	Призначення
Zabbix	NMS	Так	Обмежене	Ні	Так	Моніторинг продуктивності мережі
Grafana	Візуалізація	Так	Ні	Ні	Так	Аналітика даних
Splunk	SIEM	Так	Так	Так	Ні	Аналіз логів та кіберзагроз
IBM QRadar	SIEM	Так	Так	Так	Ні	Виявлення та кореляція загроз
Darktrace	NBAD	Так	Так	Так	Ні	Виявлення аномалій

Продовження таблиці 1.1

Система	Тип системи	Візуалізація	Виявлення загроз	Підтримка машинного навчання	Відкритий код	Призначення
Cisco Stealthwatch	NBAD	Так	Так	Так	Ні	Поведінковий аналіз трафіку
Nagios	NMS	Так	Ні	Ні	Так	Моніторинг інфраструктури
PRTG Network Monitor	NMS	Так	Ні	Ні	Ні	Моніторинг мережі
Elastic Stack (ELK)	SIEM	Так	Так	Ні	Так	Аналіз логів та безпеки
Kibana	Візуалізація	Так	Ні	Ні	Так	Аналітика мережевого трафіку

Для побудови системи інтелектуального аналізу мережевого трафіку необхідно обрати інструмент, який забезпечить стабільний збір телеметрії, гнучку інтеграцію через API та можливість подальшої аналітичної обробки даних. Серед доступних відкритих рішень найчастіше розглядаються універсальні системи моніторингу та платформи зберігання великих даних. Вибір конкретного інструмента залежить від поставлених задач, обсягу даних, вимог до масштабованості та складності інтеграції.

У межах наукового підходу до проектування системи було враховано такі критерії:

- наявність механізмів збору мережевих метрик у реальному часі;
- наявність стандартизованого REST API для доступу до даних;
- мінімальна складність розгортання у лабораторному середовищі;
- підтримка гнучких правил і порогових значень;

- можливість подальшого розширення аналітичного функціоналу.

На основі цих критеріїв було обрано систему Zabbix, оскільки вона поєднує необхідні можливості для проведення дослідження, забезпечує простий доступ до даних через API, підтримує багатий набір мережових метрик та не потребує значних обчислювальних ресурсів під час розгортання. Це дозволяє зосередитись не на інфраструктурних аспектах, а на аналітичній частині дослідження та побудові прототипу системи виявлення аномалій.

1.2. Методи інтелектуального аналізу даних у мережевій безпеці

Інтелектуальний аналіз даних у мережевій безпеці є одним з ключових напрямів сучасного захисту інформаційних систем від різноманітних кіберзагроз. Завдяки розвитку технологій обробки великих даних та штучного інтелекту, методи аналізу даних стали основою для побудови ефективних систем виявлення загроз. Ці системи здатні аналізувати величезні обсяги даних, виявляти аномалії в мережевому трафіку та швидко реагувати на можливі атаки [14]. Основними методами інтелектуального аналізу є машинне навчання (Machine Learning), глибоке навчання (Deep Learning) та алгоритми видобування даних (Data Mining) [15]. Структурне представлення аналітичного екопростору наведено на рис. 1.1.

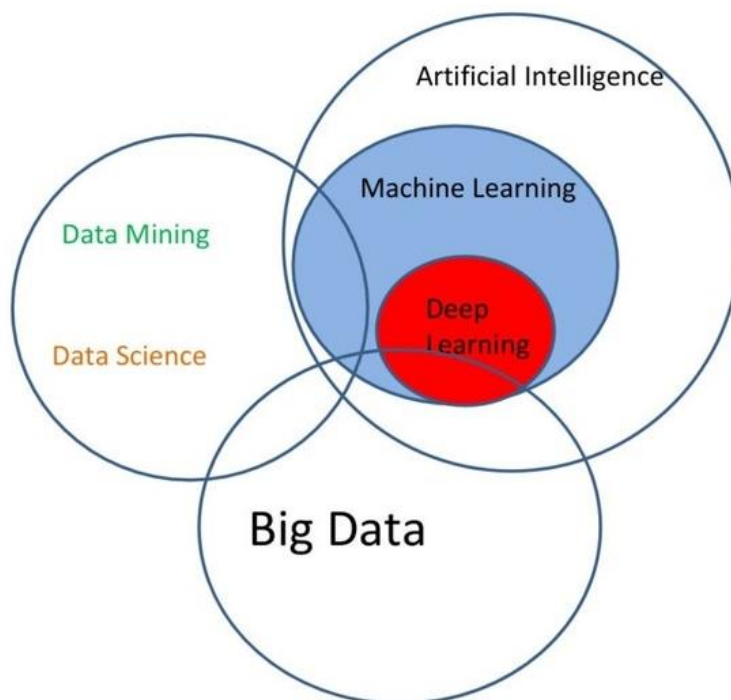


Рисунок 1.1 – Схема представлення аналітичного екопростору

1. Машинне навчання є потужним інструментом для аналізу даних у мережній безпеці, адже дозволяє автоматично виявляти шаблони та аномалії в поведінці мережі, які можуть свідчити про наявність кіберзагроз. Загальну класифікацію типів машинного навчання подано на рисунку 1.2.

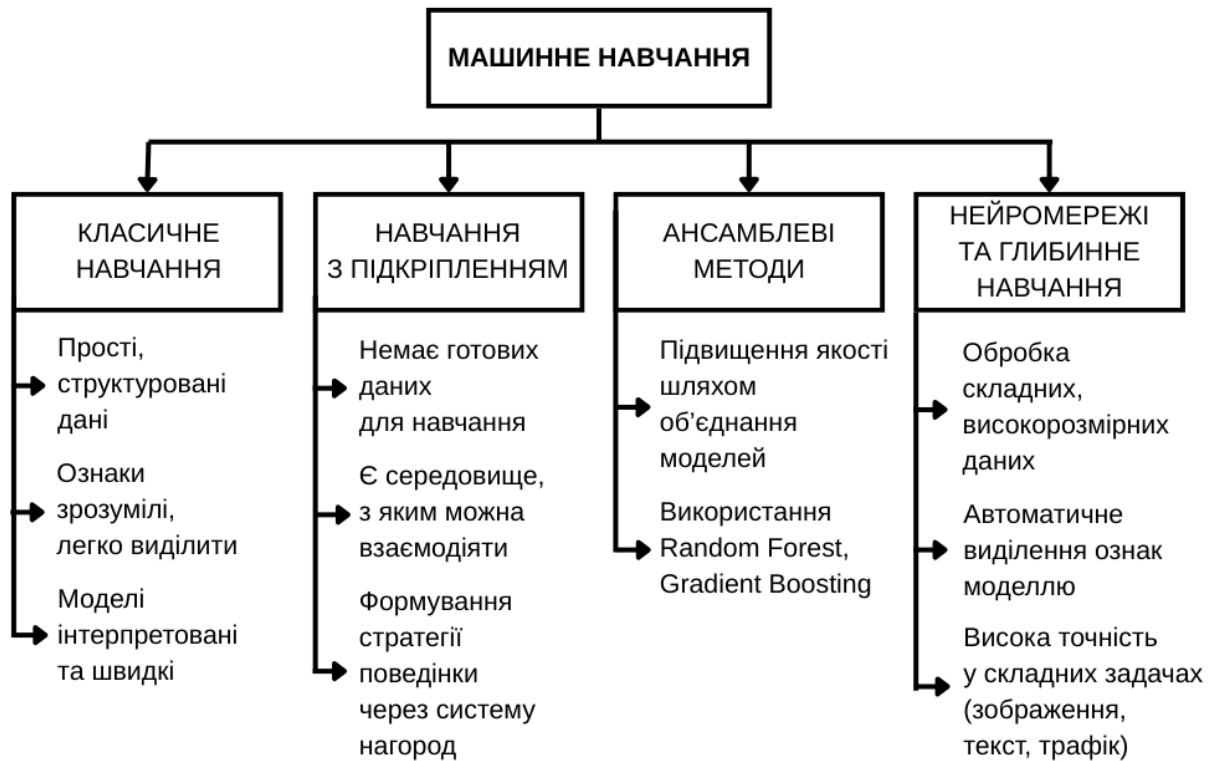


Рисунок 1.2 – Схема основних типів машинного навчання

Алгоритми машинного навчання поділяються на два основні типи: навчання з учителем (supervised learning) та навчання без учителя (unsupervised learning) [16], загальна схема яких наведена на рисунку 1.2.

- Навчання з учителем використовується для класифікації даних на основі вже наявних позначених (мітованих) даних, наприклад, виявлення шкідливих програм чи аналіз поведінки користувачів. Під час навчання система використовує історичні дані з відомими результатами для створення моделі, яку потім можна застосувати до нових даних для класифікації.

- Навчання без учителя застосовується для виявлення аномалій або незвичних патернів у мережевому трафіку, де немає чіткої мітки для кожного випадку. Цей метод дозволяє системі самостійно визначати, що є нормою для мережі, а

що є аномалією. Наприклад, виявлення незвичних обсягів трафіку або підозрілих змін у поведінці пристроїв.

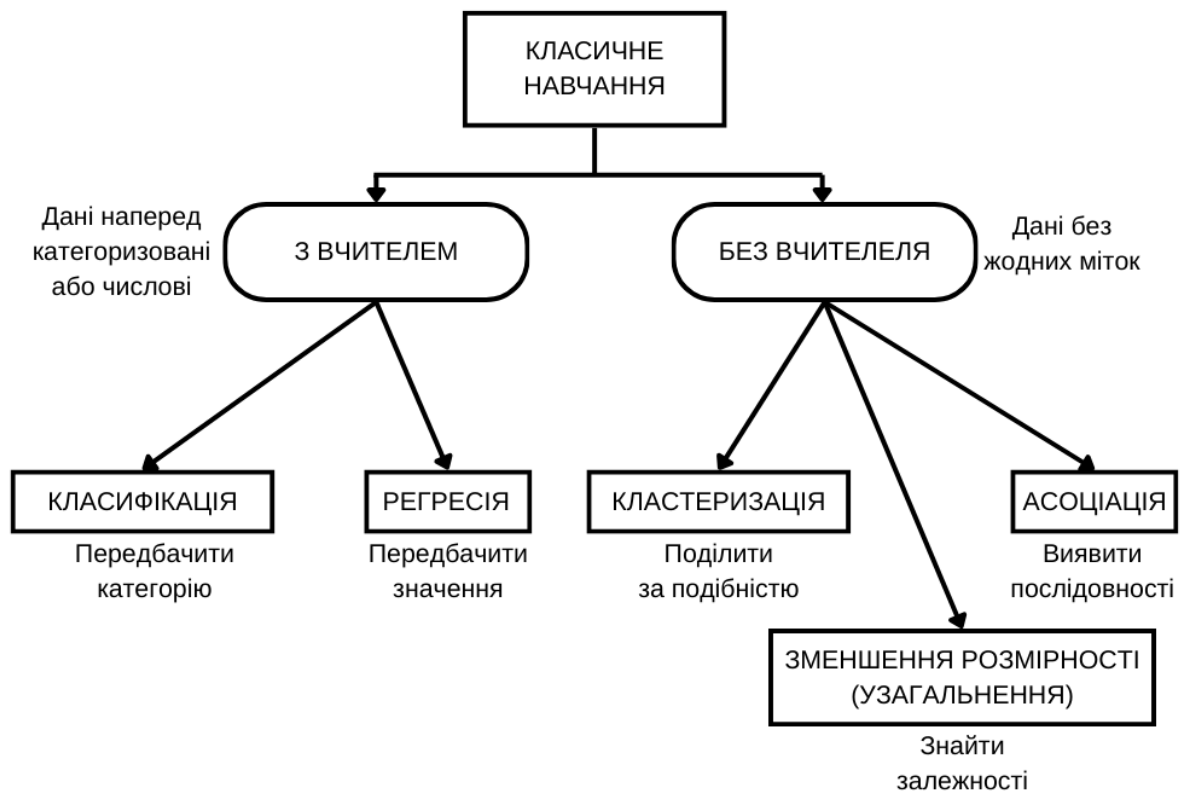


Рисунок 1.3 – Схема класичного алгоритму навчання

Машинне навчання дозволяє адаптуватися до нових загроз, навіть якщо вони не були заздалегідь відомі, що робить його потужним інструментом для забезпечення безпеки мережі в умовах постійно змінюваних атак.

2. Глибинне навчання є підгалуззю машинного навчання, яке застосовує нейронні мережі з багатьма шарами для автоматичного виявлення складних патернів у даних [17]. В глибинному навчанні використовуються великі обсяги даних для тренування моделей, що дозволяє досягати високої точності виявлення складних атак, таких як zero-day атаки (атаки нульового дня) або нові варіанти шкідливого ПЗ. Цей метод здатний обробляти великі масиви даних і знаходити зв'язки, які можуть бути неочевидними для традиційних алгоритмів аналізу. Наприклад, глибинні нейронні мережі можуть аналізувати мережевий трафік та виявляти підозрілі шаблони або аномалії, навіть якщо вони є складними, багатовимірними або раніше невідомими [18].

Поряд із високорівневими моделями глибинного навчання в мережевій безпеці застосовуються і більш прості інтелектуальні підходи, які, попри свою обмежену складність, залишаються ефективними у реальних системах моніторингу. Зокрема, правилі (rule-based) моделі використовують набір логічних умов або порогів, які визначають, коли мережевий трафік вважається аномальним [19]. Такий підхід часто використовується в оперативних системах безпеки, де важливо швидко реагувати на відхилення у показниках. Перевагою правиліх систем є їхня простота, пояснюваність і можливість адаптації під конкретне мережеве середовище.

Окрім цього, у практиці інтелектуального аналізу можуть застосовуватися й найпростіші моделі машинного навчання, такі як перцептрон – одношаровий нейрон, який виконує лінійну комбінацію вхідних параметрів і може класифікувати події на «нормальні» та «аномальні». Попри свою структурну простоту, перцептрон здатний узагальнювати дані та формувати чисельну оцінку ймовірності наявності загрози, що дозволяє використовувати його як доповнення до правиліх методів [20]. У поєднанні ці два підходи забезпечують і прозорість прийняття рішень, і елементи інтелектуальної оцінки ризиків, що є корисним для систем виявлення аномалій у мережевому трафіку.

3. Аналіз даних або data mining включає в себе використання різноманітних алгоритмів для пошуку шаблонів, кореляцій та аномалій у великих наборах даних. В умовах мережевої безпеки це означає використання таких методів, як кластеризація, асоціативний аналіз та регресія для виявлення підозрілих подій чи поведінки користувачів [21].

- Кластеризація допомагає автоматично групувати дані на основі схожих характеристик. Наприклад, у мережевому трафіку можна застосувати цей метод для виявлення груп пристроїв або користувачів, чиї дії є аномальними або відрізняються від звичних.

- Асоціативний аналіз дозволяє знаходити закономірності у взаємодії між різними об'єктами, наприклад, між типами запитів до серверів чи між кон-

кретними IP-адресами та інтервалами часу. Це допомагає виявляти можливі вектори атаки.

- Регресія застосовується для моделювання та прогнозування поведінки на основі даних, що дозволяє передбачити майбутні загрози на основі поточного стану системи.

Ці методи дозволяють автоматично знаходити приховані загрози та тренди, що можуть бути неочевидними під час традиційного аналізу.

4. Аналіз великих даних застосовується для обробки та аналізу величезних обсягів інформації, що генерується в реальному часі в мережах. Це включає дані з різних джерел, таких як журнали подій, збережені дані з мережевих пристроїв, звіти про аномальні події, а також історичні дані.

За допомогою технологій Big Data (наприклад, Hadoop чи Apache Spark) можна обробляти та аналізувати величезні обсяги даних, що дозволяє створювати точніші моделі для виявлення загроз і більш оперативно реагувати на них [22]. Цей метод є дуже корисним для великих організацій, які мають розгалужену мережу та велику кількість підключених пристроїв, оскільки здатен обробляти дані у реальному часі, що дозволяє швидко реагувати на можливі інциденти безпеки.

5. Аналіз поведінки мережі базується на вивченні регулярностей і аномалій в мережевому трафіку для виявлення підозрілих активностей. Це включає в себе вивчення таких параметрів, як обсяг трафіку, частота запитів, типи з'єднань тощо. Зазвичай використовується в поєднанні з іншими методами машинного навчання, щоб ідентифікувати підозрілу активність, яка може свідчити про наявність кібератаки.

Цей підхід є корисним для виявлення таких атак, як DDoS, brute force або навіть для визначення втечі даних, коли порушники намагаються вивести конфіденційну інформацію з мережі організації.

Використання інтелектуальних методів для аналізу мережевого трафіку дозволяє значно підвищити ефективність систем безпеки. Застосування машин-

ного навчання та глибинного навчання, зокрема, дозволяє автоматизувати процес виявлення нових типів атак, що зменшує залежність від людського фактору та час реакції на загрози. Оскільки ці методи здатні вивчати та аналізувати великі обсяги даних, вони здатні виявляти навіть малопомітні аномалії, що є важливим у контексті сучасних кібератак, які стають дедалі складнішими та витонченішими.

Такі методи, як аналіз поведінки мережі та аналіз великих даних, дозволяють мережевим адміністраторам отримати можливість у реальному часі відстежувати підозрілі зміни, на основі яких можна оперативно приймати рішення щодо блокування чи обмеження доступу до ресурсів. Використання моделей, заснованих на поведінковому аналізі, підвищує точність виявлення загроз, оскільки орієнтується не лише на відомі атаки, а й на нові, ще невизначені. Методи кластеризації та асоціативного аналізу допомагають швидко виявляти нові шаблони та тенденції в мережевому трафіку, що сприяє своєчасному реагуванню на появу нових атак. Це дає змогу не лише ефективно блокувати поточні загрози, а й запобігати майбутнім атакам завдяки розпізнаванню схожих поведінкових ознак.

Попри ефективність таких підходів, їх застосування часто потребує значних обчислювальних ресурсів, великих обсягів даних і складної інфраструктури для навчання та підтримки моделей. Це робить глибинний аналіз і поведінкове моделювання дорогими рішеннями, які не завжди доступні для невеликих організацій, локальних мереж або навчальних середовищ. У таких випадках доцільним є впровадження простіших моделей інтелектуальних систем (наприклад, з одним шаром та rule-based механізмів із ваговими коефіцієнтами). Вони забезпечують базовий рівень інтелектуального аналізу, дозволяють оцінювати ризики та виявляти аномалії без потреби у великих витратах на інфраструктуру, що робить їх оптимальними для швидкого розгортання та застосування в сценаріях із обмеженими ресурсами.

Перспективи застосування таких систем передбачають інтеграцію різних підходів і технологій для забезпечення глибшого аналізу, ніж це можливо при

використанні лише традиційних методів моніторингу та статичних правил. Таким чином, поєднання простих інтелектуальних моделей із класичними методами моніторингу може стати ефективною основою для створення доступних та гнучких рішень у сфері кібербезпеки, здатних адаптуватися до динамічних умов сучасного кіберпростору.

1.3.Сучасні кібератаки та характерні ознаки їх виявлення у мережевому трафіку

Кіберзагрози стали серйозною проблемою для організацій, адже їхня кількість і складність невідомо зростають. Вони можуть завдати значної шкоди як окремим користувачам, так і великим підприємствам. Одним з основних елементів боротьби з кіберзагрозами є виявлення їхніх ознак у мережевому трафіку, що дозволяє своєчасно реагувати та вживати необхідні заходи безпеки. Щоб зрозуміти, як виникають та поширюються кібератаки, варто розглянути їхні етапи і найпоширеніші типи.

Кібератака зазвичай складається з кількох чітко визначених етапів, кожен з яких має свою специфіку та тактику [23]. Розглянемо детальніше основні етапи кібернападу:

1. Розвідка (Reconnaissance). На першому етапі зловмисники здійснюють розвідку, щоб зібрати якомога більше інформації про потенційну ціль. Це включає виявлення вразливостей у системах, дослідження архітектури мережі та пошук точок входу. Вони можуть використовувати публічно доступні джерела, такі як соціальні мережі або інструменти сканування мереж, для пошуку уразливих місць в інфраструктурі організації.

2. Зброя (Weaponization). На цьому етапі зловмисники створюють або отримують шкідливе програмне забезпечення, яке буде використовуватися для атаки. Вони розробляють шкідливі інструменти, такі як віруси, трояни або інші види шкідливого ПЗ, що здатні скористатися виявленими вразливостями в системах або мережах цілі. Ці інструменти можуть бути спрямовані на руйнування, крадіжку даних чи забезпечення контролю над системами.

3. Доставка (Delivery). Після того як шкідливе програмне забезпечення підготовлене, зловмисники приступають до його доставки до цільової системи. Це може відбуватися через електронні листи з вкладеннями, фішингові посилення, експлойти у вразливому програмному забезпеченні чи інші методи. Метою доставки є проникнення в мережу жертви, що дозволяє зловмиснику здобути доступ до її систем.

4. Експлуатація (Exploitation). Після доставки шкідливого коду наступним етапом є експлуатація вразливостей для проникнення в цільову систему. Зловмисники можуть скористатися такими уразливими місцями, як слабкі паролі, неоновлене програмне забезпечення, чи навіть логічні помилки в коді. Як тільки доступ здобуто, атакуючі намагаються підвищити свої привілеї, що дає змогу отримати більший контроль над системою.

5. Інсталяція (Installation). Після того як зловмисник отримав доступ до системи, він встановлює додаткове шкідливе програмне забезпечення чи інструменти для забезпечення постійного доступу. Це можуть бути «троянські» програми або програми для віддаленого управління, які дають можливість здійснювати подальші атаки, спостереження або викрадення даних.

6. Командування та контроль (C2). Зловмисники встановлюють інфраструктуру для командування та контролю, що дозволяє їм віддалено управляти скомпрометованими системами. На цьому етапі створюється канал зв'язку між атакуючим і атакованою мережею, що дає змогу відправляти команди, маніпулювати даними або навіть викрадати інформацію без відома користувачів.

7. Досягнення цілей (Actions on Objectives). Після того як контроль над системою отримано, зловмисники переходять до фінальної фази – реалізації своїх цілей. Це може включати крадіжку конфіденційної інформації (наприклад, фінансові дані або паролі), порушення роботи сервісів або навіть саботаж. В залежності від мети кібернападу, результатом може стати фінансовий збиток, репутаційні втрати чи витік чутливої інформації.

Кожен етап кібератаки має важливе значення і тісно пов'язаний з наступними. Чим більше інформації зловмисники отримують на етапі розвідки, тим

ефективніше вони можуть підготувати шкідливі інструменти для атаки. Після того, як вони здійснили доставку шкідливого програмного забезпечення та експлуатували вразливості, виникає необхідність вибору відповідної техніки для досягнення своїх цілей. Саме тут особливо важливою є класифікація типів атак, адже кожен з них має свої особливості, методи реалізації та вплив на цільову систему.

Розглянемо та проаналізуємо основні типи кібератак та визначимо характерні ознаки, за якими їх можна ідентифікувати у мережевому середовищі. Такий аналіз дає змогу зрозуміти механізм реалізації атак і визначити прояви, що можуть сигналізувати про аномальну або шкідливу активність.

1. Фішингові атаки є одним із найпоширеніших методів отримання конфіденційних даних шляхом соціальної інженерії. Зловмисники використовують підроблені листи чи вебсайти, що імітують легітимні ресурси, спонукаючи користувача розкрити чутливу інформацію [24].

Ознаками фішингу можуть бути:

- використання підозрілих або нетипових доменів в електронних повідомленнях;
- наявність гіперпосилань, що перенаправляють на неавторизовані або новостворені сайти;
- заклики до термінових дій з боку користувача;
- виявлення нетипових шаблонів у мережевих HTTP/HTTPS-запитах, спрямованих на фальшиві ресурси.

2. Атаки на основі шкідливого програмного забезпечення (malware) включають віруси, трояни, програми-вимагачі та інші типи шкідливих компонентів, що можуть порушувати роботу системи або перехоплювати дані [25].

Ознаки атак за допомогою шкідливого ПЗ:

- поява аномальних фонових процесів або служб;
- нетипові спроби встановлення вихідних з'єднань зі сторонніми серверами;

- зростання кількості системних помилок або некоректна робота застосунків;

- зміна параметрів безпеки або зупинка захисного програмного забезпечення;

- фіксація підозрілих файлів чи активності у системних каталогах.

3. Атаки типу DoS і DDoS (відмова в обслуговуванні). Атаки DoS (Denial of Service) і DDoS (Distributed Denial of Service) спрямовані на перевантаження сервісу або мережі великою кількістю запитів, що призводить до відмови в обслуговуванні. В результаті ці атаки можуть спричинити затримки в роботі чи навіть повне припинення доступу до вебсайтів або сервісів [26].

Ознаки DoS/DDoS атак:

- різке та непропорційне збільшення обсягу вхідного трафіку;
- велика кількість однотипних запитів у короткі проміжки часу;
- аномальне використання системних ресурсів (CPU, RAM, мережевих інтерфейсів);

- падіння продуктивності або недоступність сервісів.

4. Атаки типу Man-in-the-Middle (MitM) передбачають перехоплення або модифікацію трафіку між двома учасниками комунікації. Зловмисник може підслуховувати та змінювати передану інформацію, що дозволяє йому отримати доступ до конфіденційних даних, таких як паролі, фінансові реквізити або інші особисті відомості [27].

Ознаки MitM атак:

- зміна SSL/TLS-сертифікатів або виявлення сертифікатів, що не відповідають очікуваним;

- фіксація ARP- або DNS-підміни;
- аномальна маршрутизація трафіку чи збільшення затримок;
- порушення цілісності даних у переданих пакетах.

5. SQL-ін'єкції – це атаки, які використовують вразливості в базах даних для того, щоб виконувати небажані SQL-запити. Це дає можливість зловмисникам отримати доступ до конфіденційної інформації або маніпулювати даними [28].

Ознаки можливих SQL-ін'єкцій:

- незвичні або некоректно сформовані HTTP-запити до вебресурсів;
- аномальні записи в логах бази даних;
- збільшення кількості помилок SQL-процесів;
- поява нетипових даних у відповідях вебсервера.

6. Zero-day експлойти використовують ще невідомі або не виправлені вразливості програмного забезпечення, що робить їх особливо небезпечними. Оскільки виробник ще не випустив оновлення безпеки, зловмисники можуть використовувати ці вразливості для атак до того, як патчі стануть доступними [29].

Ознаки Zero-day атак:

- неочікувані збої системних служб;
- фіксація нетипових змін у конфігураціях без оновлень;
- виконання нових підозрілих процесів або скриптів;
- аномальні підключення чи спроби взаємодії із зовнішніми вузлами.

7. Атаки соціальної інженерії орієнтовані на маніпулювання людьми з метою отримання конфіденційної інформації. Зловмисники можуть використовувати різні психологічні методи для того, щоб обдурити користувачів і змусити їх надати доступ до облікових записів, паролів або інших чутливих даних [30].

Ознаки атак соціальної інженерії:

- запити на передачу критично важливих даних через нестандартні канали комунікації;
- спроби створення стану терміновості або психологічного тиску;
- взаємодія з невідомими особами, що прагнуть отримати системні доступи;
- аномальна активність у користувацьких облікових записах.

Для наочності та систематизації отриманих результатів проведемо узагальнений порівняльний аналіз основних типів кібератак за їх ключовими характеристиками.

Аналіз основних типів кібератак

Тип атаки	Мета атаки	Механізм дії	Характерні ознаки у мережевому трафіку	Можливі наслідки	Рівень ризику
Phishing	Отримання конфіденційних даних користувача	Використання підроблених листів і вебресурсів	Переходи на підозрілі домени, нетипові HTTP-запити, взаємодія з ресурсами низької репутації	Компрометація облікових записів, крадіжка персональних і фінансових даних	Високий
Malware	Несанкціонований доступ, викрадення або блокування даних	Завантаження та виконання шкідливого ПЗ на пристрої жертви	Підозрілі outbound-з'єднання, активація невідомих процесів, передача зашифрованого трафіку на зовнішні сервери	Порушення роботи системи, витік даних, шифрування та втрата інформації	Високий
DoS / DDoS	Порушення доступності сервісів	Перевантаження мережі або сервера великою кількістю запитів	Різкий сплеск вхідного трафіку, однотипні запити з багатьох джерел, зниження продуктивності сервісів	Недоступність сервісів, фінансові збитки, зупинка бізнес-процесів	Високий
Man-in-the-Middle	Перехоплення або зміна даних між учасниками комунікації	Підміна трафіку, ARP-спуфінг, модифікація SSL-з'єднань	Невірні SSL-сертифікати, зміни маршрутизації, підвищена затримка, дублювання ARP-записів	Викрадення даних, компрометація сесій, втручання в комунікацію	Високий
SQL-Injection	Отримання доступу до бази даних або маніпуляція її вмістом	Вставка шкідливого SQL-коду в параметри вебзапиту	Некоректні або аномальні HTTP-запити, помилки SQL у відповідях сервера, часті повтори запитів до одного ресурсу	Несанкціонований доступ, спотворення або видалення даних, повний контроль над БД	Високий
Zero-day Exploit	Використання невідомих вразливостей	Експлуатація помилок у ПЗ до виходу оновлень безпеки	Нові аномальні процеси, неочікувані збої, нетипові підключення до зовнішніх вузлів	Повний доступ до системи, масштабний компрометаційний вплив	Дуже високий

Тип атаки	Мета атаки	Механізм дії	Характерні ознаки у мережевому трафіку	Можливі наслідки	Рівень ризику
Social Engineering	Отримання доступу через психологічний вплив на людину	Маніпуляції, створення терміновості, використання довіри користувача	Нетипові запити на передачу даних, активність із незвичних джерел, взаємодія з підозрілими контактами	Компрометація облікових записів, порушення конфіденційності, вторинні атаки	Середній/високий

Отже, проведений аналіз дає змогу визначити ключові особливості різних категорій кібератак та зрозуміти, які саме ознаки можуть свідчити про їхню наявність у мережевому середовищі. Виділення таких ознак є важливим елементом забезпечення безпеки, оскільки дозволяє своєчасно виявляти потенційні загрози та підвищувати рівень захищеності інформаційних систем. Це створює основу для подальшого вдосконалення методів аналізу трафіку та практичного застосування інструментів моніторингу.

1.4. Постановка завдання дослідження

На основі проведеного аналізу існуючих рішень для моніторингу мережевого трафіку, методів інтелектуального аналізу даних та характерних ознак сучасних кібератак було встановлено, що традиційні засоби виявлення аномалій часто забезпечують лише базовий рівень контролю і не завжди дають можливість оперативно інтерпретувати зміни у поведінці мережі. Це підкреслює необхідність розвитку підходів, здатних поєднувати можливості класичного моніторингу з аналітичними методами, які дозволяють більш точно визначати потенційно небезпечні відхилення у трафіку.

Разом із тим, сучасні високоточні методи глибинного аналізу та поведінкового моделювання є ресурсомісткими, потребують значних обчислювальних потужностей та великих обсягів підготовлених даних. У реальних умовах експлуатації, особливо у невеликих мережах чи в освітніх і тестових середовищах,

оптимальним рішенням стають прості системи інтелектуального аналізу. Такі системи забезпечують баланс між точністю та обчислювальною складністю, дозволяючи виконувати базове інтелектуальне оцінювання трафіку, використовувати прості статистичні показники та елементи машинного навчання для своєчасного виявлення аномалій.

З огляду на це, тема дослідження є актуальною та зумовлює розробку системи інтелектуального аналізу мережевого трафіку, що є метою даної роботи. Для досягнення мети передбачається використання статистичних методів оцінювання, rule-based логіки, а також елементарного машинного навчання на базі одношарового перцептрона, який навчається на наборі даних мережевих метрик. Інтеграція з платформою Zabbix забезпечує можливість одночасного збору даних і онлайн-оцінювання стану трафіку.

Відповідно, для досягнення поставленої мети було визначено такі задачі:

1. Провести аналіз існуючих рішень моніторингу мережевого трафіку та методів виявлення кібератак, з метою визначення їх можливостей, обмежень та придатності до інтеграції з аналітичними механізмами, що дозволить сформулювати теоретичне підґрунтя для подальшої розробки системи.

2. Розробити модель системи інтелектуального аналізу мережевого трафіку, що поєднує збір даних за допомогою Zabbix із статистичними методами виявлення аномалій та оцінювання ризиків на базі одношарового перцептрона, із застосуванням спеціалізованих бібліотек для обробки даних та елементарного машинного навчання, визначивши її структуру та ключові параметри.

3. Реалізувати програмний комплекс, включно з інтеграцією з Zabbix API, збором і попередньою обробкою мережевих метрик, формуванням набору даних, виконанням навчання моделі та застосуванням механізмів аналітичного оцінювання в реальному часі, а також налаштуванням автоматичного сповіщення про виявлені аномалії.

4. Провести аналіз отриманих результатів та оцінити ефективність роботи системи інтелектуального аналізу мережевого трафіку, порівнявши очікувані показники із фактичними, визначивши сильні сторони, обмеження та можливі напрями її подальшого удосконалення.

Висновки до розділу 1:

У першому розділі проаналізовано сучасні засоби моніторингу мережевого трафіку та визначено інструмент, що найбільш відповідає вимогам дослідження. Розглянуто основні методи інтелектуального аналізу даних та ключові напрями їх застосування у мережевій безпеці. Також проведено огляд сучасних типів кібератак і характерних ознак їх прояву, здійснено їх порівняння та узагальнення. Отримані результати формують базу для подальшої розробки моделі системи виявлення аномальної активності.

РОЗДІЛ 2. ПРОЄКТУВАННЯ МОДЕЛІ ІНТЕЛЕКТУАЛЬНОГО АНАЛІЗУ МЕРЕЖЕВОГО ТРАФІКУ

2.1. Вибір технологій та методів для побудови моделі інтелектуального аналізу

Інтелектуальний аналіз мережевого трафіку є ключовим елементом сучасних систем кіберзахисту, оскільки дозволяє автоматично виявляти потенційні загрози та аномалії на основі поведінкових патернів. Його мета – зменшити вплив людського фактора, скоротити час реагування на інциденти та підвищити ефективність систем моніторингу.

Під час формування архітектури моделі важливо обґрунтувати вибір техно-логій, що забезпечують сумісність компонентів, ефективність обробки даних та можливість подальшого масштабування системи. Основними критеріями вибору інструментів для цього дослідження стали:

1. Сумісність із системами моніторингу та підтримка стандартних протоколів. Обрані технології повинні забезпечувати можливість інтеграції зі Zabbix та підтримувати SNMP, ICMP, API-запити, що дає змогу отримувати телеметрію в реальному часі [31].

2. Наявність розвиненої екосистеми для обробки даних та машинного навчання. Python має широкий набір бібліотек для аналізу часових рядів, статистичного моделювання та виявлення аномалій, що дозволяє швидко реалізовувати прототипи аналітичних моделей [32].

3. Масштабованість і гнучкість. Обрані засоби повинні підтримувати можливість розширення функціональності, інтеграції нових джерел даних та переходу від rule-based підходу до алгоритмів машинного навчання без зміни загальної архітектури.

4. Простота інтеграції та низький поріг входу. Технології мають бути зрозумілими, легко налаштовуваними й не вимагати надмірних ресурсів для розгортання, що особливо важливо на етапі науково-дослідної роботи.

5. Відкритий вихідний код та доступність. Використання безкоштовних рішень із відкритою архітектурою, таких як Zabbix та Python, забезпечує прозорість, можливість кастомізації та адаптації системи до потреб дослідження.

6. Підтримка автоматизації та інтеграції зі сторонніми сервісами. Наявність REST API у Zabbix і бібліотек для роботи з Telegram дозволяє створити замкнений цикл автоматизованої обробки даних – від збору телеметрії до відправлення сповіщень адміністратору.

Майбутня модель функціонуватиме як інтегрована система, у якій Zabbix забезпечує збір телеметрії, Python виконує інтелектуальну обробку мережевих метрик, а Telegram-бот реалізує механізм зворотного зв'язку з адміністратором. Завдяки такому поєднанню компонентів система не лише аналізує параметри мережі, а й автоматично реагує на виявлені відхилення, що підвищує оперативність та ефективність процесу моніторингу.

Zabbix – це потужна система моніторингу з відкритим кодом, що дозволяє контролювати стан серверів, мережевого обладнання, додатків і сервісів у режимі реального часу. На відміну від комерційних аналогів (наприклад, SolarWinds, Datadog чи PRTG), Zabbix є повністю безкоштовним і має відкриту архітектуру, що дозволяє легко інтегрувати його з іншими системами. Це робить Zabbix оптимальним вибором для цього проєкту.

Для аналітичного шару використовується мова програмування Python, яка завдяки своїй простоті та великій кількості бібліотек стала стандартом де-факто в галузі штучного інтелекту та обробки даних.

Основу аналітичного середовища становлять такі бібліотеки:

- pandas, numpy – збір і попередня обробка даних, нормалізація часових рядів;
- matplotlib, seaborn – візуалізація метрик і динаміки трафіку;
- scikit-learn – реалізація алгоритмів машинного навчання (кластеризація, класифікація, ізоляційне лісове моделювання);
- requests, pyzabbix – отримання даних із Zabbix API у форматі JSON.

Використання цих бібліотек створює основу для застосування алгоритмів машинного навчання, які забезпечують автоматичне виявлення аномалій у мережевому трафіку [33]. Моделі кластеризації та ізоляційні методи дозволяють знаходити нетипові відхилення, що можуть свідчити про перевантаження мережі, DDoS-атаки або витік даних. Завдяки цьому Python-модуль виконує роль інтелектуального шару, який отримує дані із Zabbix, аналізує їх відносно нормальної поведінки, визначає можливі аномалії та формує висновки для подальшого реагування.

На відміну від класичних систем моніторингу, які обмежуються фіксацією порогових значень, інтелектуальні системи здатні:

- адаптуватися до динамічних змін у мережі – моделі машинного навчання можуть оновлюватися, враховуючи нові дані;
- зменшувати кількість помилкових сповіщень, автоматично відсіюючи нетипові, але безпечні коливання;
- передбачати можливі інциденти на основі історичних трендів трафіку.

Таким чином, інтеграція аналітики Python у традиційний моніторинг Zabbix забезпечує розширену функціональність, перетворюючи систему з реактивної на проактивну.

Розроблений підхід дозволяє побудувати та протестувати модель інтелектуального аналізу мережевого трафіку у кілька послідовних етапів:

1. Розгортання та налаштування Zabbix-сервера. Встановлення системи моніторингу у віртуальному середовищі (наприклад, Ubuntu Server або Windows Server).
2. Додавання Zabbix Agent і збір тестових мережевих метрик. Це дасть змогу збирати показники навантаження процесора, ICMP-статистику, кількість активних TCP-з'єднань, швидкість інтерфейсів та інші параметри.
3. Розробка Python-модуля та реалізація інтелектуальної моделі аналізу трафіку, що поєднує правилкові умови з простою перцептронною моделлю. Ви-

користання бібліотеки `pyzabbix` для автоматичного отримання мережеских метрик, а також `pandas` та `numpy` для формування й обробки набору даних та виконання базових обчислень.

4. Тестування, візуалізація та інтеграція з Telegram-ботом. Побудова графіків зміни трафіку, визначення аномалій та відправлення результатів через Telegram-бота.

5. Інтерпретація та оцінювання роботи аналітичної системи. Оцінювання точності виявлення аномалій і кореляції між попередженнями Zabbix та висновками інтелектуальної моделі.

Такий підхід забезпечить послідовне формування та перевірку всіх компонентів моделі, дозволяючи оцінити коректність роботи алгоритмів і створити основу для подальшого розширення системи. Це робить модель гнучкою, масштабованою та здатною забезпечувати проактивний моніторинг мережевого трафіку.

2.2. Проектування архітектури моделі інтелектуального аналізу мережевого трафіку

Проектування архітектури майбутньої моделі інтелектуального аналізу є одним з ключових етапів побудови системи, здатної виявляти аномалії та потенційні кіберзагрози у режимі реального часу. Основна ідея полягає у створенні багаторівневої структури, де кожен рівень виконує окрему функцію, але водночас органічно взаємодіє з іншими. Такий підхід дозволяє забезпечити модульність, надійність, адаптивність та можливість подальшого масштабування системи.

Запропонована архітектура складається з трьох взаємопов'язаних рівнів: збору даних, аналітики та сповіщення, що формують повноцінний цикл обробки інформації [34]. Усі компоненти функціонують у межах єдиного інформаційного простору, а обмін даними між ними відбувається через стандартизовані протоколи, що гарантує незалежність модулів і спрощує їх подальшу модернізацію.

Загальна схема системи наведена на рисунку 2.1.

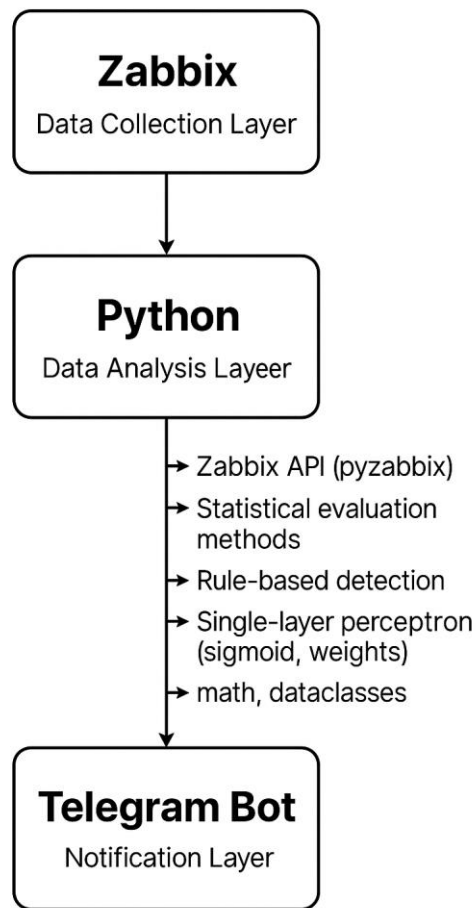


Рисунок 2.1 – Архітектура системи інтелектуального аналізу мережевого трафіку

1. Рівень збору даних (Data Collection Layer)

Перший рівень відповідає за формування масиву даних, який надалі буде аналізуватися інтелектуальною моделлю. На цьому рівні застосовується система Zabbix як надійний засіб централізованого збору телеметрії. Zabbix має розвинену інфраструктуру збору даних: він може опитувати хости через агент, використовувати ICMP для тестування доступності, збирати дані через SNMP, читати кількість активних TCP-сесій, визначати навантаження на процесор та інші параметри.

На цьому етапі фіксуються такі ключові показники:

- індикатор використання процесорних ресурсів;
- швидкість вхідного та вихідного трафіку;
- метрики доступності та стабільності каналу;

- кількість активних TCP-з'єднань, що може бути ознакою SYN-flood атаки;
- додаткові метрики стану системи, які можуть впливати на поведінку мережі.

Ці параметри є базовими індикаторами стабільності мережі та дозволяють виявляти перевантаження, збої або підозрілу активність на ранніх етапах.

Усі зібрані дані надходять до бази даних Zabbix Server, що зазвичай використовує MySQL або PostgreSQL [35]. У подальшому ці дані можуть бути отримані за допомогою Zabbix API та використані в аналітичному модулі.

Додатковою перевагою є можливість налаштування історичних трендів, які дозволяють простежити зміну навантаження протягом доби, тижня чи місяця – це важливо для навчання моделей машинного аналізу.

Таким чином, рівень збору даних формує фундамент аналітичної системи, забезпечуючи її актуальною та достовірною інформацією про стан мережевого середовища.

2. Рівень аналітики (Data Analysis Layer)

Другий рівень архітектури відповідає за інтерпретацію даних, отриманих із системи моніторингу Zabbix, та за визначення можливих аномалій у поведінці мережевого трафіку. На цьому рівні формується логіка інтелектуального аналізу, яка дозволяє не лише фіксувати відхилення окремих параметрів, а й оцінювати загальний стан мережі.

Процес обробки та аналізу трафіку подано у вигляді логічної схеми на рис. 2.2.

У запропонованій архітектурі застосовується комбінований підхід, що поєднує два методи:

- rule-based аналіз, який здійснює визначення конкретних типів аномалій на основі порогових значень та логічних правил (наприклад, різке збільшення вхідного трафіку, підвищення ICMP-втрат, аномальна кількість TCP-з'єднань тощо);

- інтегральна оцінка ризику, що формується за допомогою простої нейронної моделі, яка агрегує значення кількох ключових параметрів у єдиний показник рівня ризику.

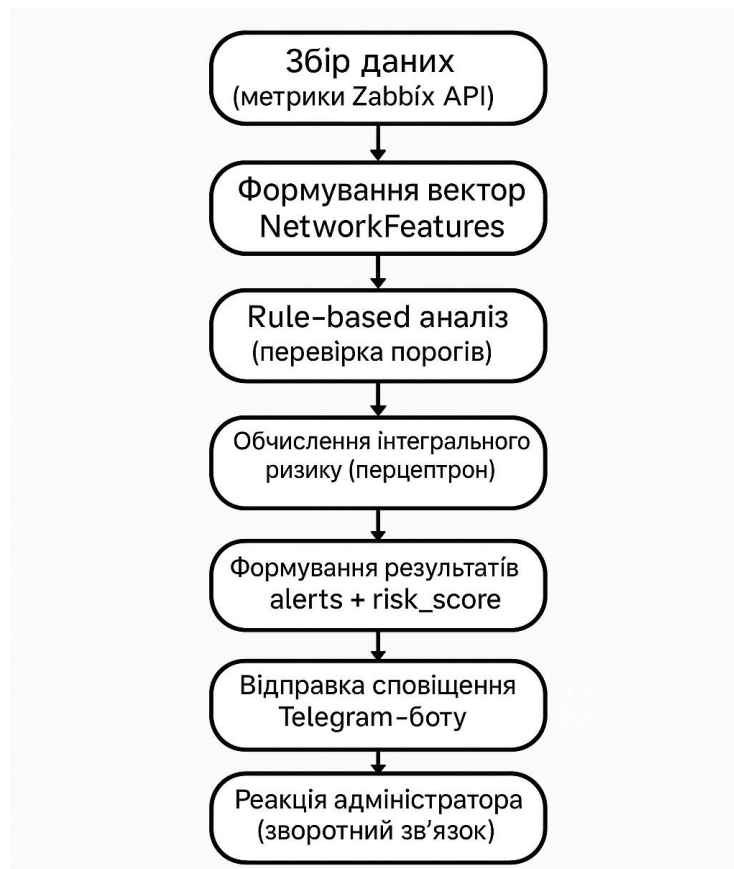


Рисунок 2.2 – Логічна схема процесу аналізу мережевого трафіку

У роботі для інтегральної оцінки використовується одношарова перцептронна модель з вагами, отриманими під час легкого навчання на попередньо сформованому наборі даних мережевих метрик. Це дозволяє моделі адаптивно враховувати статистичні властивості трафіку та підсилює rule-based аналіз, забезпечуючи узагальнену оцінку ризику навіть у ситуаціях, коли окремі показники не перевищують порогів.

Такий підхід дозволяє поєднати переваги обох методів: rule-based аналіз забезпечує точність і пояснюваність рішень, а нейронна складова дозволяє оцінити загальний стан системи й виявити потенційно небезпечні тенденції в поведінці мережі.

На рівні аналітики здійснюється:

- узагальнення отриманих від Zabbix значень основних мережеских показників;
- порівняння параметрів із визначеними порогами та формування попереджень про аномальну активність;
- обчислення інтегрального рівня ризику на основі вагової моделі перцептрона;
- класифікація стану системи на Normal, Warning або Critical.

Отримані результати передаються на наступний рівень архітектури для інформування адміністратора про можливі інциденти.

Запропонований підхід забезпечує розширюваність аналітичного модуля: у майбутньому до системи можуть бути додані нові правила, нові ознаки або складніші алгоритми аналізу, включно з моделями машинного навчання. Крім того, аналітичний рівень може бути інтегрований із зовнішніми системами візуалізації (наприклад, Grafana [36]) або платформами обробки великих обсягів даних.

3. Рівень сповіщення (Notification Layer)

Фінальний рівень архітектури відповідає за оперативне інформування адміністратора про виявлені відхилення в роботі мережі. На цьому рівні результати аналітичного модуля перетворюються на зрозумілі та структуровані повідомлення, які надсилаються адміністратору через Telegram-бот [37]. Такий спосіб взаємодії забезпечує швидку доставку повідомлень і дає можливість контролювати стан мережі навіть за відсутності доступу до панелі моніторингу.

Telegram-бот інтегрований із аналітичним модулем і отримує інформацію у випадках, коли система фіксує одну або кілька аномалій, або коли інтегральний рівень ризику переходить у стан Warning чи Critical. На основі цих даних формується коротке, але інформативне сповіщення, яке може містити:

- час виникнення події;
- тип і короткий опис виявленої аномалії;
- значення параметрів, що перевищили пороги;
- оцінку інтегрального ризику;

- рекомендацію щодо подальших дій (за потреби).

Таким чином, адміністратор отримує змогу реагувати на інциденти в реальному часі без необхідності постійного перегляду дашбордів або журналів подій. Це особливо важливо у випадках, коли йдеться про швидкі атаки на мережеву інфраструктуру, наприклад DDoS або SYN-flood, де своєчасність реагування суттєво впливає на наслідки.

Telegram-бот також може підтримувати базовий зворотний зв'язок із системою, наприклад надаючи можливість отримати поточну статистику або перелік останніх інцидентів за командою користувача. Це робить систему більш інтерактивною та зручною у використанні.

Взаємодія між аналітичним модулем і ботом здійснюється через стандартизовані HTTP-запити, що гарантує незалежність компонентів і спрощує можливість подальшої інтеграції з іншими каналами сповіщення, такими як електронна пошта, корпоративні месенджери чи системи керування подіями.

Рівень сповіщення значно підвищує ефективність архітектури, оскільки забезпечує:

- швидкість реагування – критичні інциденти не залишаються непоміченими;
- зручність взаємодії – адміністратор отримує інформацію у зрозумілому форматі;
- гнучкість інтеграції – можливість підключення нових каналів комунікації;
- безпеку – передача інформації здійснюється через авторизовані API-запити.

У підсумку запропонована архітектура забезпечує повний цикл аналізу мережевого трафіку – від збору до аналітичного опрацювання і сповіщення, створюючи основу для подальшої практичної реалізації системи.

2.3. Визначення параметрів і критеріїв виявлення аномалій

Вибір параметрів, за якими оцінюється стан мережі, є ключовим етапом побудови моделі інтелектуального аналізу трафіку. Від набору ознак залежить, які саме типи аномалій система здатна виявляти, наскільки рано вона може зафіксувати початок інциденту та як точно відокремлюватиме кібератаку від звичайних робочих піків навантаження. У даному дослідженні обрані параметри використовуються не лише для порогового rule-based аналізу, але й формують вектор ознак навчального набору даних для побудови простої моделі машинного навчання, що підсилює точність інтегральної оцінки ризику.

У даній роботі для побудови моделі обрано групу параметрів, які відображають різні рівні функціонування мережевої інфраструктури:

- показники завантаження хосту (CPU, кількість процесів, користувачів);
- показники інтенсивності трафіку (вхідний / вихідний обсяг);
- показники якості мережевого шляху (ICMP-втрати та затримка);
- показники стану транспортних з'єднань (кількість встановлених TCP-сесій).

Такий підхід дозволяє одночасно враховувати симптоми перевантаження ресурсу, аномальну поведінку протоколів та індикатори можливих атак, причому без необхідності глибокого аналізу вмісту пакетів (Deep Packet Inspection), що є ресурсоємною задачею [38].

Логічні взаємозв'язки між параметрами, порогами та механізмом прийняття рішення про аномалію наведено на схемі формування критеріїв виявлення аномалій (рис. 2.3).

1. Завантаження процесора (CPU) як індикатор ресурсних атак.

Параметр CPU utilization обрано як базовий показник стану обчислювального ресурсу хоста. У багатьох типах атак (DoS, DDoS, brute-force, скрипти-сканери) кінцевою метою є не тільки перевантаження каналу, але й виснаження ресурсів системи, що призводить до деградації сервісів або повної відмови в обслуговуванні. З теоретичної точки зору, зростання завантаження CPU може бути наслідком:

- обробки надмірної кількості мережевих сесій;
- генерації великої кількості контекстних перемикань між процесами;
- інтенсивної обробки вхідних пакетів ядром ОС;
- виконання шкідливого коду або скриптів.

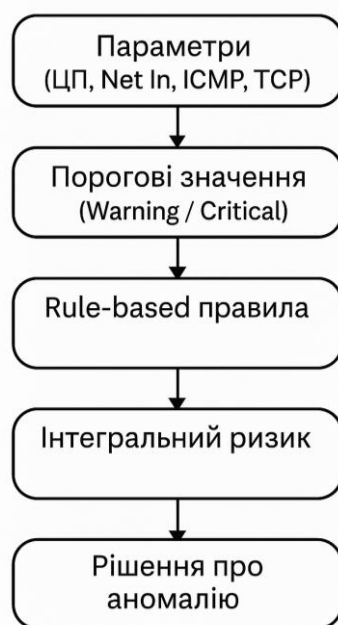


Рисунок 2.3 – Схема формування критеріїв виявлення аномалій

На відміну від суто мережевих метрик, CPU характеризує реакцію кінцевого вузла на зовнішнє навантаження. Навіть якщо канал ще не перевантажений, але процесор постійно працює у стані близько до граничного, це може бути ранньою ознакою атаки на рівні застосувань або протоколів [39].

Саме тому CPU використовується не лише як окремий тригер, але й як складова інтегральної оцінки ризику: поєднання підвищеного трафіку з високим завантаженням ЦП значно ймовірніше вказує на небажаний вплив, ніж один із цих факторів поодиноці.

2. Кількість процесів та користувачів як контекстні системні показники.

Параметри кількість активних процесів і кількість користувачів не є прямими показниками атаки, але виконують роль контекстних ознак. Вони дозволяють моделі:

- відрізняти «практично нормальне» навантаження (наприклад, багато користувачів у піковий час) від аномального (різко зросла кількість процесів без зростання кількості легітимних користувачів);

- виявляти підозрілі ситуації, коли одночасно спостерігається невелика кількість користувачів, нетипово висока кількість процесів і зростання навантаження CPU.

Теоретично це відповідає концепції кореляційного аналізу ознак: окремо взятий показник може бути нейтральним, але їхня сукупність може свідчити про аномалію (наприклад, масовий запуск процесів, ініційованих шкідливим ПЗ або скриптом-ботом).

3. Вхідний і вихідний трафік як основний індикатор flood/DDoS-активності.

Параметри вхідного (Net In) та вихідного (Net Out) трафіку є класичними показниками для виявлення flood- та DDoS-атак, які відносяться до категорії volume-based атак. Теоретично такі атаки спрямовані на:

- вичерпання пропускної здатності каналу;
- перевантаження мережевого інтерфейсу;
- створення черг у буферах маршрутизаторів.

Для таких сценаріїв характерні:

- різкі стрибки вхідного трафіку;
- неузгодженість між кількістю легітимних користувачів та обсягом трафіку;
- тривалі періоди «аномально високого» навантаження без очевидної бізнес-причини.

Використання порогів попередження та критичного значення дозволяє розділяти стани:

- «потенційно підозрілий сплеск» (можливий пік легітимного навантаження);
- «критичне перевищення», більш типове для вірусних активностей або DDoS.

Таким чином, параметри обсягу трафіку безпосередньо пов'язані з моделлю мережових нападів, описаною в літературі з кібербезпеки: більшість flood-атак мають яскраво виражену об'ємну складову [40].

4. ICMP-втрати та затримка як показники якості каналу та перевантаження.

Використання ICMP loss (відсоток втрат ICMP-пакетів) та ICMP latency (затримка відповіді) спирається на класичну концепцію QoS (Quality of Service) та теорію черг у мережових системах. Коли канал перевантажений:

- зростає час обробки пакетів у маршрутизаторах;
- збільшується ймовірність відкидання пакетів;
- по ICMP відображається втрата й затримка.

Теоретично, високі ICMP-втрати та збільшений RTT (round-trip time) характерні для:

- мережових заторів (congestion);
- ICMP flood атак;
- перевантаження проміжних вузлів.

На відміну від чисто трафікових метрик, ICMP-показники відображають якість маршруту та працездатність мережевої інфраструктури між хостом і цільовим вузлом. Це важливо, оскільки атака може бути спрямована не на кінцевий хост, а на проміжні вузли. Тому ICMP використовується як пасивний “барометр” стану мережі: навіть якщо обсяг трафіку ще не критичний, стабільне зростання затримки й втрат може свідчити про те, що система знаходиться близько до точки насичення [41].

5. Кількість встановлених TCP-з'єднань як індикатор протокольних атак.

Параметр кількість встановлених TCP-сесій (TCP established) обрано як ключову ознаку для виявлення атак, орієнтованих на виснаження стану (state exhaustion), зокрема:

- атак типу SYN flood, коли зломисник ініціює велику кількість напіввідкритих чи встановлених з'єднань;
- спроб утримувати ресурси сервера зайнятими якомога довше.

Теоретично TCP – це протокол зі збереженням стану: кожне з’єднання потребує:

- виділення запису в таблиці сесій;
- використання оперативної пам’яті;
- періодичного обслуговування.

Різке зростання кількості активних сесій при відсутності відповідного зростання кількості легітимних користувачів чи обсягів корисного трафіку є класичною ознакою протокольної атаки. Саме тому цей параметр широко використовується у багатьох системах IDS/IPS та фаєрволах як індикатор нетипової активності [42].

6. Інтегральний підхід до параметрів і критеріїв.

Окремо взятий параметр (наприклад, лише CPU або лише трафік) часто дає обмежену картину: нормальна життєдіяльність системи також включає піки навантаження, тимчасові стрибки трафіку або затримок. Тому у моделі використовується комбінований підхід, коли:

- кожний параметр має власні порогові критерії (локальні аномалії);
- сукупність параметрів використовується для формування інтегрального показника ризику.

Такий підхід узгоджується з загальними принципами інтелектуального аналізу даних: важливими є не лише окремі ознаки, а й їхнє поєднання. Наприклад:

- підвищений вхідний трафік при нормальних ICMP-показниках і низькому CPU може бути звичайним піком;
- підвищений трафік разом з високими ICMP-втратами та зростанням CPU набагато ймовірніше вказує на DDoS або flood-атаку.

Саме тому критерії виявлення аномалій включають:

- локальні пороги для окремих параметрів (попереджувальні та критичні значення);
- глобальний критерій у вигляді інтегральної оцінки стану (Normal / Warning / Critical).

У даній роботі такий інтегральний показник формується за допомогою простої навченої моделі – одношарового перцептрона, який обробляє вибрані параметри як єдиний вектор ознак і повертає числову оцінку рівня ризику аномальної активності. Це дозволяє фіксувати потенційні атаки навіть тоді, коли окремі метрики ще не вийшли за порогові межі.

У підсумку вибір саме цих параметрів є обґрунтованим, оскільки вони охоплюють різні рівні моделі OSI – від системних ресурсів до транспортного рівня та характеристик мережевого маршруту. Такий підхід забезпечує можливість виявляти як об’ємні атаки на кшталт flood чи DDoS, так і протокольні загрози, зокрема SYN flood, а також фіксувати непрямі симптоми перевантаження або деградації роботи системи. Важливо й те, що всі обрані метрики є стандартними для системи моніторингу Zabbix, що дає змогу отримувати їх без складного налаштування чи необхідності використання спеціалізованих даних або глибокого аналізу пакетів. Крім того, такі параметри добре інтерпретуються адміністратором, що робить модель практичною та ефективною у реальних умовах експлуатації.

Висновки до розділу 2:

У розділі було спроектовано модель інтелектуального аналізу мережевого трафіку, визначено технології та інструменти для її побудови, розроблено архітектуру системи та обґрунтовано вибір параметрів і критеріїв виявлення аномалій. Запропонована модель поєднує засоби моніторингу Zabbix, аналітичний модуль на Python та механізм сповіщення через Telegram-бота, забезпечуючи повний цикл – від збору та обробки мережевих даних до автоматичного реагування на потенційні загрози.

РОЗДІЛ 3. РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ МОДЕЛІ ІНТЕЛЕКТУАЛЬНОГО АНАЛІЗУ МЕРЕЖЕВОГО ТРАФІКУ

3.1. Налаштування та розгортання інфраструктури збору мережевих даних Zabbix

Для побудови моделі інтелектуального аналізу мережевого трафіку необхідно розгорнути надійну інфраструктуру збору даних. Першим етапом є встановлення та налаштування Zabbix Server, який забезпечує комплексний контроль за станом мережі та серверів у реальному часі.

Як основну платформу для розгортання Zabbix Server було обрано операційну систему Ubuntu Desktop [43]. Такий вибір зумовлений кількома ключовими факторами, що роблять Linux оптимальною платформою для розгортання серверів моніторингу:

1. Низьке споживання ресурсів Linux-системи потребують значно менше оперативної пам'яті та процесорного навантаження порівняно з Windows, що особливо важливо для серверних рішень або віртуалізованого середовища [44]. Завдяки цьому Zabbix працює стабільно навіть на машинах із мінімальними апаратними характеристиками.

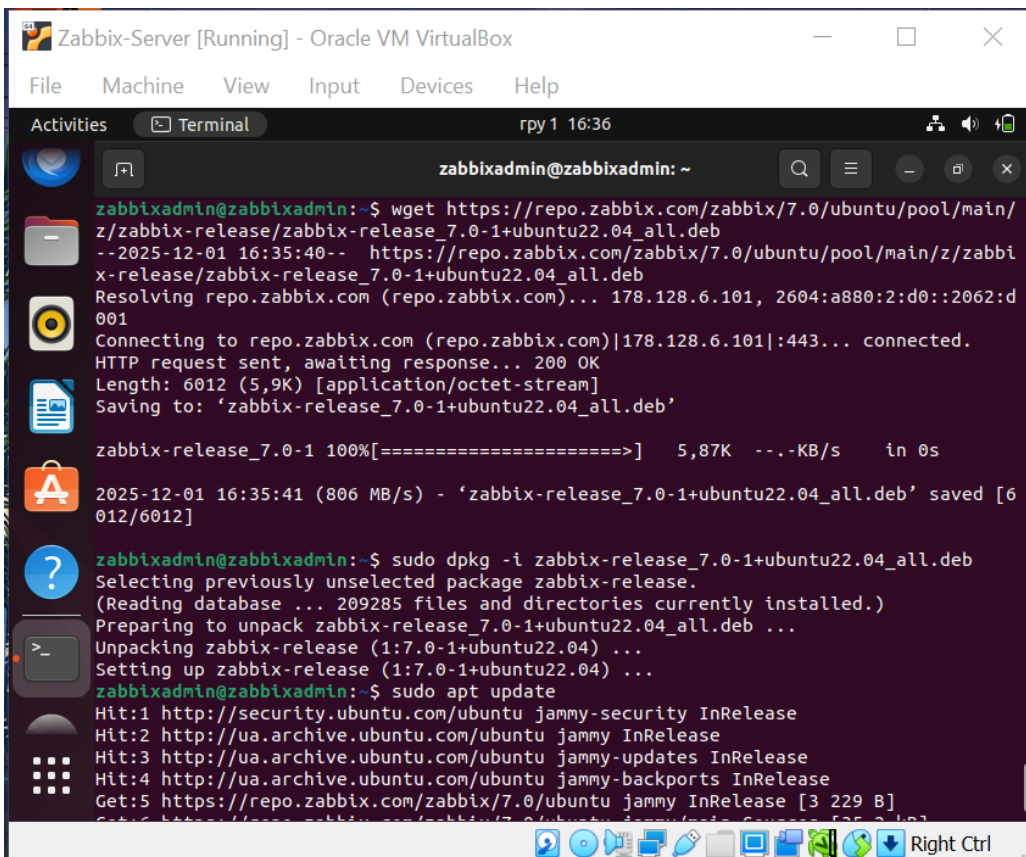
2. Зручність адміністрування. Linux пропонує гнучке керування системою через термінал, зручні інструменти автоматизації, пакетні менеджери, а також можливість швидко налаштовувати сервіси, мережеві параметри та служби. Це значно спрощує експерименти, тестування та розгортання моніторингової інфраструктури.

3. Підвищена безпека операційної системи. Linux традиційно вважається більш захищеною платформою завдяки відкритому вихідному коду, регулярним оновленням, активній спільноті та розвиненим механізмам контролю доступу [45]. У серверному середовищі це критично важливо, адже саме на сервери спрямована більшість атак.

4. Поширеність Linux у серверній інфраструктурі. Переважна більшість сучасних серверів працює саме під управлінням дистрибутивів Linux (Ubuntu

Server, Debian, CentOS, RHEL тощо). Це обумовлено стабільністю, масштабованістю, безкоштовністю ліцензій та розвинутою екосистемою для корпоративних рішень. Саме тому використання Linux у дослідницькому середовищі дає змогу наблизити експериментальні умови до реальних практик індустрії.

Для того щоб отримати актуальну версію системи моніторингу, на серверну машину було додано офіційний репозиторій Zabbix. На рисунку 3.1 показано процес завантаження пакета репозиторію за допомогою команди `wget`, після чого цей пакет було встановлено через інструмент `dpkg`. Додавання репозиторію дозволяє отримувати всі подальші оновлення та залежності безпосередньо з офіційних джерел, що забезпечує стабільність та безпечність роботи системи. Після інсталяції репозиторію виконується оновлення локального списку пакетів, що дозволяє системі побачити нові доступні версії компонентів Zabbix. Далі встановлюються пакунки серверної частини, веб-інтерфейсу та агента, які необхідні для повноцінної роботи системи моніторингу.

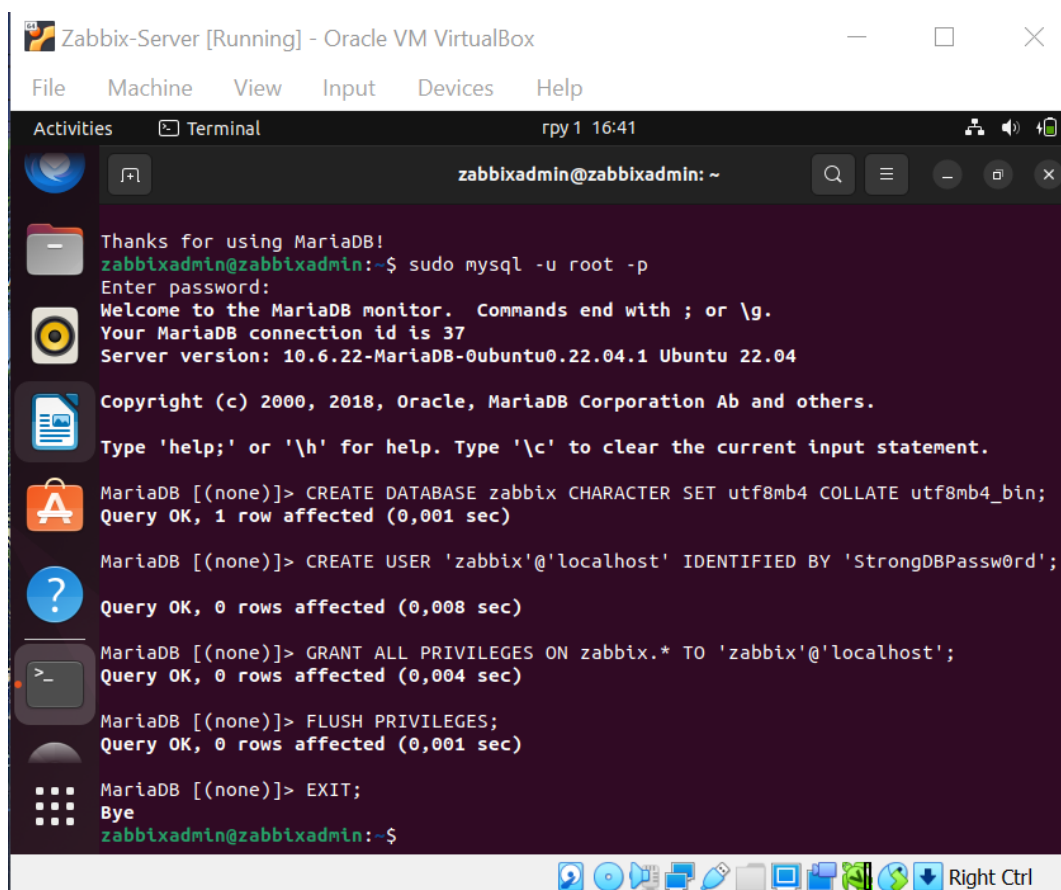


```
zabbixadmin@zabbixadmin: ~  
zabbixadmin@zabbixadmin:~$ wget https://repo.zabbix.com/zabbix/7.0/ubuntu/pool/main/z/zabbix-release/zabbix-release_7.0-1+ubuntu22.04_all.deb  
--2025-12-01 16:35:40-- https://repo.zabbix.com/zabbix/7.0/ubuntu/pool/main/z/zabbix-release/zabbix-release_7.0-1+ubuntu22.04_all.deb  
Resolving repo.zabbix.com (repo.zabbix.com)... 178.128.6.101, 2604:a880:2:d0::2062:d001  
Connecting to repo.zabbix.com (repo.zabbix.com)|178.128.6.101|:443... connected.  
HTTP request sent, awaiting response... 200 OK  
Length: 6012 (5,9K) [application/octet-stream]  
Saving to: 'zabbix-release_7.0-1+ubuntu22.04_all.deb'  
  
zabbix-release_7.0-1 100%[=====] 5,87K --.-KB/s in 0s  
  
2025-12-01 16:35:41 (806 MB/s) - 'zabbix-release_7.0-1+ubuntu22.04_all.deb' saved [6012/6012]  
  
zabbixadmin@zabbixadmin:~$ sudo dpkg -i zabbix-release_7.0-1+ubuntu22.04_all.deb  
Selecting previously unselected package zabbix-release.  
(Reading database ... 209285 files and directories currently installed.)  
Preparing to unpack zabbix-release_7.0-1+ubuntu22.04_all.deb ...  
Unpacking zabbix-release (1:7.0-1+ubuntu22.04) ...  
Setting up zabbix-release (1:7.0-1+ubuntu22.04) ...  
zabbixadmin@zabbixadmin:~$ sudo apt update  
Hit:1 http://security.ubuntu.com/ubuntu jammy-security InRelease  
Hit:2 http://ua.archive.ubuntu.com/ubuntu jammy InRelease  
Hit:3 http://ua.archive.ubuntu.com/ubuntu jammy-updates InRelease  
Hit:4 http://ua.archive.ubuntu.com/ubuntu jammy-backports InRelease  
Get:5 https://repo.zabbix.com/zabbix/7.0/ubuntu jammy InRelease [3 229 B]  
Get:6 https://repo.zabbix.com/zabbix/7.0/ubuntu jammy Release [329 B]  
Get:7 https://repo.zabbix.com/zabbix/7.0/ubuntu jammy/main amd64 Packages [329 B]  
Get:8 https://repo.zabbix.com/zabbix/7.0/ubuntu jammy/main i386 Packages [329 B]  
Fetched 1008 B in 0s (1008 B/s)  
Reading package lists... Done
```

Рисунок 3.1 – Завантаження та встановлення офіційного репозиторію Zabbix у системі Ubuntu

Zabbix використовує реляційну базу даних для зберігання всієї службової інформації: налаштувань хостів, історичних значень метрик, тригерів, логів подій тощо. У межах роботи як СУБД було обрано MariaDB, яка є оптимізованою, легкою та повністю сумісною з MySQL.

На рисунку 3.2 представлено підключення до MariaDB під адміністраторським обліковим записом та створення окремої бази даних zabbix. Окрім цього, створюється спеціальний користувач zabbix, якому призначаються всі необхідні привілеї для роботи з цією базою. Такий підхід відповідає принципам безпеки – кожен сервіс має працювати в межах окремого облікового запису з обмеженими правами.



```
Zabbix-Server [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities Terminal rpy 1 16:41
zabbixadmin@zabbixadmin: ~
Thanks for using MariaDB!
zabbixadmin@zabbixadmin:~$ sudo mysql -u root -p
Enter password:
Welcome to the MariaDB monitor. Commands end with ; or \g.
Your MariaDB connection id is 37
Server version: 10.6.22-MariaDB-0ubuntu0.22.04.1 Ubuntu 22.04

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> CREATE DATABASE zabbix CHARACTER SET utf8mb4 COLLATE utf8mb4_bin;
Query OK, 1 row affected (0,001 sec)

MariaDB [(none)]> CREATE USER 'zabbix'@'localhost' IDENTIFIED BY 'StrongDBPassw0rd';
Query OK, 0 rows affected (0,008 sec)

MariaDB [(none)]> GRANT ALL PRIVILEGES ON zabbix.* TO 'zabbix'@'localhost';
Query OK, 0 rows affected (0,004 sec)

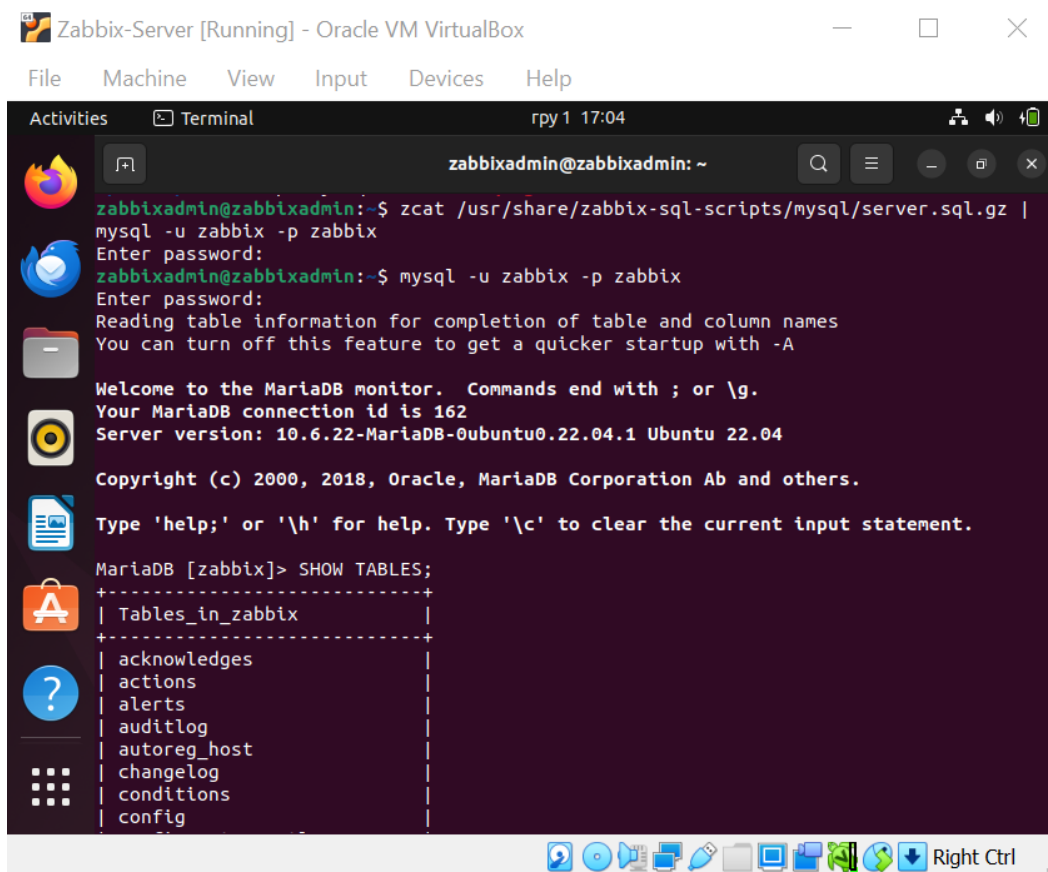
MariaDB [(none)]> FLUSH PRIVILEGES;
Query OK, 0 rows affected (0,001 sec)

MariaDB [(none)]> EXIT;
Bye
zabbixadmin@zabbixadmin:~$
```

Рисунок 3.2 – Створення бази даних MariaDB та призначення привілеїв користувачу Zabbix

Після створення бази на рисунку 3.3 демонструється імпорт початкової структури таблиць. Ця структура містить сотні таблиць Zabbix, включаючи hosts, items, history, events, alerts та інші, які необхідні для збереження даних системи

моніторингу. Імпорт здійснюється шляхом розпакування SQL-скрипту та передачі його у MariaDB. На завершення виконано контрольну перевірку – команда SHOW TABLES; підтверджує успішне створення всієї структури бази даних.



The screenshot shows a terminal window titled "Zabbix-Server [Running] - Oracle VM VirtualBox". The terminal is running a series of commands to import the Zabbix database structure into MariaDB. The commands and their outputs are as follows:

```
zabbixadmin@zabbixadmin:~$ zcat /usr/share/zabbix-sql-scripts/mysql/server.sql.gz |
mysql -u zabbix -p zabbix
Enter password:
zabbixadmin@zabbixadmin:~$ mysql -u zabbix -p zabbix
Enter password:
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 162
Server version: 10.6.22-MariaDB-0ubuntu0.22.04.1 Ubuntu 22.04

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [zabbix]> SHOW TABLES;
+-----+
| Tables_in_zabbix |
+-----+
| acknowledges     |
| actions           |
| alerts            |
| auditlog          |
| autoreg_host      |
| changelog         |
| conditions        |
| config            |
+-----+
```

Рисунок 3.3 – Імпорт початкової структури таблиць Zabbix у базу даних MariaDB

Після створення бази даних та встановлення всіх компонентів запускається служба zabbix-server. На рисунку 3.4 наведено приклад виведення команди `systemctl status zabbix-server`, яка демонструє, що сервер запущено успішно та працює стабільно. У виведенні можна побачити всі активні процеси, які відповідають за різні підсистеми Zabbix: обробку даних, синхронізацію конфігурацій, роботу алертів, механізми виявлення, обробку історії тощо.

Це підтверджує, що сервер зміг підключитися до бази даних, коректно зчитує конфігурації та готовий приймати дані від агентів.

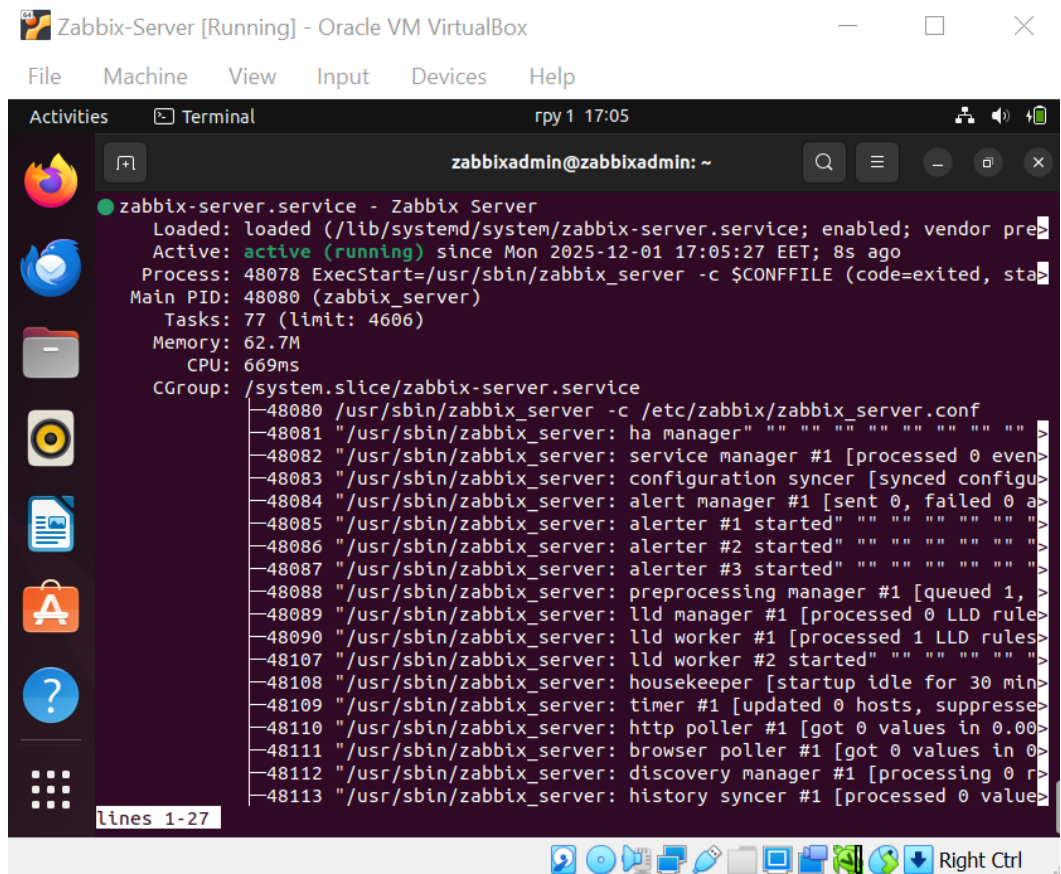


Рисунок 3.4 – Перевірка статусу служби Zabbix Server після запуску

Після успішного запуску серверної частини відбувається налаштування системи через веб-інтерфейс. На рисунку 3.5 показано етап конфігурації підключення до бази даних у браузері. Тут повторно зазначаються параметри доступу до MariaDB – ім'я бази, користувач, пароль та хост. Якщо всі параметри введено правильно, Zabbix здійснює перевірку підключення та переходить до наступного етапу інсталяції.

На завершальному екрані встановлення користувачу пропонується переглянути підсумок конфігурації та завершити інсталяцію.

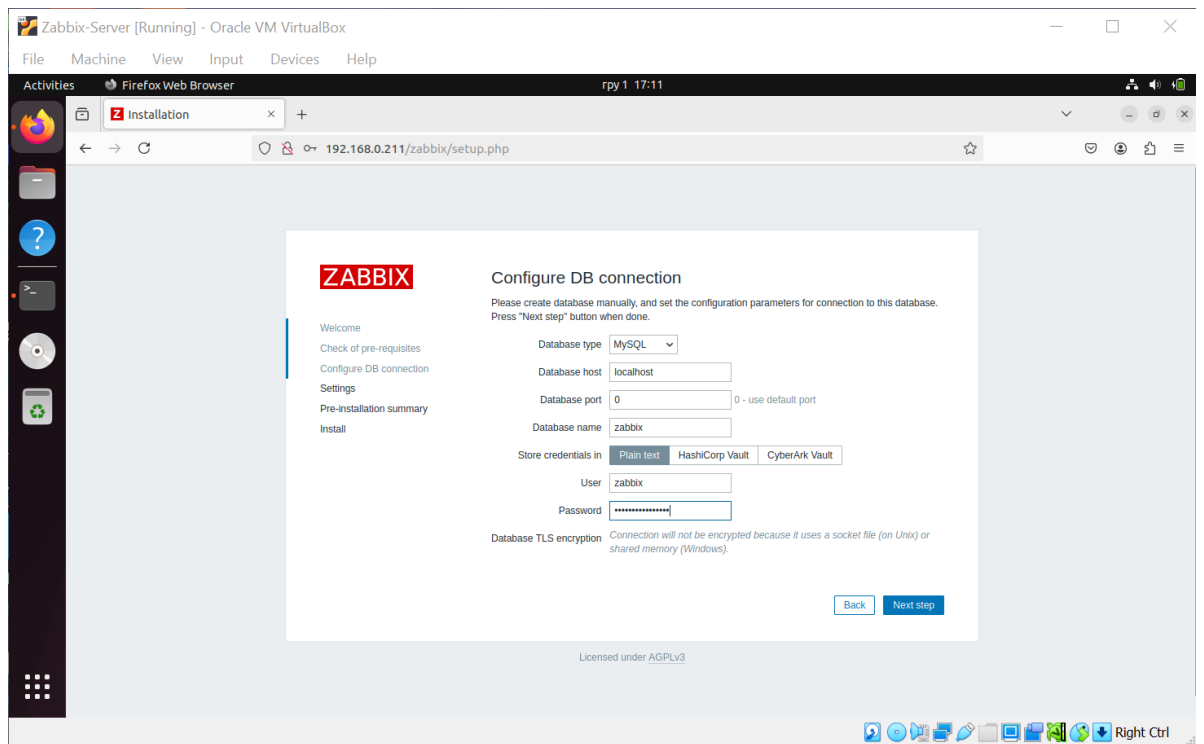


Рисунок 3.5 – Конфігурація підключення до бази даних через веб-інтерфейс Zabbix

Після завершення інсталяції та авторизації у веб-інтерфейсі відкривається головна панель керування Zabbix, зображена на рисунку 3.6. На ній відображаються системні параметри: статус сервера, версія компонентів, кількість доступних шаблонів, хостів, елементів даних і тригерів. Тут також можна побачити швидкість обробки даних Zabbix Server (Values per second) та поточні проблеми у мережі, якщо такі є.

Цей крок завершує налаштування серверної частини, і система повністю готова до приймання даних від агентів, а також до подальшої інтеграції з модулем інтелектуального аналізу мережевого трафіку.

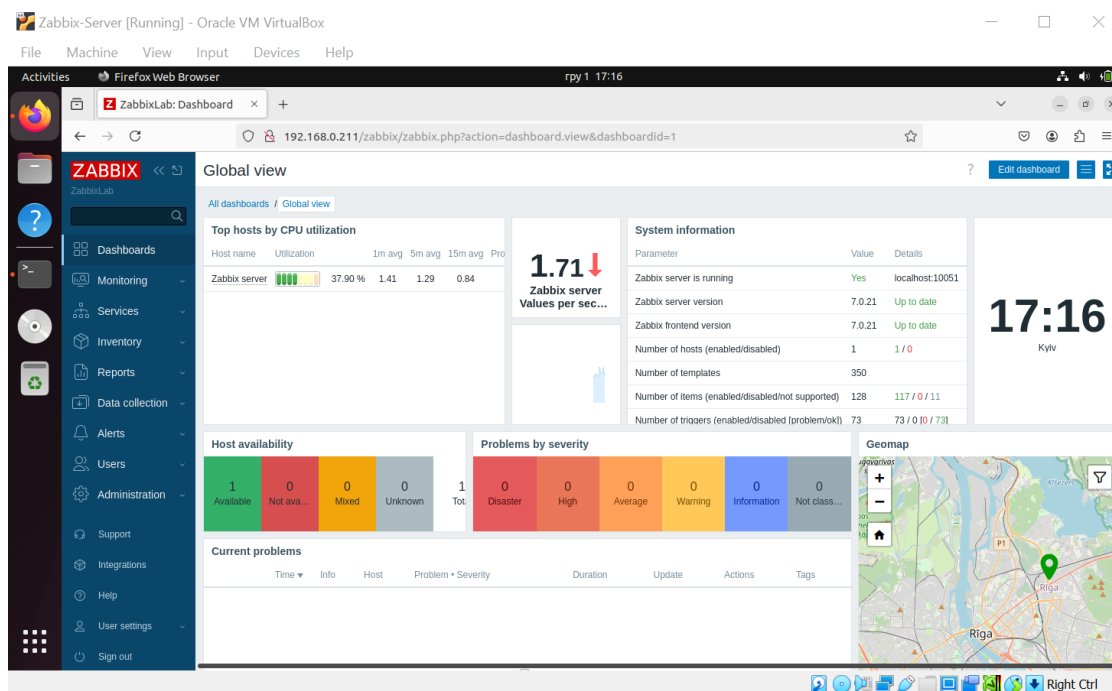


Рисунок 3.6 – Головна панель Zabbix після завершення встановлення та первинного налаштування

Для забезпечення збору телеметричних даних з клієнтських вузлів було розгорнуто Zabbix Agent 2 на окремій віртуальній машині з операційною системою Ubuntu [46]. Агент відповідає за отримання системних метрик, моніторинг мережевих показників, а також виконання активних та пасивних перевірок. Саме ці дані надалі формують основу для інтелектуального аналізу мережевого трафіку.

У першому кроці було здійснено встановлення пакета `zabbix-agent2` за допомогою стандартного менеджера пакетів `apt`. Команда `sudo apt install zabbix-agent2` -у автоматично завантажує та встановлює всі залежності, необхідні для роботи агента. Процес інсталяції показано на рисунку 3.7.

У результаті на систему додаються модулі, які забезпечують розширену функціональність агента, включаючи підтримку плагінів, розширену обробку метрик та сучасні механізми збору даних.

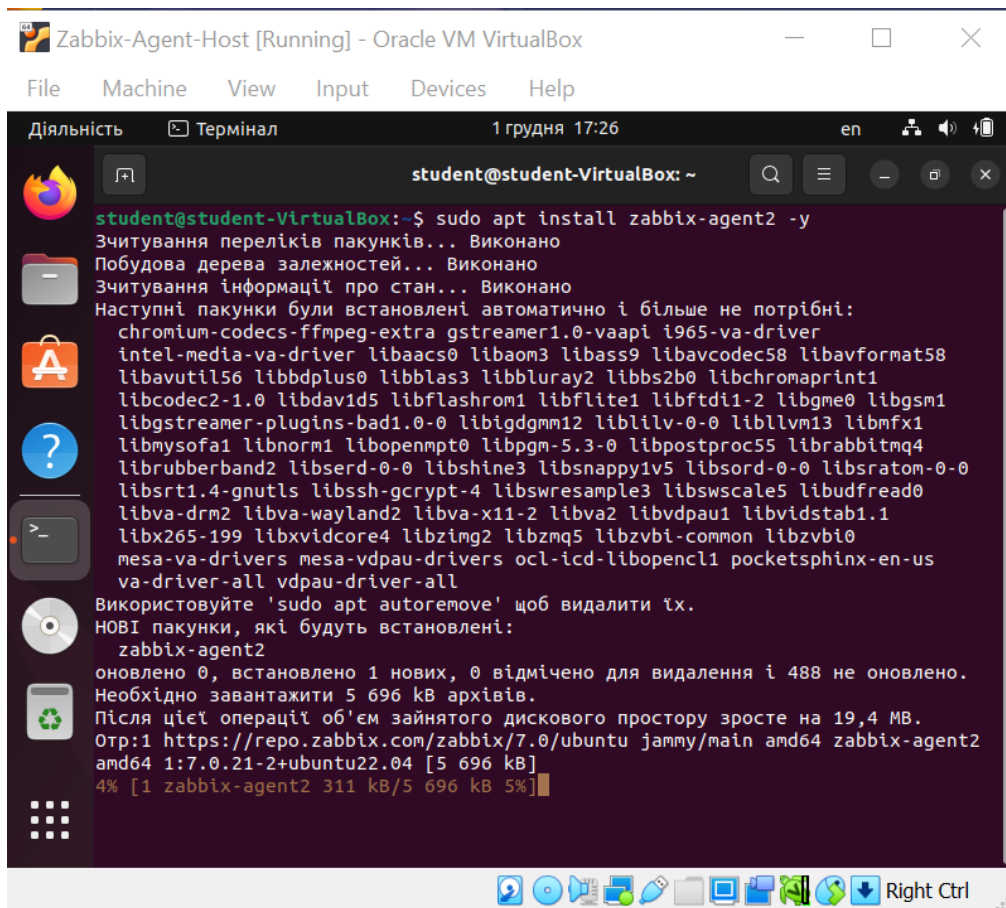


Рисунок 3.7 – Завантаження та встановлення пакета Zabbix Agent 2 у системі Ubuntu

Для встановлення стабільного та двостороннього з'єднання між агентом і сервером було відредаговано конфігураційний файл `/etc/zabbix/zabbix_agent2.conf`. На цьому етапі визначаються ключові параметри, які забезпечують можливість збору метрик у двох режимах – пасивному та активному.

Основні параметри:

- `Server = 192.168.0.211`. Вказує IP-адресу сервера Zabbix, який матиме право виконувати пасивні перевірки, тобто опитувати агент щодо системних метрик.
- `ServerActive = 192.168.0.211`. Визначає адресу сервера для активних перевірок, коли сам агент ініціює з'єднання та надсилає дані на сервер.

- Hostname = ZabbixAgentHost. Є унікальним ідентифікатором клієнтської машини. Це ім'я повинно повністю відповідати назві хоста, створеного у веб-інтерфейсі Zabbix, інакше агент не буде прив'язаний до відповідного запису.

На рисунку 3.8 показано внесення всіх параметрів. Правильне встановлення цих значень є критично важливим для коректного отримання сервером як активних, так і пасивних метрик від клієнтської машини.

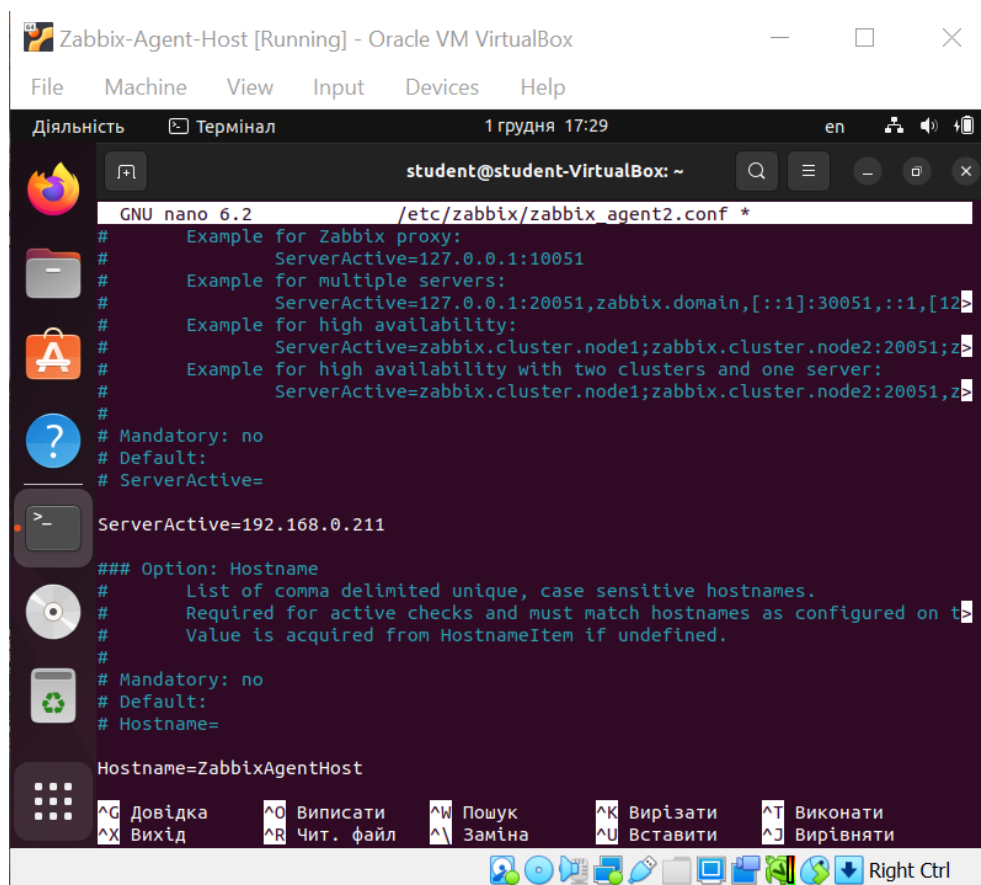


Рисунок 3.8 – Налаштування параметрів у файлі zabbix_agent2.conf

Для того щоб агент почав роботу, його службу необхідно увімкнути у системі. Використання команд, продемонстрованих на рис. 3.9 дозволяє перезапустити агент після внесення змін, додати його до автозапуску та перевірити актуальний стан служби. Після виконання цих команд ми бачимо, що служба перебуває в активному режимі (active (running)), що свідчить про успішне підключення та готовність агента до передачі даних.

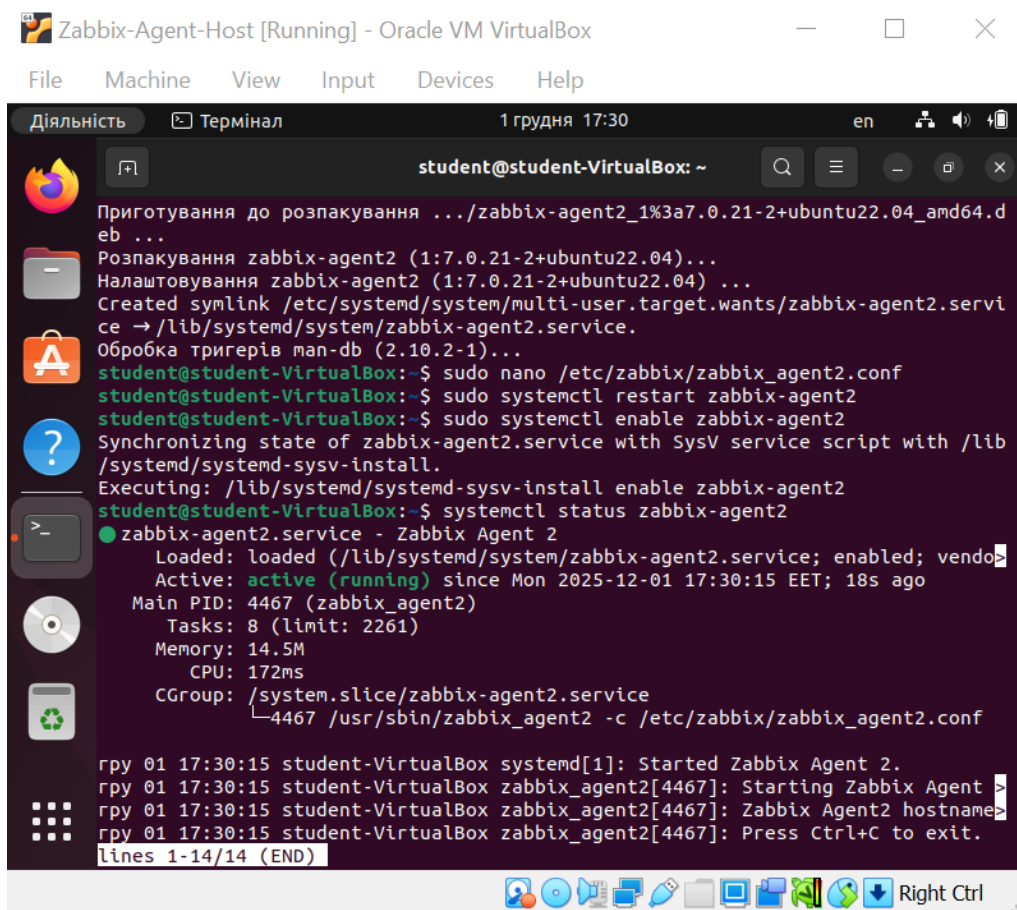


Рисунок 3.9 – Перевірка статусу служби Zabbix Agent 2 після її активації

Після запуску агента здійснено конфігурацію нового хоста у веб-інтерфейсі Zabbix. На рисунку 3.10 показано створення запису для клієнтської машини з такими параметрами:

- Host name: ZabbixAgentHost
- Template: Linux by Zabbix Agent – стандартний шаблон для збору системних метрик
- Agent interface: 192.168.0.168, порт 10050

Вказані значення дозволяють серверу встановити зв'язок із вузлом та автоматично розпочати опитування доступних елементів даних.

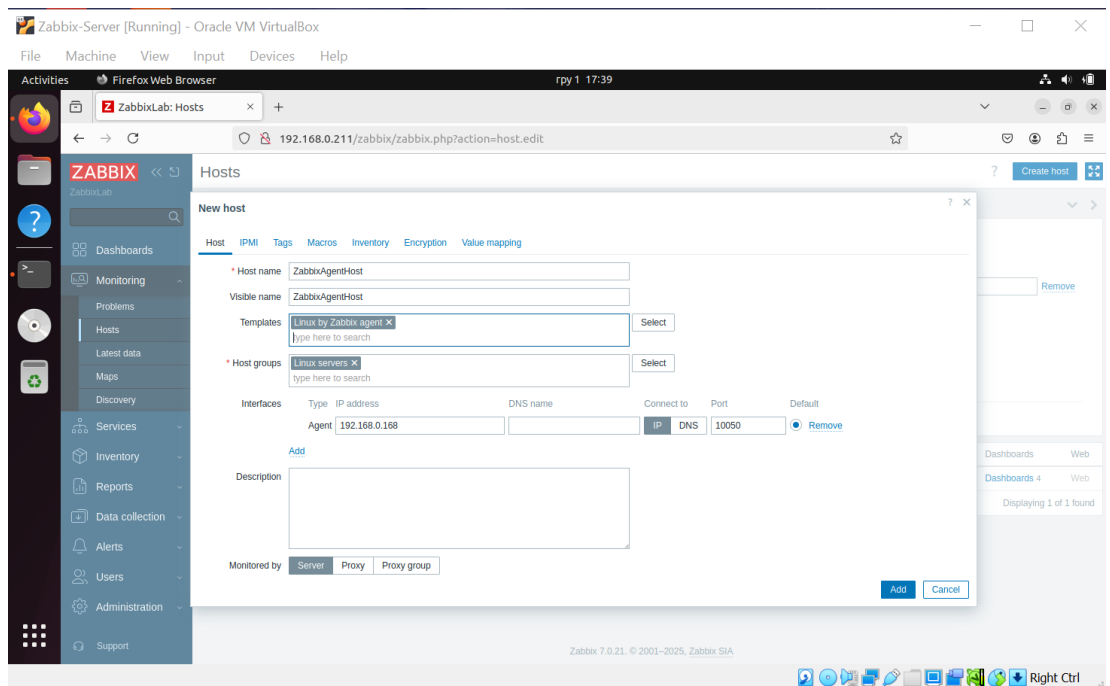


Рисунок 3.10 – Додавання клієнтського хоста ZabbixAgentHost у веб-інтерфейсі Zabbix

Щойно хост додається до системи, Zabbix починає збір інформації про стан клієнтського пристрою. На рисунку 3.11 демонструється панель Host dashboard, де у режимі реального часу відображаються файлові системи, мережеві інтерфейси та параметри продуктивності.

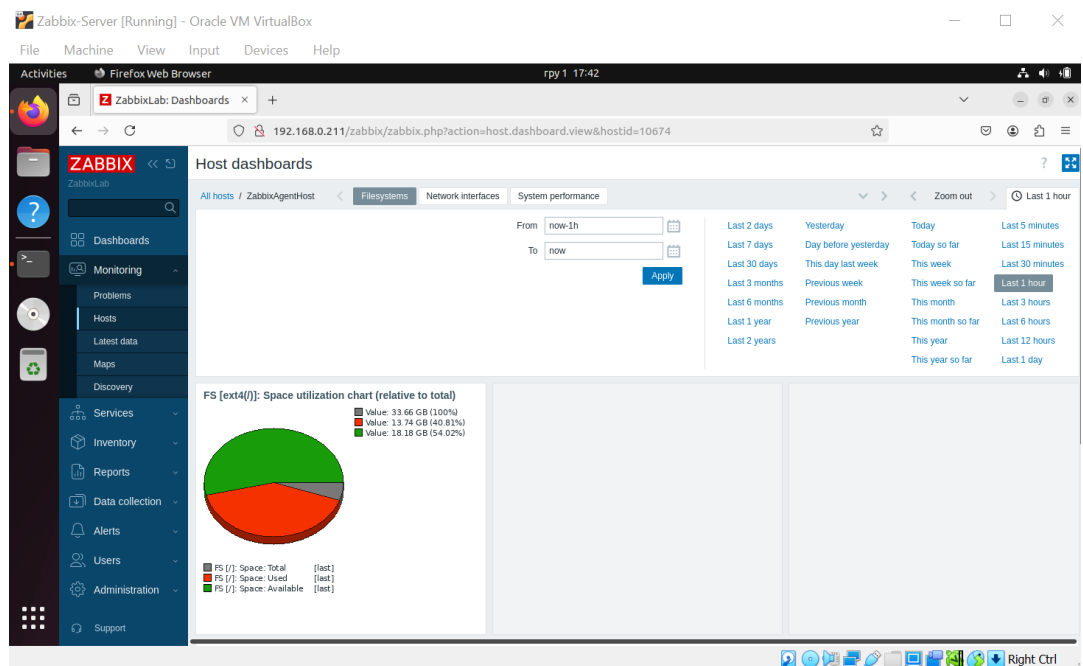


Рисунок 3.11 – Перегляд початкових системних метрик хоста у Host dashboard

Для перевірки роботи агента було змодельовано навантаження (рис. 3.12) на клієнтську машину (відкриття відео у браузері). У результаті цього експерименту зміни у використанні ресурсів – зокрема CPU та пам'яті – відобразилися на графіках у Zabbix. Таке тестування підтверджує коректність передачі даних і дозволяє надалі використовувати агент як джерело метрик для навчання інтелектуальної моделі.

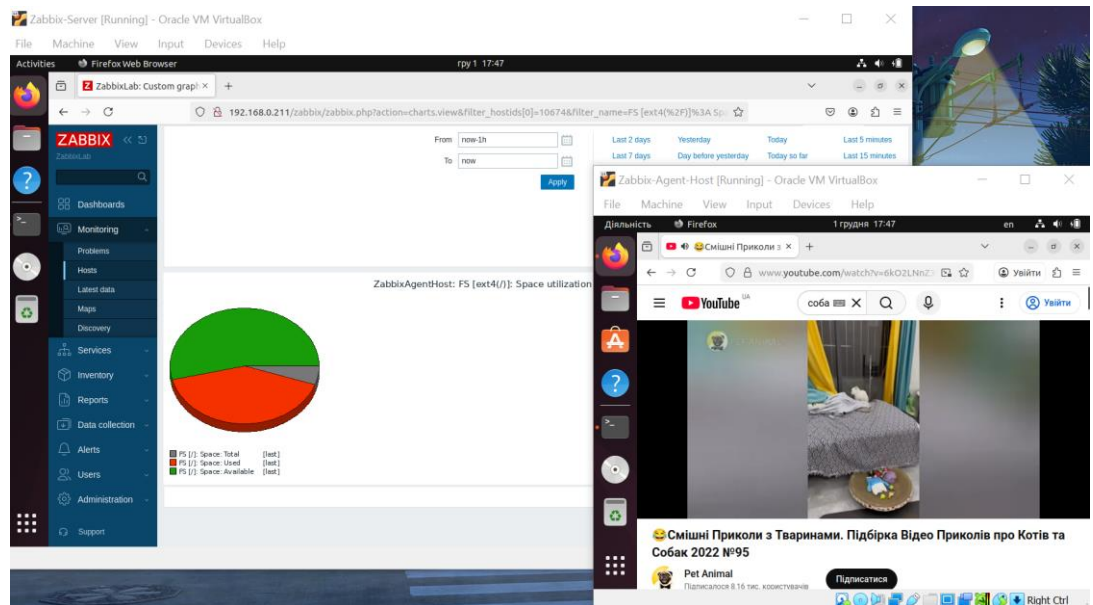


Рисунок 3.12 – Тестування роботи впливу навантаження на клієнтську машину

Зміни у графіках Zabbix після навантаження на клієнтську машину показано на рисунку 3.13. На панелі також видно, що моніторинг здійснюється з основного фізичного хосту, а не з віртуального середовища. Така конфігурація стала можливою завдяки використанню режиму Bridge Adapter, що забезпечує повноцінну присутність усіх віртуальних машин у локальній мережі та їхню взаємодію між собою і з основною системою [47].

Завдяки цьому сервер Zabbix, розгорнутий у віртуальному середовищі, може отримувати дані як від віртуальних агентів, так і виконувати API-запити з основного хосту. Це спрощує побудову подальшого модуля автоматизованого аналізу, оскільки API Zabbix залишатиметься доступним із фізичної машини для формування зовнішніх програмних компонентів, скриптів або моделей машинного навчання. Такий підхід дозволяє інтегрувати моніторингові дані у

Python-модуль інтелектуального аналізу трафіку без додаткових мережеских налаштувань.

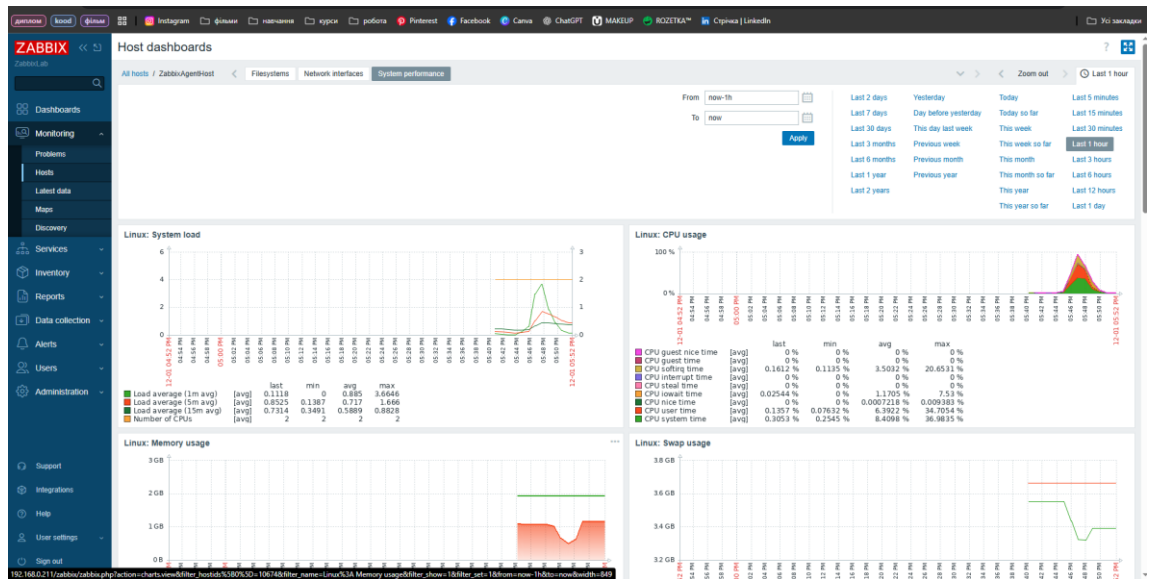


Рисунок 3.13 – Зміни у графіках Zabbix після навантаження на клієнтську машину

Для створення реалістичних умов роботи системи моніторингу було розгорнуто окреме середовище на базі Kali Linux – спеціалізованого дистрибутива, призначеного для тестування безпеки та моделювання кібератак [48]. На рисунку 3.14 продемонстровано успішне налаштування мережевого інтерфейсу Kali та отримання IP-адреси у спільному мережевому сегменті.

```
KaliLinux [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help

kaliattacker@kali: ~
$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host noprefixroute
        valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 08:00:27:65:b3:dc brd ff:ff:ff:ff:ff:ff
    inet 192.168.0.190/24 brd 192.168.0.255 scope global dynamic noprefixroute eth0
        valid_lft 7196sec preferred_lft 6256sec
    inet6 fe80::8280:cfc1:5625:1e1d/64 scope link
        valid_lft forever preferred_lft forever

kaliattacker@kali: ~
$
```

Рисунок 3.14 – Отримання мережеских налаштувань у Kali Linux після встановлення дистрибутива

Використання Kali Linux у даному дослідженні є обґрунтованим з кількох причин:

1. Наявність понад 600 вбудованих інструментів для тестування безпеки, включаючи генератори трафіку, утиліти для DoS-атак, мережеві сканери, аналізатори протоколів та інструменти для аудиту систем.

2. Повна сумісність із Linux-інфраструктурою, що дозволяє коректно взаємодіяти із Zabbix-сервером у режимі Bridge Adapter та генерувати трафік у тій самій локальній мережі.

3. Можливість створювати контрольовані сценарії атак, які не зашкодять реальним системам, але дадуть змогу відстежити реакцію Zabbix і зібрати необхідні дані для подальшого інтелектуального аналізу.

4. Широке використання у професійній кібербезпеці, що робить проведені експерименти максимально наближеними до практичних умов.

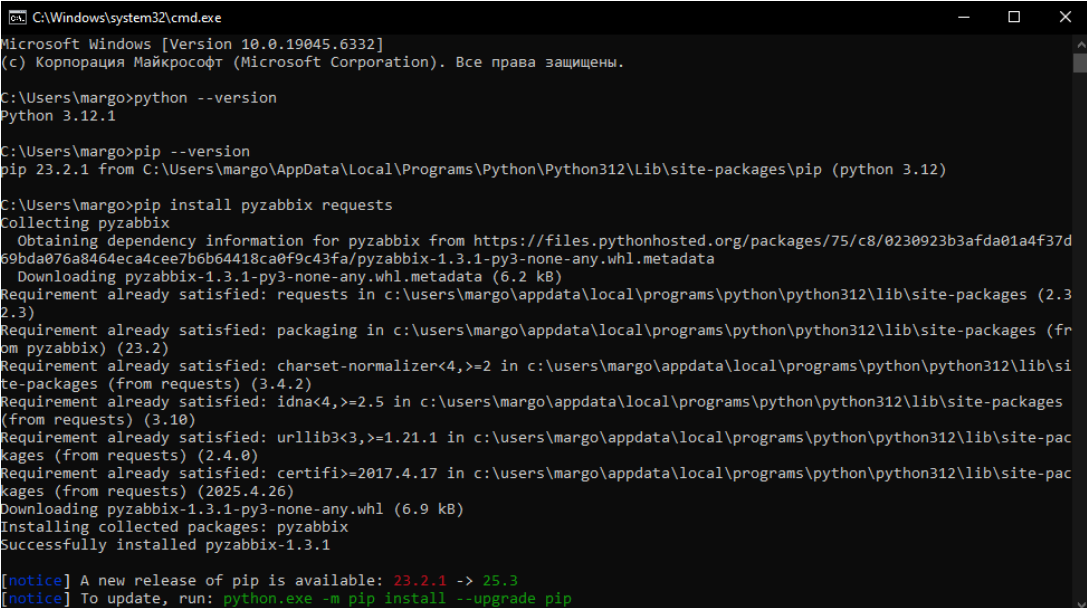
Завдяки цим характеристикам Kali Linux є оптимальним середовищем для моделювання навантажень на мережу та перевірки можливостей системи моніторингу.

3.2. Реалізація аналітичної підсистеми моделі інтелектуального аналізу

Аналітична підсистема моделі інтелектуального аналізу реалізована як набір Python-модулів, які спільно забезпечують повний цикл роботи системи від збору даних до інтелектуального моніторингу й сповіщення адміністратора. До складу програмної частини входять три основні скрипти.

- `collect_dataset.py` – модуль збору мережевих метрик із Zabbix і формування навчального датасету `dataset.csv`;
- `train_model.py` – модуль навчання простої лінійної моделі (перцептроноподібної логістичної регресії) на основі зібраного датасету;
- `zabbix_monitorAI.py` – модуль онлайн-моніторингу, який завантажує ваги з файлів, виконує rule-based аналіз, обчислює інтегральний ризик і надсилає Telegram-сповіщення.

Першим етапом є підготовка програмного середовища та доступу до Zabbix API. Аналітичний модуль запускається на основному хості з ОС Windows, де було встановлено Python 3.12 та необхідні бібліотеки. На рисунку 3.15 показано перевірку версій Python/pip та встановлення пакета pyzabbix, який забезпечує високо рівневу роботу з Zabbix API, а також бібліотеки requests для виконання HTTP-запитів до Telegram Bot API. Окрім того, для етапу навчання використовуються бібліотеки pandas, numpy та scikit-learn, які застосовуються у скрипті train_model.py для обробки датасету та побудови моделі.



```
C:\Windows\system32\cmd.exe
Microsoft Windows [Version 10.0.19045.6332]
(c) Корпорация Майкрософт (Microsoft Corporation). Все права защищены.

C:\Users\margo>python --version
Python 3.12.1

C:\Users\margo>pip --version
pip 23.2.1 from C:\Users\margo\AppData\Local\Programs\Python\Python312\Lib\site-packages\pip (python 3.12)

C:\Users\margo>pip install pyzabbix requests
Collecting pyzabbix
  Obtaining dependency information for pyzabbix from https://files.pythonhosted.org/packages/75/c8/0230923b3afda01a4f37d69bda076a8464eca4cee7b6b64418ca0f9c43fa/pyzabbix-1.3.1-py3-none-any.whl.metadata
  Downloading pyzabbix-1.3.1-py3-none-any.whl.metadata (6.2 kB)
Requirement already satisfied: requests in c:\users\margo\appdata\local\programs\python\python312\lib\site-packages (2.32.3)
Requirement already satisfied: packaging in c:\users\margo\appdata\local\programs\python\python312\lib\site-packages (from pyzabbix) (23.2)
Requirement already satisfied: charset-normalizer<4,>=2 in c:\users\margo\appdata\local\programs\python\python312\lib\site-packages (from requests) (3.4.2)
Requirement already satisfied: idna<4,>=2.5 in c:\users\margo\appdata\local\programs\python\python312\lib\site-packages (from requests) (3.10)
Requirement already satisfied: urllib3<3,>=1.21.1 in c:\users\margo\appdata\local\programs\python\python312\lib\site-packages (from requests) (2.4.0)
Requirement already satisfied: certifi>=2017.4.17 in c:\users\margo\appdata\local\programs\python\python312\lib\site-packages (from requests) (2025.4.26)
Downloading pyzabbix-1.3.1-py3-none-any.whl (6.9 kB)
Installing collected packages: pyzabbix
Successfully installed pyzabbix-1.3.1

[notice] A new release of pip is available: 23.2.1 -> 25.3
[notice] To update, run: python.exe -m pip install --upgrade pip
```

Рисунок 3.15 – Встановлення бібліотек pyzabbix та requests у середовищі Python на основному хості

Для встановлення з'єднання зі Zabbix у коді визначаються такі параметри:

- адреса веб-інтерфейсу Zabbix (ZABBIX_URL);
- облікові дані адміністратора (ZABBIX_USER, ZABBIX_PASSWORD);
- ідентифікатор хоста HOST_ID, який відповідає машині

ZabbixAgentHost.

Підключення виконується наступними командами:

```
# Підключення до Zabbix
zapi = ZabbixAPI(ZABBIX_URL)
zapi.login(ZABBIX_USER, ZABBIX_PASSWORD)
print("Підключено до Zabbix API. Версія:", zapi.api_version())
```

Цей фрагмент підтверджує коректність автентифікації та дозволяє здійснювати подальші запити до сервера моніторингу.

Окремо в коді оголошено ключі елементів даних (item key), що відповідають зібраним у Zabbix метрикам: CPU, вхідний/вихідний трафік, ICMP-втрати, ICMP-затримка, кількість встановлених TCP-з'єднань тощо.

На рисунку 3.16 наведено приклад роботи з інтерфейсом Zabbix, де можна переглянути та отримати ключ `net.if.in["enp0s3"]`, який використовується для зчитування вхідного трафіку мережевого інтерфейсу.

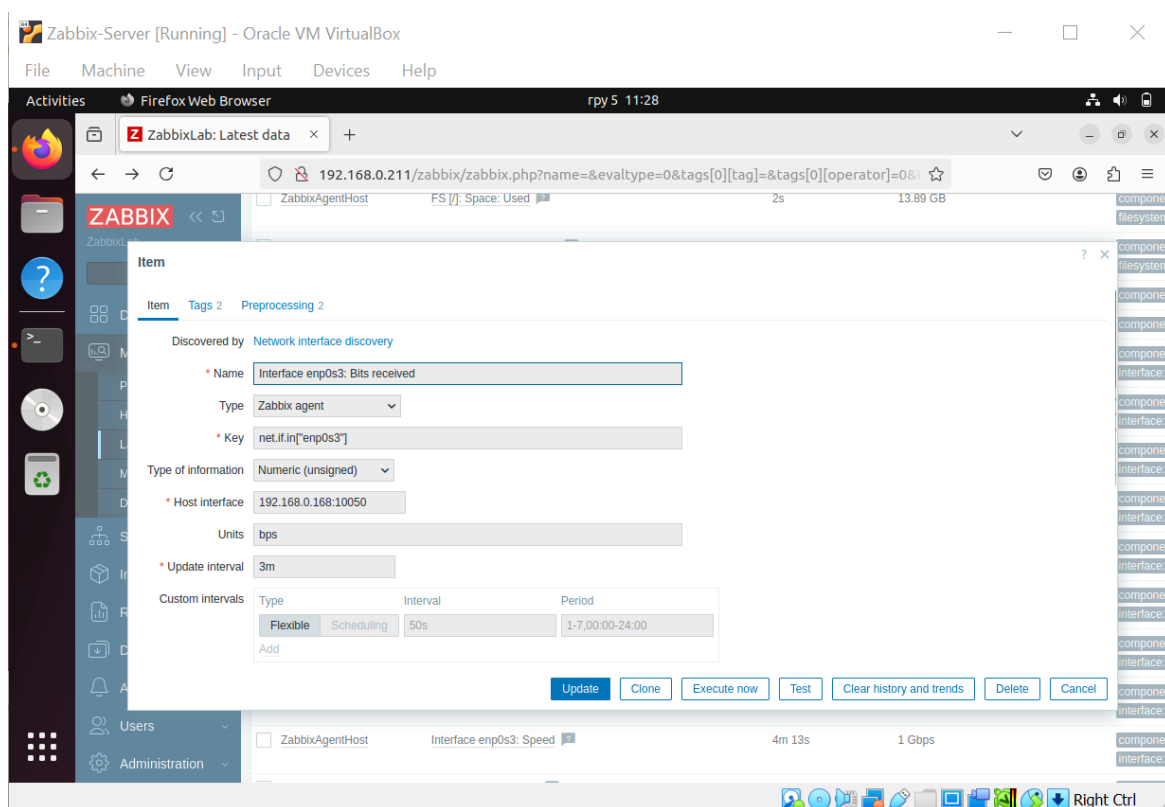


Рисунок 3.16 – Перегляд ключа елемента даних для мережевого інтерфейсу у Zabbix

На рис. 3.17 показано конфігурацію item типу Simple check для обчислення втрат ICMP-пакетів – важливий параметр для виявлення ICMP flood.

Рисунок 3.17 – Створення елемента даних типу Simple check для параметра ICMP packet loss у Zabbix

Зчитування актуальних значень item виконується функцією `get_last_value()`, яка звертається до методу `zapi.item.get()`:

```
# ----- Отримання з Zabbix останнього значення item за key_. -----
items = zapi.item.get(
    hostids=HOST_ID,
    search={"key_": key},
    output=["itemid", "name", "lastvalue", "units"],
    sortfield="itemid",
    limit=1,
)
```

Функція повертає числове значення метрики та її одиниці вимірювання, або ж повідомляє, що дані відсутні. Це дозволяє забезпечити стабільність джерела інформації для аналітики.

Для етапу формування датасету у модулі `collect_dataset.py` зчитані метрики доповнюються часовою міткою та автоматичною міткою класу `label`, яка формується за простим rule-based правилом. Кожен рядок зберігається у файл `dataset.csv`, що надалі використовується скриптом `train_model.py` для навчання моделі.

Зібрані метрики в модулі онлайн-моніторингу об'єднуються у структуру `NetworkFeatures`, реалізовану через `@dataclass`:

```
# ---- Формуємо вектор ознак (вхід для моделі) ----
features = NetworkFeatures(
    cpu=cpu,
    net_in=net_in,
    net_out=net_out,
    icmp_loss=icmp_loss,
    icmp_lat=icmp_lat,
    tcp_est=tcp_est,
    users=users,
    processes=procs,
)
```

Завдяки такому підходу формується вектор ознак, який є вхідними даними для інтелектуальної моделі.

Аналітична частина алгоритму реалізована у класі `RuleBasedAnomalyModel`, який складається з двох взаємодоповнюючих компонентів:

1. Rule-based виявлення аномалій. На основі встановлених порогів (CPU, трафік, ICMP-затримка, втрати, TCP-з'єднання) виконуються логічні перевірки.

Наприклад:

```
# 1) Підозра на DDoS / Flood за вхідним трафіком
if x.net_in > self.net_in_crit:
    alerts.append(
        f"Підозра на DDoS/Flood: вхідний трафік {x.net_in:.0f}
{net_in_units or 'bps'} "
        f"(поріг {self.net_in_crit} bps)."
```

Подібні правила дозволяють виявити:

- різкі піки вхідного трафіку (DDoS/Flood);
- підвищені втрати ICMP-пакетів (ICMP flood);
- зростання мережевої затримки;
- аномально високий рівень встановлених TCP-з'єднань (SYN flood);
- перевищення порогу завантаження CPU.

Rule-based частина фіксує конкретні патерни атак, забезпечуючи інтерпретованість результатів.

2. Простий перцептрон – оцінка інтегрального ризику. Друга складова моделі виконує узагальнення усіх ознак у єдиний показник – `risk_score`. У даній реалізації ваги перцептрона не задаються вручну, а завантажуються з файлів `weights.txt` та `bias.txt`, які формуються окремим навчальним модулем `train_model.py` на основі датасету `dataset.csv`.

Перцептрон складається з лінійної комбінації нормалізованих ознак та сигмоїдної активаційної функції:

```
# Лінійна комбінація нормалізованих ознак (вхід перцептрона)
risk_raw = (
    self.w_cpu      * cpu +
    self.w_net_in   * net_in +
    self.w_net_out  * net_out +
    self.w_icmp_loss * icmp_loss +
    self.w_icmp_lat  * icmp_lat +
    self.w_tcp_est  * tcp_est +
    self.bias
)

# Пропускаємо через сигмоїду, щоб отримати значення в [0..1]
RISK_SCALE = 8.0 # коефіцієнт "приглушення"
risk = self._sigmoid(risk_raw / RISK_SCALE)
return risk
```

Отримане значення ризику належить до діапазону `[0; 1]` та інтерпретується як:

- Normal — немає загроз;
- Warning — потенційні відхилення;
- Critical — висока ймовірність атаки або стресу мережі.

Модель повертає:

- список rule-based попереджень;
- числовий рівень ризику;
- текстову категорію ризику.

Таким чином, rule-based частина відповідає за виявлення конкретних патернів атак, а перцептронна модель, навчені ваги якої отримані у `train_model.py`,

забезпечує інтегральну оцінку стану трафіку з урахуванням статистичних властивостей даних.

Для оперативного інформування адміністратора модуль підключено до Telegram Bot API.

Створений бот ZabbixAI Alert (рис. 3.18) має унікальний токен доступу.



Рисунок 3.18 – Реєстрація Telegram-бота ZabbixAI Alert через BotFather та отримання токена доступу

Щоб отримати chat_id, використовується запит до методу getUpdates (рис. 3.19).

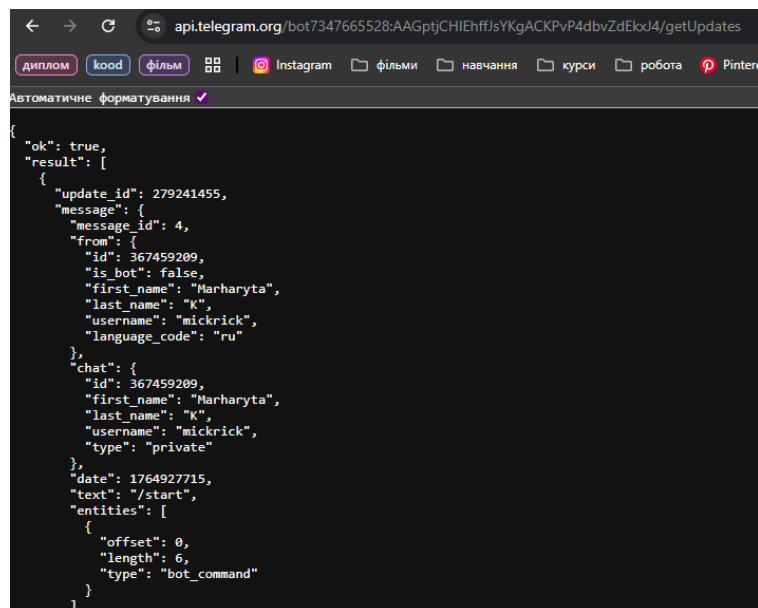


Рисунок 3.19 – Перевірка роботи Telegram Bot API методом getUpdates та отримання chat_id користувача

Надсилання повідомлень здійснюється функцією:

```
# ----- Повідомлення в TELEGRAM -----
def send_telegram(message: str) -> None:
    url = f"https://api.telegram.org/bot{TELEGRAM_TOKEN}/sendMessage"
    data = {"chat_id": CHAT_ID, "text": message}
    try:
        r = requests.post(url, data=data, timeout=10)
        if r.status_code != 200:
            print(f"Помилка Telegram API: {r.status_code} {r.text}")
    except Exception as e:
        print(f"Помилка відправки в Telegram: {e}")
```

У реальному середовищі токен бота має зберігатися у захищеній формі, наприклад у змінних середовища.

Основна функція `main()` виконує безперервний цикл моніторингу:

```
while True:
    # отримання метрик
    # формування вектора ознак
    # інтелектуальний аналіз (rule-based + перцептрон з навченими вагами)
    # надсилання Telegram-сповіщень
    time.sleep(25)
```

У кожній ітерації:

- зчитуються всі ключові метрики (CPU, трафік, ICMP-показники, TCP-з'єднання, користувачі, процеси);
- метрики виводяться в консоль для контролю;
- формується об'єкт `NetworkFeatures`;
- модель аналізує стан системи з урахуванням як порогових правил, так і інтегрального ризику;
- у разі виявлення аномалій формується структуроване Telegram-сповіщення.

Таким чином, аналітична підсистема виконує не просто збір даних, а інтелектуальну інтерпретацію стану мережі, автоматично перетворюючи технічні показники у зрозумілі повідомлення з оцінкою ризику, що підвищує ефективність реагування на інциденти та дозволяє адаптувати модель до оновлених даних шляхом повторного навчання.

3.3. Тестування моделі інтелектуального аналізу мережевого трафіку

Для оцінювання працездатності розробленої моделі інтелектуального аналізу було проведено комплексне тестування, що охоплює моніторинг у стані спокою, імітацію мережевих атак, реакцію Zabbix-сервера, роботу Python-модуля в реальному часі та генерацію сповіщень у Telegram-боті.

На початковому етапі було зафіксовано значення ключових метрик у стані спокою. На рисунку 3.20 наведено графік, де видно, що всі показники знаходяться в межах норми: вхідний/вихідний трафік залишаються на низьких рівнях, ICMP-затримка мінімальна, втрати пакетів відсутні.

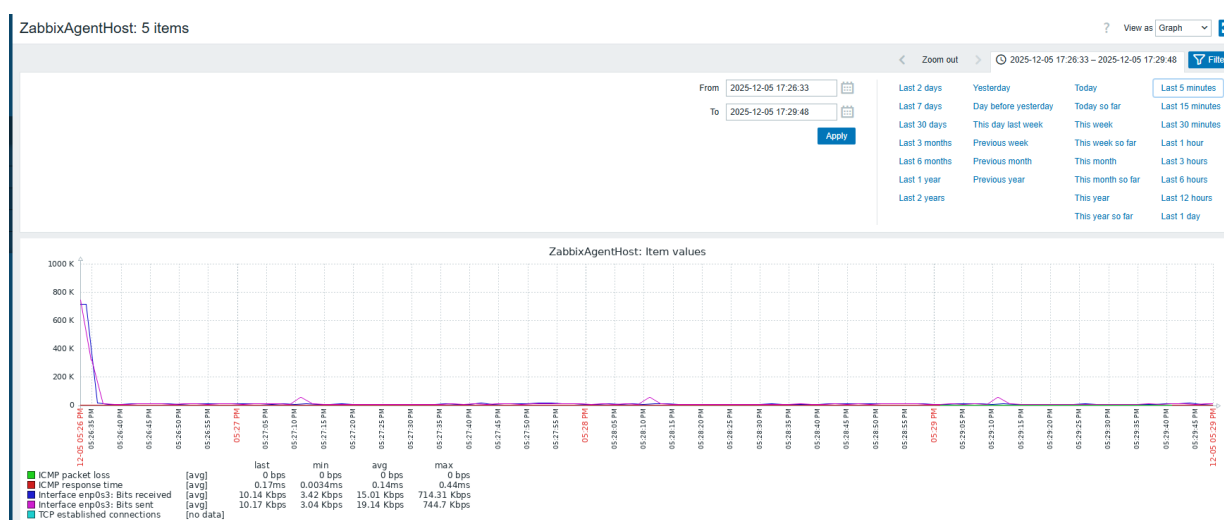


Рисунок 3.20 – Графік моніторингу мережевого трафіку у стані спокою

Далі було створено штучне навантаження на мережевий інтерфейс для моделювання підвищених потоків трафіку. На рисунку 3.21 продемонстровано різке збільшення швидкості отримання та передачі даних, що відображається на графіку у вигляді характерних піків.

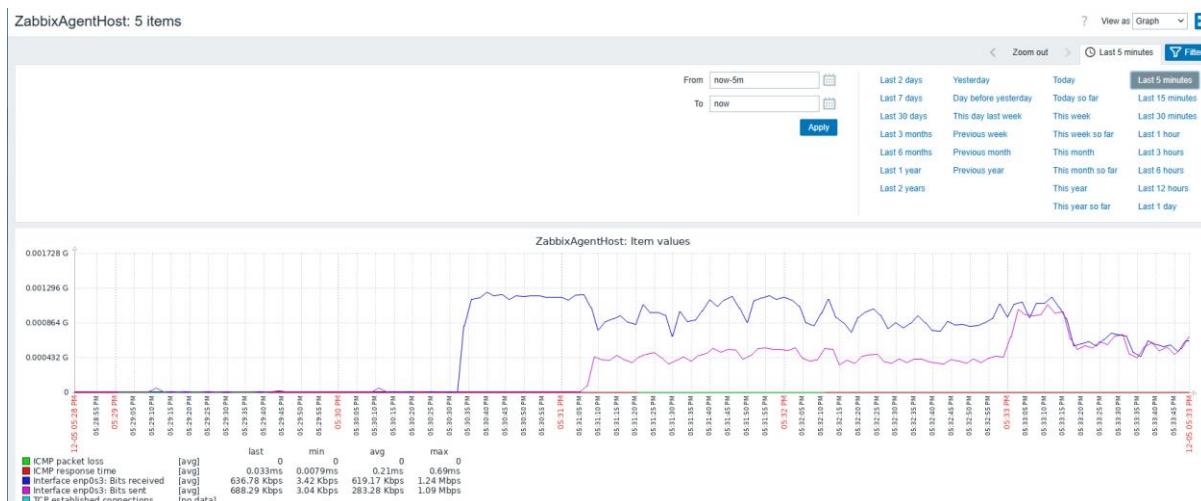


Рисунок 3.21 – Графік моніторингу мережевого трафіку під час навантаження

Для тестування реакції системи на мережеві атаки було використано Kali Linux як атакувальну машину. На рисунку 3.22 наведено запуск атаки типу Ping Flood командою `sudo ping -f 192.168.0.168`. Цей тип атаки генерує надмірну кількість ICMP-запитів, що призводить до суттєвого збільшення затримки та втрати пакетів на цільовому хості.

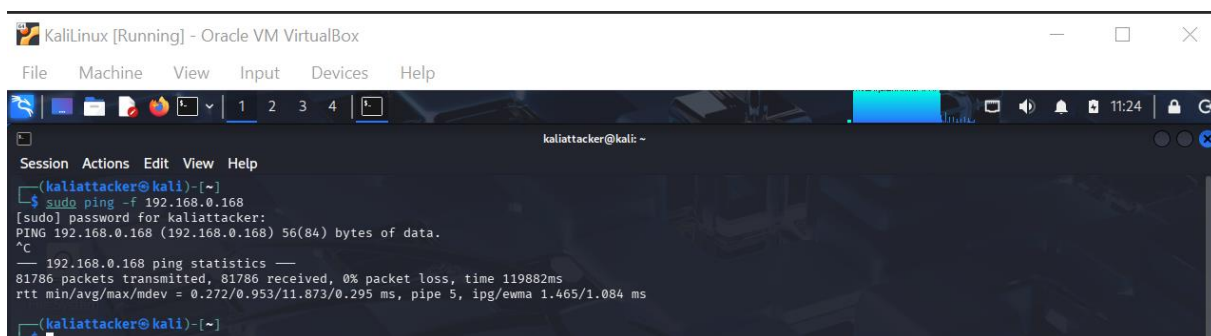


Рисунок 3.22 – Виконання ICMP Ping Flood на атакованому хості

Після запуску атаки Zabbix автоматично зафіксував відхилення та створив записи у розділі Problems. На рисунку 3.23 видно активні тригери, зокрема ті, що пов'язані з перевищенням навантаження CPU та зміною кількості інстальованих пакетів.

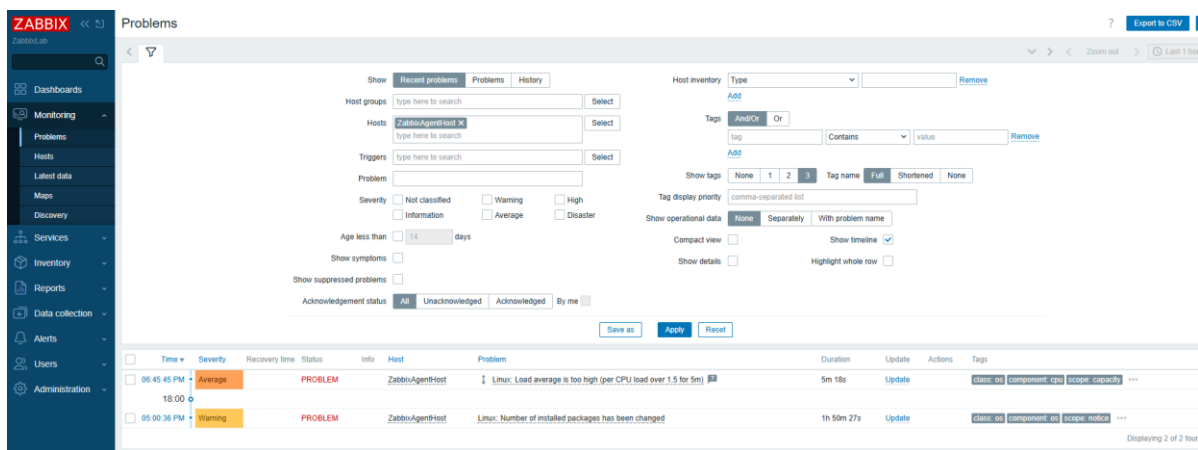


Рисунок 3.23 – Система Zabbix фіксує проблеми під час атаки

Після завершення атаки один із тригерів переходить у стан RESOLVED, як показано на рисунку 3.24, що підтверджує коректну роботу механізмів самодіагностики Zabbix.

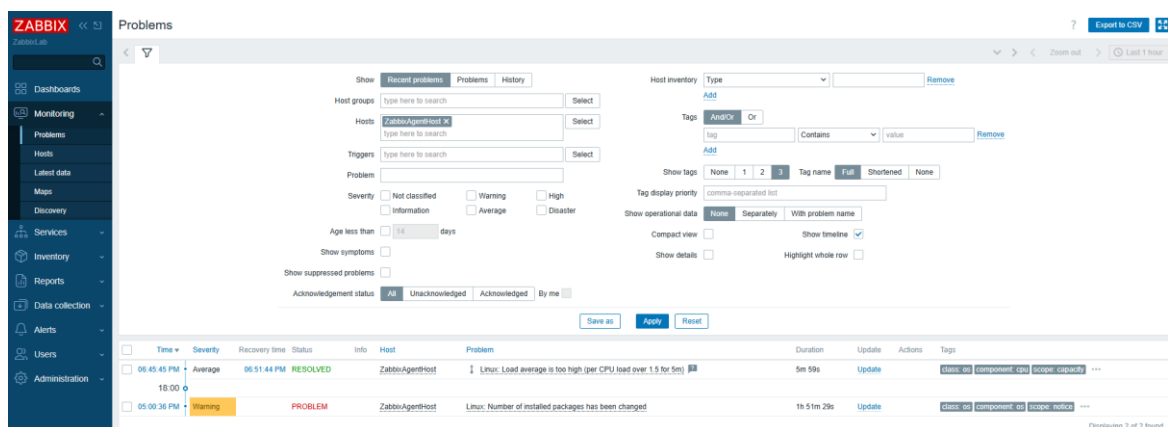


Рисунок 3.24 – Автоматичне відновлення стану після зняття навантаження

На наступному етапі було запущено розроблений Python-модуль моніторингу. На рисунку 3.25 показано фрагмент роботи програми. Дані зчитуються у режимі реального часу, і одразу ж виводяться на екран задані параметри. Система також обчислює інтегральний ризик за допомогою перцептрона та виводить рівень: Normal, Warning, або Critical.

```
Windows PowerShell
KeyboardInterrupt
PS D:\Megan\универ\магістратура\ДИПЛОМ\zabbix-monitor> python zabbix_monitorAI.py
Підключено до Zabbix API. Версія: 7.0.21
---- Поточні значення ----
CPU: 48.48%
Net in: 1353440 bps
Net out: 27688 bps
ICMP loss: 0.0%
ICMP latency: 0.001 s
TCP established: 0
Logged in users: 2
Processes: 14
Інтегральний ризик (0..1): 0.449, рівень: Normal
>> надсилаю в Telegram:
  [x] Виявлено аномальні показники на хості ZabbixAgentHost:

  • Підозра на DDoS/Flood: вхідний трафік 1353440 bps (поріг 400000 bps).

Інтегральний ризик: 0.449 (Normal)
---- Поточні значення ----
CPU: 59.63%
Net in: 891312 bps
Net out: 4680 bps
ICMP loss: 79.8%
ICMP latency: 0.000 s
TCP established: 0
Logged in users: 2
Processes: 14
Інтегральний ризик (0..1): 0.559, рівень: Warning
>> Надсилаю в Telegram:
  [x] Виявлено аномальні показники на хості ZabbixAgentHost:

  • Підозра на DDoS/Flood: вхідний трафік 891312 bps (поріг 400000 bps).
  • Високі втрати ICMP-пакетів: 79.8% (поріг 15.0%). Можливий ICMP flood або перевантаження мережі.
  • Високе навантаження на CPU: 59.63% (поріг 50.0%).

Інтегральний ризик: 0.559 (Warning)
```

Рисунок 3.25 – Робота аналітичного модуля в реальному часі

На рисунку 3.26 демонструється робота Telegram-бота ZabbixAI Alert. При виявленні аномалій формується структуроване повідомлення, де зазначено тип виявленого відхилення (наприклад, “Підозра на DDoS/Flood”), фактичні значення метрик та інтегральний ризик моделі. Повідомлення надходять миттєво, що дозволяє адміністратору оперативно реагувати на атаки.

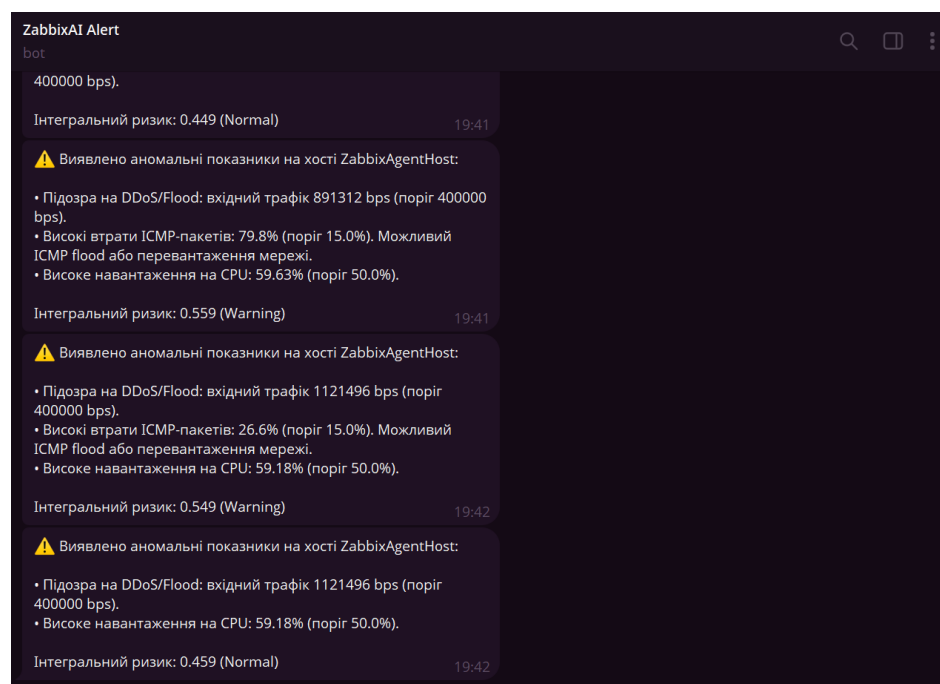


Рисунок 3.26 – Повідомлення Telegram-бота про виявлені аномалії

На рисунку 3.27 наведено тест із навмисним зростанням навантаження CPU, створеним на хості ZabbixAgentHost. Аналітичний модуль виявляє цю аномалію та надсилає відповідне попередження в Telegram, що підтверджує коректну інтеграцію метрик системного рівня.

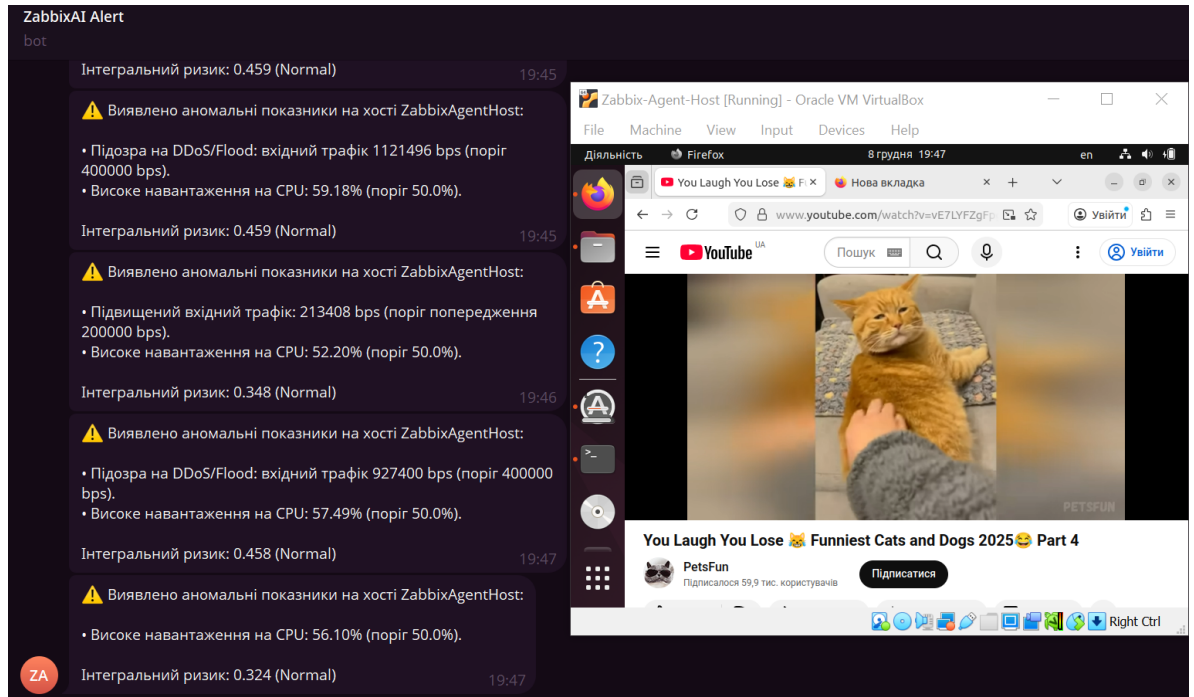


Рисунок 3.27 – Реакція системи на навантаження CPU

Для посиленої атаки було використано інтенсивний ICMP Flood за допомогою скрипта `while true; do ping 192.168.0.168 > /dev/null 2>&1; done`.

На рисунку 3.28 видно, як під час атаки різко зростає затримка ICMP-пакетів, збільшується кількість втрат, а іноді з’являються повідомлення “Destination Host Unreachable”.

```
kaliattacker@kali:~$ sudo ping -s 60000 192.168.0.168
[sudo] password for kaliattacker:
PING 192.168.0.168 (192.168.0.168) 60000(60028) bytes of data.
60008 bytes from 192.168.0.168: icmp_seq=1 ttl=64 time=30.8 ms
60008 bytes from 192.168.0.168: icmp_seq=2 ttl=64 time=28.9 ms
60008 bytes from 192.168.0.168: icmp_seq=3 ttl=64 time=37.5 ms
60008 bytes from 192.168.0.168: icmp_seq=4 ttl=64 time=28.0 ms
60008 bytes from 192.168.0.168: icmp_seq=5 ttl=64 time=27.7 ms
60008 bytes from 192.168.0.168: icmp_seq=6 ttl=64 time=49.4 ms
60008 bytes from 192.168.0.168: icmp_seq=7 ttl=64 time=27.7 ms
60008 bytes from 192.168.0.168: icmp_seq=8 ttl=64 time=30.4 ms
60008 bytes from 192.168.0.168: icmp_seq=9 ttl=64 time=32.7 ms
60008 bytes from 192.168.0.168: icmp_seq=10 ttl=64 time=35.1 ms
60008 bytes from 192.168.0.168: icmp_seq=11 ttl=64 time=30.4 ms
60008 bytes from 192.168.0.168: icmp_seq=12 ttl=64 time=30.1 ms
60008 bytes from 192.168.0.168: icmp_seq=13 ttl=64 time=28.4 ms
60008 bytes from 192.168.0.168: icmp_seq=14 ttl=64 time=31.4 ms

From 192.168.0.189 icmp_seq=56 Destination Host Unreachable
From 192.168.0.189 icmp_seq=61 Destination Host Unreachable
From 192.168.0.189 icmp_seq=62 Destination Host Unreachable
From 192.168.0.189 icmp_seq=64 Destination Host Unreachable
From 192.168.0.189 icmp_seq=67 Destination Host Unreachable
ping: sendmsg: No route to host
From 192.168.0.189 icmp_seq=68 Destination Host Unreachable
From 192.168.0.189 icmp_seq=74 Destination Host Unreachable
From 192.168.0.189 icmp_seq=75 Destination Host Unreachable
From 192.168.0.189 icmp_seq=77 Destination Host Unreachable
ping: sendmsg: No route to host
From 192.168.0.189 icmp_seq=79 Destination Host Unreachable
ping: sendmsg: No route to host
From 192.168.0.189 icmp_seq=80 Destination Host Unreachable
From 192.168.0.189 icmp_seq=86 Destination Host Unreachable
From 192.168.0.189 icmp_seq=88 Destination Host Unreachable
ping: sendmsg: No route to host
^C
--- 192.168.0.168 ping statistics ---
89 packets transmitted, 14 received, +13 errors, 84.2697% packet loss, time 96406ms
rtt min/avg/max/mdev = 27.674/32.038/49.369/5.527 ms, pipe 2

(kaliattacker@kali)-[~]
$ while true; do nc 192.168.0.168 80 >/dev/null 2>&1 & done
```

Рисунок 3.28 – Генерація ICMP Flood на Kali Linux

Рисунок 3.29 демонструє історію метрик після проведення кількох атак. Чітко видно різкі зміни:

- стрибки CPU load,
- піки вхідного трафіку (до десятків Мбіт/с),
- зростання ICMP latency,
- збільшення кількості процесів.

Графік однозначно підтверджує відповідність між періодами проведення атаки та відхиленнями системних параметрів.

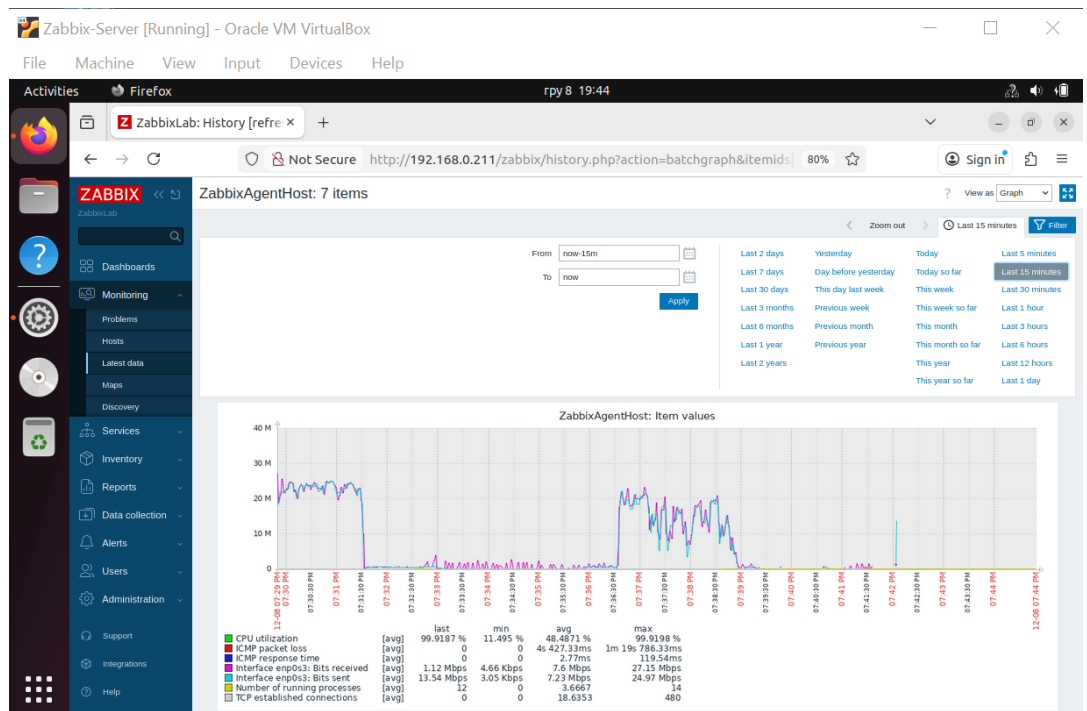


Рисунок 3.29 – Зміна метрик Zabbix під час атак у різні моменти часу

Також для тестування поведінки моделі було проведено серію активних мережевих атак з Kali Linux. Метою тестування є перевірка того, наскільки швидко та коректно модель реагує на різні типи навантажень, а також наскільки стабільно працює Telegram-сповіщення та консольний моніторинг.

На рисунку 3.30 зображено запуск атаки типу ICMP Flood:

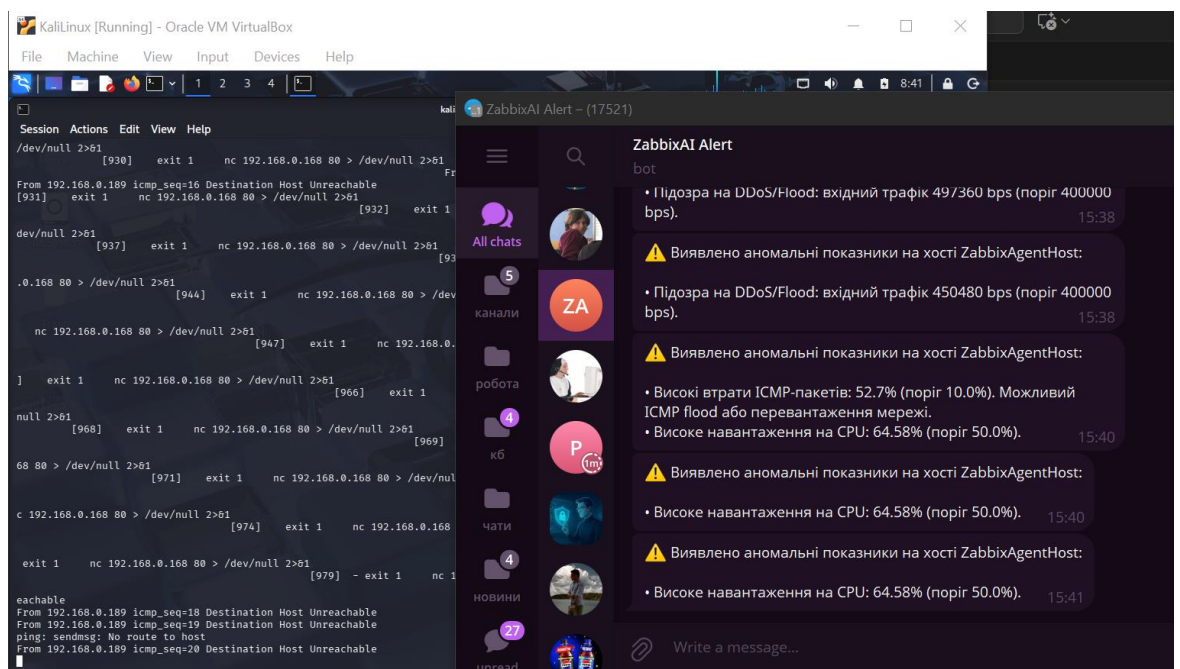


Рисунок 3.30 – Запуск ICMP Flood у Kali Linux та реакція системи

На рисунку 3.31 наведено приклад генерування трафіку:

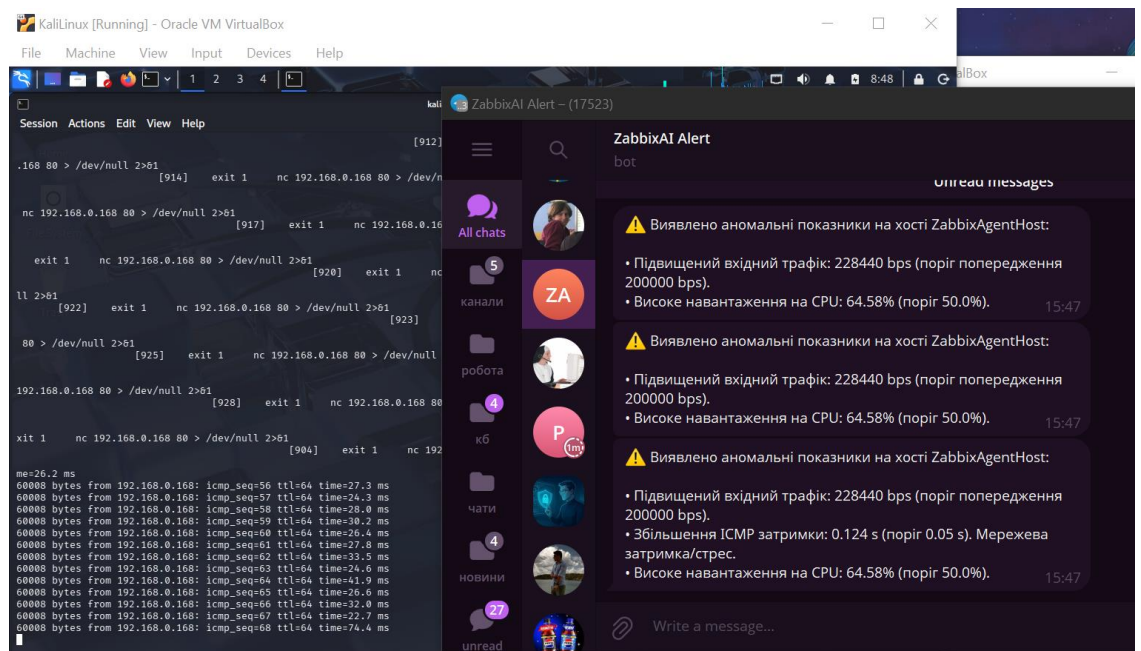


Рисунок 3.31 – Симуляція UDP Flood та сповіщення системи

На рисунку 3.32 показано різке збільшення кількості встановлених TCP-з'єднань:

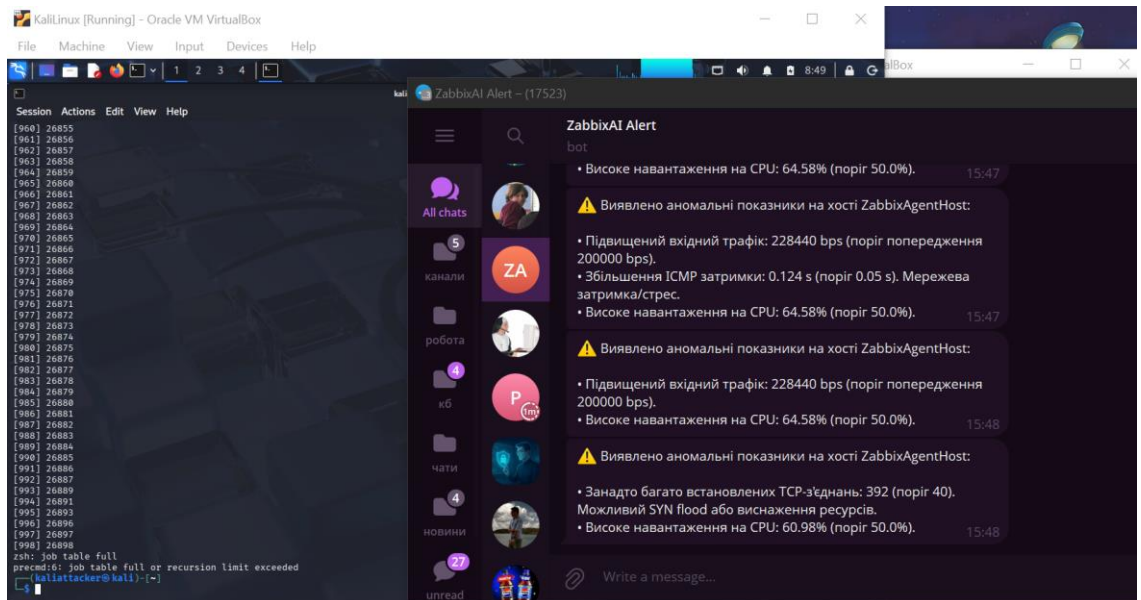


Рисунок 3.32 – Атака типу SYN Flood та повідомлення про неї

У терміналі Python-модуля (рис. 3.33) бачимо:

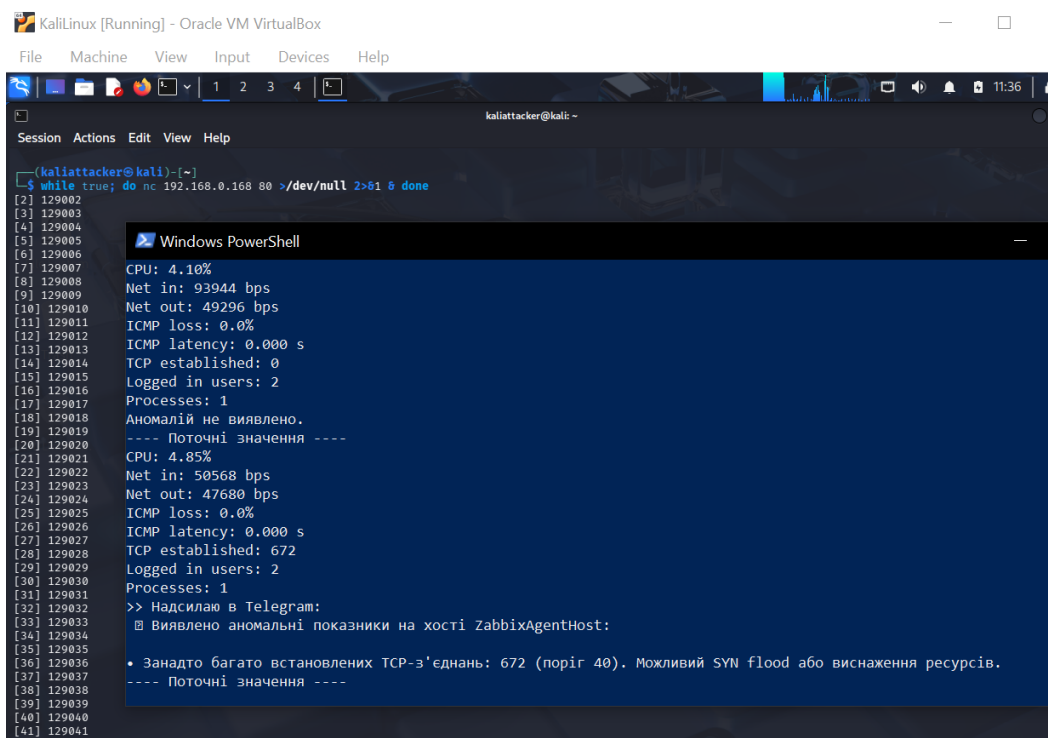


Рисунок 3.33 – Виявлення SYN-flood у Python-модулі (термінал)

Важливо, щоб дані, що відображаються в терміналі повністю збігалися з повідомленнями, отриманими у Telegram (рис. 3.34). Це доводить коректність роботи алгоритму та валідність логіки формування сповіщень.

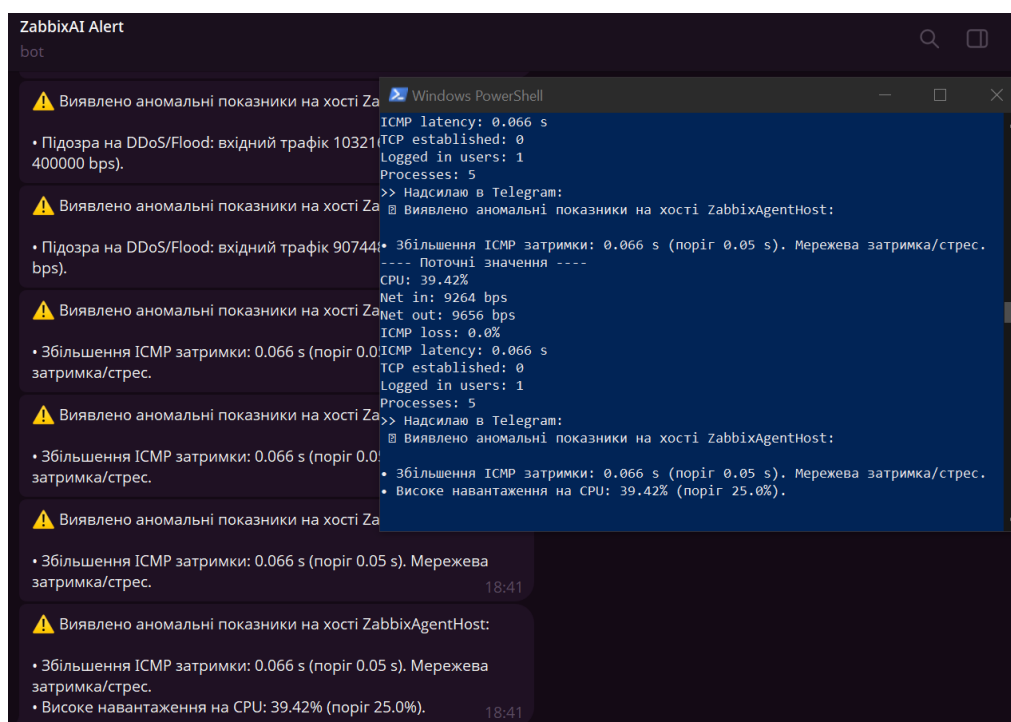


Рисунок 3.34 – Збіг даних аналітичного модуля з Telegram-сповіщеннями

Проведене тестування показало, що розроблена модель коректно реагує на зміни мережевого трафіку та успішно виявляє різні типи аномалій. Усі відхилення були зафіксовані як у консольному виведенні Python-модуля, так і на графіках у Zabbix, при цьому Telegram-бот своєчасно надсилав сповіщення про виявлені ризики. Значення метрик у різних джерелах збігалися, що підтверджує коректність механізмів збору та обробки даних.

У ході тестування rule-based частина моделі стабільно фіксувала типові сценарії аномалій, тоді як навчена перцептронна модель забезпечувала узагальнену оцінку стану системи. Це дозволило виявляти ситуації, коли окремі параметри ще не перевищували критичних значень, але сукупність їхніх змін вказувала на потенційну загрозу.

Загалом система продемонструвала стабільну роботу в режимі реального часу, адекватну реакцію на зміну навантаження та ефективне інформування адміністратора. Результати тестування підтверджують, що модель здатна забезпечувати базовий інтелектуальний аналіз мережевого трафіку та може бути подальше розширена або повторно навчена під інші сценарії.

Висновки до розділу 3:

У цьому розділі було здійснено повну реалізацію та перевірку роботи моделі інтелектуального аналізу мережевого трафіку. Проведено налаштування інфраструктури збору даних у Zabbix, встановлення та конфігурацію агента, а також розгортання аналітичної підсистеми, що включає модулі для збору датасету, навчання моделі та онлайн-моніторингу. Реалізований Python-модуль поєднує rule-based перевірки з перцептронною оцінкою ризику, що підвищує точність виявлення аномалій. Окрему увагу приділено тестуванню моделі під різними сценаріями навантаження та імітованими атаками, результати якого підтвердили коректну взаємодію всіх компонентів і здатність системи стабільно працювати в режимі реального часу.

ВИСНОВКИ

У межах виконаної кваліфікаційної роботи було проведено комплексне дослідження методів моніторингу мережевого трафіку та підходів до інтелектуального аналізу даних, що дало змогу сформулювати теоретичну основу для розробки практичної моделі виявлення аномалій. Аналіз сучасних кібератак та їхніх характерних ознак підтвердив актуальність застосування комбінованих підходів, які поєднують класичний моніторинг із методами аналітичної обробки даних та елементами машинного навчання.

На основі цього було розроблено архітектуру моделі інтелектуального аналізу мережевого трафіку, що інтегрує платформу Zabbix, аналітичний модуль та механізм автоматичного сповіщення. Запропоноване рішення дозволяє обробляти зібрані мережеві метрики, виявляти аномальні стани за допомогою rule-based логіки та оцінювати рівень ризику через одношаровий перцептрон, що поєднує простоту реалізації з достатньою ефективністю для оперативного моніторингу.

Практична реалізація моделі підтвердила її працездатність: система коректно реагує на зміни трафіку, фіксує ознаки можливих атак та формує повідомлення в реальному часі. Проведені експерименти свідчать, що розроблена модель може бути застосована як елемент системи кіберзахисту, забезпечуючи своєчасну індикацію небезпечних відхилень і підвищуючи загальний рівень мережевої безпеки.

Отримані результати демонструють ефективність запропонованого підходу, а також підтверджують його потенціал для подальшого розвитку – зокрема, шляхом розширення набору метрик, удосконалення моделей оцінки ризику або інтеграції більш складних методів машинного навчання.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Splunk. Security Information and Event Management (SIEM). Overview of analytics capabilities. URL: https://www.splunk.com/en_us/blog/learn/siem-security-information-event-management.html?utm_source=chatgpt.com
2. IBM. IBM QRadar SIEM: Advanced Threat Detection and Incident Response. URL: <https://www.ibm.com/products/qradar-siem>
3. Darktrace. Enterprise Immune System: Autonomous Cyber AI. URL: <https://www.darktrace.com/news/darktrace-launches-enterprise-immune-system-version-4>
4. Cisco. Cisco Secure Network Analytics (Stealthwatch). Machine learning-based threat detection overview. URL: <https://www.cisco.com/site/us/en/products/security/security-analytics/secure-network-analytics/index.html>
5. Zabbix. Zabbix Documentation: Monitoring Templates. URL: <https://www.zabbix.com/documentation/current/en/manual/config/templates>
6. Paessler AG. PRTG Network Monitor: Features and Sensors Overview. URL: <https://www.paessler.com/prtg/features>
7. Zabbix. Zabbix Features: Monitoring, Alerting, Reporting. URL: <https://www.zabbix.com/features>
8. Grafana Labs. Grafana: The Open-Source Analytics & Monitoring Platform. URL: <https://grafana.com/docs/grafana/latest/introduction/>
9. Splunk. What Is Splunk? Data Analytics & SIEM Platform Overview. URL: https://www.splunk.com/en_us/blog/learn/what-splunk-does.html
10. Nagios. Nagios Core – Monitoring Engine Overview. URL: <https://www.nagios.org/projects/nagios-core/>
11. Paessler AG. PRTG Network Monitor: All-In-One Network Monitoring Solution. URL: <https://www.paessler.com/prtg>
12. Elastic. Elastic Stack: Search, Analyze, and Visualize Data in Real Time. URL: <https://www.elastic.co/elastic-stack>
13. Elastic. Kibana: Data Visualization & Exploration. URL: <https://www.elastic.co/kibana>

14. Chandola V., Banerjee A., Kumar V. Anomaly Detection: A Survey. ACM Computing Surveys, 2009. DOI: <https://doi.org/10.1145/1541880.1541882>
15. Han J., Kamber M., Pei J. Data Mining: Concepts and Techniques. 3rd ed. Morgan Kaufmann, 2012.
16. Bishop C. M. Pattern Recognition and Machine Learning. Springer, 2006.
17. Goodfellow I., Bengio Y., Courville A. Deep Learning. MIT Press, 2016. URL: <https://www.deeplearningbook.org/>
18. Vinayakumar R., Soman K. P., Poornachandran P. Deep Learning Approaches for Intrusion Detection in Cybersecurity. In: 2017 ICACCI. IEEE, 2017.
19. Scarfone K., Mell P. Guide to Intrusion Detection and Prevention Systems (IDPS). NIST SP 800-94, 2007. URL: <https://csrc.nist.gov/publications/detail/sp/800-94/final>
20. Rosenblatt F. The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain. Psychological Review, 1958.
21. Tan P.-N., Steinbach M., Kumar V. Introduction to Data Mining. Pearson, 2005.
22. Kaisler S., Armour F., Espinosa J. A., Money W. Big Data: Issues and Challenges Moving Forward. HICSS 2013. IEEE.
23. Hutchins E. M., Cloppert M. J., Amin R. M. Intelligence-Driven Computer Network Defense. Lockheed Martin, 2011.
24. APWG. Phishing Activity Trends Report. URL: <https://apwg.org/trendsreports/>
25. Symantec. Internet Security Threat Report (ISTR). URL: <https://www.broadcom.com/support/security-center>
26. Mirkovic J., Reiher P. A Taxonomy of DDoS Attacks and DDoS Defense Mechanisms. ACM SIGCOMM CCR, 2004.
27. Conti M., Dragoni N., Nurmi J. A Survey of Man-In-The-Middle Attacks. IEEE Communications Surveys & Tutorials, 2016.
28. OWASP. SQL Injection. URL: https://owasp.org/www-community/attacks/SQL_Injection

29. Bilge L., Dumitras T. Before We Knew It: An Empirical Study of Zero-Day Attacks. ACM CCS, 2012.
30. Mitnick K., Simon W. The Art of Deception: Controlling the Human Element of Security. Wiley, 2002.
31. Zabbix. Supported Technologies (SNMP, IPMI, JMX, Agent, Traps, Web, API). URL: <https://www.zabbix.com/documentation/current/en/manual/config/items/itemtypes>
32. McKinney W. Python for Data Analysis. 2nd ed. O'Reilly Media, 2017.
33. Pedregosa F. et al. Scikit-learn: Machine Learning in Python. Journal of Machine Learning Research, 2011.
34. Marz N., Warren J. Big Data: Principles and Best Practices of Scalable Realtime Data Systems. Manning Publications, 2015.
35. Zabbix. Database Requirements (MySQL, PostgreSQL, Oracle, SQLite). URL: <https://www.zabbix.com/documentation/current/en/manual/installation/requirements>
36. Grafana Labs. Grafana: The Open-Source Analytics & Monitoring Platform. URL: <https://grafana.com/docs/grafana/latest/introduction/>
37. Telegram. Bots: An Introduction for Developers. URL: <https://core.telegram.org/bots>
38. Finsterbusch M., Richter C., Rocha E., Müller J., Hanssger K. A Survey of Deep Packet Inspection Technology. IEEE Communications Surveys & Tutorials, 2014.
39. Stallings W. Network Security Essentials: Applications and Standards. 6th ed. Pearson, 2017.
40. Douligieris C., Serpanos D. Network Security: Current Status and Future Directions. Wiley-Interscience, 2007.
41. Kompella R. R., Levchenko K., Snoeren A. C., Varghese G. Every Micro-second Counts. SIGCOMM '09. ACM, 2009.
42. Scarfone K., Mell P. Guide to Intrusion Detection and Prevention Systems (IDPS). NIST SP 800-94, 2007.

43. Canonical. Ubuntu Server Documentation: Overview and Use Cases. URL: <https://ubuntu.com/server/docs>
44. Nemeth E., Snyder G., Hein T., Whaley B. UNIX and Linux System Administration Handbook. 5th ed. Addison-Wesley, 2017.
45. Garfinkel S., Spafford G., Schwartz M. Practical UNIX and Internet Security. 4th ed. O'Reilly Media, 2003.
46. Zabbix. Zabbix Agent 2 Documentation: Overview, Installation and Usage. URL: <https://www.zabbix.com/documentation/current/en/manual/concepts/agent2>
47. Oracle. VirtualBox Networking: Bridged Networking Mode. URL: https://www.virtualbox.org/manual/ch06.html#network_bridged
48. Offensive Security. Kali Linux Documentation: Penetration Testing and Ethical Hacking Platform. URL: <https://www.kali.org/docs/>

ДОДАТКИ

Лістинг коду collect_dataset.py

```

from pyzabbix import ZabbixAPI
import time
import csv
import os
from datetime import datetime
from typing import Optional

# ----- НАЛАШТУВАННЯ ZABBIX -----
ZABBIX_URL = "http://192.168.0.211/zabbix"
ZABBIX_USER = "Admin"
ZABBIX_PASSWORD = "StrongPassword"
HOST_ID = "10674"

# КЛЮЧІ ITEM
CPU_KEY = "system.cpu.util[,system]"
NET_IF_IN_KEY = 'net.if.in["enp0s3"]'
NET_IF_OUT_KEY = 'net.if.out["enp0s3"]'
ICMP_LOSS_KEY = "icmppingloss[192.168.0.168]"
ICMP_LAT_KEY = "icmppingsec[192.168.0.168]"
TCP_EST_KEY = "net.tcp.socket.count[,,,,established]"

# ----- ПОРОГИ ДЛЯ АВТОМАТИЧНОЇ МІТКИ -----
CPU_THRESHOLD = 50.0
NET_IN_WARN = 200_000
NET_IN_CRIT = 400_000
NET_OUT_CRIT = 400_000
ICMP_LOSS_THRESHOLD = 15.0
ICMP_LAT_THRESHOLD = 0.1
TCP_EST_THRESHOLD = 40

DATASET_PATH = "dataset.csv"
POLL_INTERVAL_SEC = 15 # інтервал між вимірюваннями

def get_last_value(zapi: ZabbixAPI, key: str):
    items = zapi.item.get(
        hostids=HOST_ID,
        search={"key_": key},
        output=["itemid", "name", "lastvalue", "units"],
        sortfield="itemid",
        limit=1,
    )
    if not items:
        print(f"[WARN] Item з key='{key}' не знайдено")
        return None, None

```

```

item = items[0]
try:
    value = float(item["lastvalue"])
except ValueError:
    value = None

units = item.get("units", "")
return value, units

def compute_label(
    cpu: Optional[float],
    net_in: Optional[float],
    net_out: Optional[float],
    icmp_loss: Optional[float],
    icmp_lat: Optional[float],
    tcp_est: Optional[float],
) -> int:
    """
    Просте rule-based правило для формування мітки:
    1 – якщо помітні аномалії, 0 – інакше.
    Використовується як 'вчитель' для перцептрона.
    """

    alerts = []

    if net_in is not None:
        if net_in > NET_IN_CRIT:
            alerts.append("crit_net_in")
        elif net_in > NET_IN_WARN:
            alerts.append("warn_net_in")

    if net_out is not None and net_out > NET_OUT_CRIT:
        alerts.append("crit_net_out")

    if icmp_loss is not None and icmp_loss > ICMP_LOSS_THRESHOLD:
        alerts.append("icmp_loss")

    if icmp_lat is not None and icmp_lat > ICMP_LAT_THRESHOLD:
        alerts.append("icmp_lat")

    if tcp_est is not None and tcp_est > TCP_EST_THRESHOLD:
        alerts.append("tcp_est")

    if cpu is not None and cpu > CPU_THRESHOLD:
        alerts.append("cpu_high")

    # Якщо є хоч одна аномалія – label = 1
    return 1 if alerts else 0

```

```

def main():
    # Підключення до Zabbix
    zapi = ZabbixAPI(ZABBIX_URL)
    zapi.login(ZABBIX_USER, ZABBIX_PASSWORD)
    print("Підключено до Zabbix API. Версія:", zapi.api_version())

    # Перевіряємо, чи існує файл – якщо ні, записуємо заголовок
    file_exists = os.path.exists(DATASET_PATH)
    if not file_exists:
        with open(DATASET_PATH, mode="w", newline="", encoding="utf-8") as f:
            writer = csv.writer(f)
            writer.writerow([
                "timestamp",
                "cpu",
                "net_in",
                "net_out",
                "icmp_loss",
                "icmp_lat",
                "tcp_est",
                "label",
            ])
        print(f"[INFO] Створено новий файл датасету: {DATASET_PATH}")

    print("[INFO] Починаю збір даних... Натисни Ctrl+C для зупинки.")

    while True:
        cpu, _ = get_last_value(zapi, CPU_KEY)
        net_in, _ = get_last_value(zapi, NET_IF_IN_KEY)
        net_out, _ = get_last_value(zapi, NET_IF_OUT_KEY)
        icmp_loss, _ = get_last_value(zapi, ICMP_LOSS_KEY)
        icmp_lat, _ = get_last_value(zapi, ICMP_LAT_KEY)
        tcp_est, _ = get_last_value(zapi, TCP_EST_KEY)

        label = compute_label(cpu, net_in, net_out, icmp_loss, icmp_lat,
                               tcp_est)

        ts = datetime.now().isoformat(timespec="seconds")

        print("---- Знято значення ----")
        print(f"time={ts}, cpu={cpu}, net_in={net_in}, net_out={net_out}, "
              f"icmp_loss={icmp_loss}, icmp_lat={icmp_lat}, tcp_est={tcp_est}, "
              f"label={label}")

        # Додаємо рядок у CSV
        with open(DATASET_PATH, mode="a", newline="", encoding="utf-8") as f:
            writer = csv.writer(f)
            writer.writerow([
                ts,
                cpu if cpu is not None else "",
                net_in if net_in is not None else "",

```

```
        net_out if net_out is not None else "",
        icmp_loss if icmp_loss is not None else "",
        icmp_lat if icmp_lat is not None else "",
        tcp_est if tcp_est is not None else "",
        label,
    ])

    time.sleep(POLL_INTERVAL_SEC)

if __name__ == "__main__":
    main()
```

Лістинг коду train_model.py

```
import pandas as pd
import numpy as np
from sklearn.linear_model import LogisticRegression

DATASET_PATH = "dataset.csv"

def main():
    # 1. Завантажуємо датасет
    data = pd.read_csv(DATASET_PATH)

    # 2. Вибираємо ознаки (X) та мітки (y)
    # ВАЖЛИВО: порядок ознак має збігатися з порядком у перцептроні:
    # [cpu, net_in, net_out, icmp_loss, icmp_lat, tcp_est]
    feature_cols = ["cpu", "net_in", "net_out", "icmp_loss", "icmp_lat",
                    "tcp_est"]

    # Викидаємо рядки з пропущеними значеннями
    data = data.dropna(subset=feature_cols + ["label"])

    if data.empty:
        print("[ERROR] У датасеті немає повних рядків для навчання.")
        return

    X = data[feature_cols].values.astype("float32")
    y = data["label"].values.astype("int32") # 0 або 1

    print(f"[INFO] Завантажено {len(X)} зразків для навчання.")
    if len(X) < 20:
        print("[WARN] Дуже мало даних для навчання. Бажано мати хоча б 100+ рядків.")

    # 3. Створюємо модель логістичної регресії
    # Це одне лінійне нейронне рівняння + сигмоїда, тобто аналог твого перцептрона.
    model = LogisticRegression(
        solver="lbfgs",
        max_iter=1000
    )

    # 4. Навчання моделі
    print("[INFO] Починаю навчання моделі (LogisticRegression)...")
    model.fit(X, y)

    # 5. Отримуємо ваги та зсув (bias)
    # coef_ має форму (1, n_features), intercept_ – (1,)
    weights = model.coef_[0] # масив довжини 6
```

```

bias = model.intercept_[0] # одне число

print("[INFO] Отримані ваги:", weights)
print("[INFO] Отриманий bias:", bias)

# 6. Зберігаємо у файли, які читає zabbix_monitorAI.py
np.savetxt("weights.txt", weights)
np.savetxt("bias.txt", np.array([bias]))

print("[INFO] Ваги збережено у weights.txt")
print("[INFO] Зсув (bias) збережено у bias.txt")

if __name__ == "__main__":
    main()

```

Лістинг коду zabbix_monitorAI.py

```

from pyzabbix import ZabbixAPI
import requests
import time
from dataclasses import dataclass
from typing import Optional, List
from math import exp
import numpy as np # для завантаження ваг

# ----- НАЛАШТУВАННЯ ZABBIX -----
ZABBIX_URL = "http://192.168.0.211/zabbix"
ZABBIX_USER = "Admin"
ZABBIX_PASSWORD = "StrongPassword"

# hostid ZabbixAgentHost
HOST_ID = "10674"

# КЛЮЧІ ITEM
CPU_KEY = "system.cpu.util[,system]" # Завантаження CPU (system)
NET_IF_IN_KEY = 'net.if.in["enp0s3"]' # Вхідний трафік мережевого
інтерфейсу
NET_IF_OUT_KEY = 'net.if.out["enp0s3"]' # Вихідний трафік мережевого
інтерфейсу
ICMP_LOSS_KEY = "icmpingloss[192.168.0.168]" # Відсоток втрат ICMP-пакетів
до вказаної IP-адреси
ICMP_LAT_KEY = "icmpingsec[192.168.0.168]" # Затримка ICMP-пакетів (ping)
до вказаної IP-адреси
TCP_EST_KEY = "net.tcp.socket.count[,,,established]" # Кількість
встановлених TCP-з'єднань
USERS_KEY = "system.users.num" # кількість користувачів
PROCESSES_KEY = "proc.num[,run]" # кількість процесів у стані run

# ----- ПОРОГИ (параметри rule-based частини моделі) -----
# рогові значення, від яких буду вважати ситуацію підозрілою

CPU_THRESHOLD = 35.0 # %
NET_IN_WARN = 200_000 # bps (0.2 Mbps)
NET_IN_CRIT = 400_000 # bps (0.4 Mbps)
NET_OUT_CRIT = 400_000 # bps (0.4 Mbps)
ICMP_LOSS_THRESHOLD = 15.0 # %
ICMP_LAT_THRESHOLD = 0.1 # s (50 ms)
TCP_EST_THRESHOLD = 40 # к-сть з'єднань

# ----- ЗАВАНТАЖЕННЯ ВАГ ПЕРЦЕПТРОНА -----
# Очікуємо, що weights.txt містить 6 чисел: [w_cpu, w_net_in, w_net_out,
w_icmp_loss, w_icmp_lat, w_tcp_est]
# а bias.txt – одне число.
try:
    W = np.loadtxt("weights.txt")
    B = np.loadtxt("bias.txt")
    W_CPU, W_NET_IN, W_NET_OUT, W_ICMP_LOSS, W_ICMP_LAT, W_TCP_EST =
W.tolist()
    BIAS = float(B)

```

```

        print("[INFO] Ваги перцептрона завантажено з weights.txt / bias.txt")
    except Exception as e:
        print(f"[WARN] Не вдалося завантажити ваги, використовую значення за замовчуванням. Причина: {e}")
        W_CPU      = 0.2
        W_NET_IN    = 0.3
        W_NET_OUT   = 0.1
        W_ICMP_LOSS = 0.2
        W_ICMP_LAT  = 0.15
        W_TCP_EST   = 0.15
        BIAS        = -1.0

# ----- НАЛАШТУВАННЯ TELEGRAM -----
TELEGRAM_TOKEN = "*****" # токен
CHAT_ID = *****        # chat_id

# ----- Повідомлення в TELEGRAM -----
def send_telegram(message: str) -> None:
    url = f"https://api.telegram.org/bot{TELEGRAM_TOKEN}/sendMessage"
    data = {"chat_id": CHAT_ID, "text": message}
    try:
        r = requests.post(url, data=data, timeout=10)
        if r.status_code != 200:
            print(f"Помилка Telegram API: {r.status_code} {r.text}")
    except Exception as e:
        print(f"Помилка відправки в Telegram: {e}")

# ----- Отримання з Zabbix останнього значення item за key_. -----
def get_last_value(zapi: ZabbixAPI, key: str):
    items = zapi.item.get(
        hostids=HOST_ID,
        search={"key_": key},
        output=["itemid", "name", "lastvalue", "units"],
        sortfield="itemid",
        limit=1,
    )
    if not items:
        print(f"Item з key='{key}' не знайдено")
        return None, None

    item = items[0]
    try:
        value = float(item["lastvalue"])
    except ValueError:
        value = None

    units = item.get("units", "")
    return value, units

# ----- ФОРМАЛЬНА МОДЕЛЬ ВХІДНИХ ОЗНАК -----
@dataclass
class NetworkFeatures:
    """
    Вектор ознак мережевого стану (зберігаються всі метрики в одному об'єкті,
    щоб передавати в аналітичну модель).
    """

```

```

cpu: Optional[float]
net_in: Optional[float]
net_out: Optional[float]
icmp_loss: Optional[float]
icmp_lat: Optional[float]
tcp_est: Optional[float]
users: Optional[float]
processes: Optional[float]

# ----- МОДЕЛЬ ІНТЕЛЕКТУАЛЬНОГО АНАЛІЗУ -----

class RuleBasedAnomalyModel:
    """
    Комбінована модель інтелектуального аналізу мережевого трафіку:
    - rule-based правила для виявлення конкретних аномалій;
    - простий перцептрон (1 нейрон) для оцінки інтегрального ризику на основі
    ознак.
    """

    def __init__(
        self,
        cpu_threshold: float,
        net_in_warn: float,
        net_in_crit: float,
        icmp_loss_threshold: float,
        icmp_lat_threshold: float,
        tcp_est_threshold: float,
        w_cpu: float,
        w_net_in: float,
        w_net_out: float,
        w_icmp_loss: float,
        w_icmp_lat: float,
        w_tcp_est: float,
        bias: float,
    ) -> None:
        # порогові значення (rule-based частина)
        self.cpu_threshold = cpu_threshold
        self.net_in_warn = net_in_warn
        self.net_in_crit = net_in_crit
        self.icmp_loss_threshold = icmp_loss_threshold
        self.icmp_lat_threshold = icmp_lat_threshold
        self.tcp_est_threshold = tcp_est_threshold

        # ваги для інтегрального ризику (перцептрон)
        self.w_cpu = w_cpu
        self.w_net_in = w_net_in
        self.w_net_out = w_net_out
        self.w_icmp_loss = w_icmp_loss
        self.w_icmp_lat = w_icmp_lat
        self.w_tcp_est = w_tcp_est
        self.bias = bias

    @staticmethod
    def _sigmoid(x: float) -> float:
        """Сигмоїдна активаційна функція для 'нейрона'.

        Перетворює довільне число у діапазон [0;1], що зручно для інтерпретації
        як рівень ризику (0 - немає ризику, 1 - максимальний).

```

```

"""
try:
    return 1.0 / (1.0 + exp(-x))
except OverflowError:
    # на випадок дуже великих/малих значень
    return 0.0 if x < 0 else 1.0

def _compute_risk_score(self, x: NetworkFeatures) -> float:
    """
    Обчислення інтегрального ризику на основі нормалізованих ознак і ваго-
    вих коефіцієнтів.
    Фактично одношаровий перцептрон:
        risk_raw = w·z + b,
        risk = sigmoid(risk_raw),
    де z – нормалізовані ознаки.
    """

    # Нормалізація: приводимо ознаки до відносної шкали [0..~2],
    # де 1 ~ порогове значення, >1 – перевищення порогу.
    cpu      = x.cpu      if x.cpu      is not None else 0.0
    net_in   = x.net_in   if x.net_in   is not None else 0.0
    net_out  = x.net_out  if x.net_out  is not None else 0.0
    icmp_loss = x.icmp_loss if x.icmp_loss is not None else 0.0
    icmp_lat  = x.icmp_lat if x.icmp_lat is not None else 0.0
    tcp_est  = x.tcp_est  if x.tcp_est  is not None else 0.0

    # Лінійна комбінація нормалізованих ознак (вхід перцептрона)
    risk_raw = (
        self.w_cpu      * cpu +
        self.w_net_in   * net_in +
        self.w_net_out  * net_out +
        self.w_icmp_loss * icmp_loss +
        self.w_icmp_lat  * icmp_lat +
        self.w_tcp_est  * tcp_est +
        self.bias
    )

    # Пропускаємо через сигмоїду, щоб отримати значення в [0..1]
    RISK_SCALE = 8.0 # коефіцієнт "приглушення"
    risk = self._sigmoid(risk_raw / RISK_SCALE)
    return risk

def evaluate(self, x: NetworkFeatures, net_in_units: str = "bps",
net_out_units: str = "bps"):
    """
    Виконує інтелектуальний аналіз (rule-based + перцептрон) та повертає:
    - alerts: список текстових попереджень (якщо виявлені аномалії);
    - risk_score: інтегральний ризик у діапазоні [0..1];
    - risk_level: текстова категорія ризику (Normal / Warning / Criti-
cal).
    """
    alerts: List[str] = []

    # ---- RULE-BASED ЧАСТИНА ----
    # 1) Підозра на DDoS / Flood за вхідним трафіком
    if x.net_in is not None:
        if x.net_in > self.net_in_crit:
            alerts.append(

```

```

        f"Підозра на DDoS/Flood: вхідний трафік {x.net_in:.0f}
{net_in_units or 'bps'} "
        f"(поріг {self.net_in_crit} bps).")
    )
    elif x.net_in > self.net_in_warn:
        alerts.append(
            f"Підвищений вхідний трафік: {x.net_in:.0f} {net_in_units
or 'bps'} "
            f"(поріг {self.net_in_warn} bps).")
        )

    if x.net_out is not None and x.net_out > NET_OUT_CRIT:
        alerts.append(
            f"Підвищений вихідний трафік: {x.net_out:.0f} {net_out_units
or 'bps'} "
            f"(поріг {NET_OUT_CRIT} bps). Можливий витік даних або ано-
мальна активність.")
        )

    # 2) ICMP flood / проблеми з доступністю
    if x.icmp_loss is not None and x.icmp_loss > self.icmp_loss_threshold:
        alerts.append(
            f"Високі втрати ICMP-пакетів: {x.icmp_loss:.1f}% "
            f"(поріг {self.icmp_loss_threshold}%). Можливий ICMP flood або
перевантаження мережі.")
        )

    if x.icmp_lat is not None and x.icmp_lat > self.icmp_lat_threshold:
        alerts.append(
            f"Збільшення ICMP затримки: {x.icmp_lat:.3f} s "
            f"(поріг {self.icmp_lat_threshold} s). Мережева за-
тримка/стрес.")
        )

    # 3) SYN flood / виснаження з'єднань
    if x.tcp_est is not None and x.tcp_est > self.tcp_est_threshold:
        alerts.append(
            f"Занадто багато встановлених TCP-з'єднань: {x.tcp_est:.0f} "
            f"(поріг {self.tcp_est_threshold}). Можливий SYN flood або
виснаження ресурсів.")
        )

    # 4) CPU – загальне навантаження на систему
    if x.cpu is not None and x.cpu > self.cpu_threshold:
        alerts.append(
            f"Високе навантаження на CPU: {x.cpu:.2f}% "
            f"(поріг {self.cpu_threshold}%).")
        )

    # ---- ПЕРЦЕПТРОН: ІНТЕГРАЛЬНИЙ РИЗИК ----
    risk_score = self._compute_risk_score(x)

    # Категоризація ризику за порогоми
    if risk_score >= 0.6:
        risk_level = "Critical"
    elif risk_score >= 0.3:
        risk_level = "Warning"
    else:
        risk_level = "Normal"

```

```

        return alerts, risk_score, risk_level

def main():
    # Підключення до Zabbix
    zapi = ZabbixAPI(ZABBIX_URL)
    zapi.login(ZABBIX_USER, ZABBIX_PASSWORD)
    print("Підключено до Zabbix API. Версія:", zapi.api_version())

    # Ініціалізуємо комбіновану модель інтелектуального аналізу
    model = RuleBasedAnomalyModel(
        cpu_threshold=CPU_THRESHOLD,
        net_in_warn=NET_IN_WARN,
        net_in_crit=NET_IN_CRIT,
        icmp_loss_threshold=ICMP_LOSS_THRESHOLD,
        icmp_lat_threshold=ICMP_LAT_THRESHOLD,
        tcp_est_threshold=TCP_EST_THRESHOLD,
        w_cpu=W_CPU,
        w_net_in=W_NET_IN,
        w_net_out=W_NET_OUT,
        w_icmp_loss=W_ICMP_LOSS,
        w_icmp_lat=W_ICMP_LAT,
        w_tcp_est=W_TCP_EST,
        bias=BIAS,
    )

    # Безкінечний цикл моніторингу
    while True:
        # ---- Зчитуємо всі метрики з Zabbix ----
        cpu, cpu_units = get_last_value(zapi, CPU_KEY)
        net_in, net_in_units = get_last_value(zapi, NET_IF_IN_KEY)
        net_out, net_out_units = get_last_value(zapi, NET_IF_OUT_KEY)
        icmp_loss, _ = get_last_value(zapi, ICMP_LOSS_KEY)
        icmp_lat, _ = get_last_value(zapi, ICMP_LAT_KEY)
        tcp_est, _ = get_last_value(zapi, TCP_EST_KEY)
        users, _ = get_last_value(zapi, USERS_KEY)
        procs, _ = get_last_value(zapi, PROCESSES_KEY)

        # ---- Виводимо в консоль для нагляду ----
        print("---- Поточні значення ----")
        if cpu is not None:
            print(f"CPU: {cpu:.2f}{cpu_units or '%'}")
        else:
            print("CPU: немає даних")

        if net_in is not None:
            print(f"Net in: {net_in:.0f} {net_in_units or 'bps'}")
        else:
            print("Net in: немає даних")

        if net_out is not None:
            print(f"Net out: {net_out:.0f} {net_out_units or 'bps'}")
        else:
            print("Net out: немає даних")

        if icmp_loss is not None:
            print(f"ICMP loss: {icmp_loss:.1f}%")
        if icmp_lat is not None:

```

```

        print(f"ICMP latency: {icmp_lat:.3f} s")
    if tcp_est is not None:
        print(f"TCP established: {tcp_est:.0f}")
    if users is not None:
        print(f"Logged in users: {users:.0f}")
    if procs is not None:
        print(f"Processes: {procs:.0f}")

    # ---- Формуємо вектор ознак (вхід для моделі) ----
    features = NetworkFeatures(
        cpu=cpu,
        net_in=net_in,
        net_out=net_out,
        icmp_loss=icmp_loss,
        icmp_lat=icmp_lat,
        tcp_est=tcp_est,
        users=users,
        processes=procs,
    )

    # ---- ІНТЕЛЕКТУАЛЬНИЙ АНАЛІЗ ----
    alerts, risk_score, risk_level = model.evaluate(
        features,
        net_in_units=net_in_units or "bps",
        net_out_units=net_out_units or "bps",
    )

    print(f"Інтегральний ризик (0..1): {risk_score:.3f}, рівень: {risk_level}")

    # ---- ВІДПРАВКА ALERT У TELEGRAM ----
    # Надсилаються повідомлення, якщо є конкретні rule-based попередження,
    або загальний рівень ризику не Normal
    if alerts or risk_level in ("Warning", "Critical"):
        msg_lines = []

        if alerts:
            msg_lines.append("⚠ Виявлено аномальні показники на хості ZabbixAgentHost:")
            msg_lines.append("")
            msg_lines.extend(f"• {a}" for a in alerts)

        # додаємо інформацію про загальний ризик
        msg_lines.append("")
        msg_lines.append(f"Інтегральний ризик: {risk_score:.3f} ({risk_level})")

        msg = "\n".join(msg_lines)
        print(">> Надсилаю в Telegram:\n", msg)
        send_telegram(msg)
    else:
        print("Аномалій не виявлено, ризик у нормі.")

    # Інтервал опитування
    time.sleep(25)

if __name__ == "__main__":
    main()

```