МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Чорноморський національний університет імені Петра Могили

Факультет комп'ютерних наук

Кафедра комп'ютерної інженерії

ДОПУЩЕНО ДО ЗАХИСТУ

Завідувачка кафедри,

д-р техн. наук, проф.

_____Ірина ЖУРАВСЬКА

« __ » _____ 202__ р.

КВАЛІФІКАЦІЙНА РОБОТА

НА ЗДОБУТТЯ ОСВІТНЬОГО СТУПЕНЯ БАКАЛАВРА

Інтелектуальна система виявлення аномалій мережевого трафіку

Спеціальність 123 Комп'ютерна інженерія

Освітня програма «Комп'ютерна інженерія»

Здобувач _____ Владислав ЄВЛАМПІЄВ

підпис

«__» _____ 202__ р.

Керівник старший викладач

_____ Іван БУРЛАЧЕНКО

підпис

«__» _____ 202__ р.

Миколаїв – 2026

Факультет	Комп’ютерних наук
Кафедра	Комп’ютерної інженерії
Рівень вищої освіти	Перший (бакалаврський)
Освітній ступень	Бакалавр
Спеціальність	123 Комп’ютерна інженерія
Освітня програма	Комп’ютерна інженерія

ЗАТВЕРДЖУЮ
Завідувач кафедри комп’ютерної інженерії
_____ Ірина ЖУРАВСЬКА
«_____» _____ 2025 р.

ЗАВДАННЯ
на кваліфікаційну роботу здобувача

Євлампієва Владислава Юрійовича
(прізвище, ім’я, по батькові здобувача)

1. Тема кваліфікаційної роботи
Інтелектуальна система виявлення аномалій мережевого трафіку

Затверджена наказом по ЧНУ ім. Петра Могили від 25.11.2025 № 294.

2. Строк представлення кваліфікаційної роботи «____» _____ 20__ р.

3. Очікуваний результат роботи та початкові дані, якщо такі потрібні
Очікуваним результатом роботи є прототип апаратно-програмного комплексу інтелектуальної системи виявлення аномалій мережевого трафіку на базі мікросервісної архітектури з підсистемою апаратного сповіщення у режимі реального часу.

4. Перелік питань, що підлягають розробці:
1) проаналізувати та дослідити предмету область, методи захисту інформації та існуючі аналоги систем моніторингу мережевої безпеки;
2) сформувати специфікації функціональних і нефункціональних вимог до розроблюваного апаратно-програмного забезпечення;
3) обґрунтувати вибір апаратних засобів та програмного стеку;
4) розробити мікросервісну архітектуру системи та спроєктувати структурні схеми взаємодії компонентів комплексу;
5) програмно реалізувати модулі аналізу мережевих метаданих, логування інцидентів та системи маршрутизації сповіщень через WebSockets;

6) протестувати працездатність створеного комплексу в умовах імітації аномального мережевого трафіку.

5. Перелік графічних матеріалів

Презентація

6. Консультанти:

Консультант	Кафедра (організація)	Частина роботи

Керівник роботи _____ Іван БУРЛАЧЕНКО
Особистий підпис *Власне ім'я ПРІЗВИЩЕ*

Здобувач _____ Владислав ЄВЛАМПІЄВ
Особистий підпис *Власне ім'я ПРІЗВИЩЕ*

Дата видачі завдання «_____» _____ 2025 р.

КАЛЕНДАРНИЙ ПЛАН

виконання кваліфікаційної бакалаврської роботи

Тема: Інтелектуальна система виявлення аномалій мережевого трафіку

№	Найменування роботи	Початок	Закінчення	Примітки
1	Розробка та затвердження завдання на виконання КБР	04.12.2025	14.12.2025	Виконано
2	Огляд літератури за темою роботи	16.12.2025	20.12.2025	Виконано
3	Складання календарного плану КБР	21.12.2025	23.12.2025	Виконано
4	Аналіз предметної області	04.01.2026	07.01.2026	Виконано
5	Розробка проєктних рішень	09.01.2026	15.01.2026	Виконано
6	Моделювання та конструювання АПЗ	16.01.2026	30.01.2026	Виконано
7	Перевірка працездатності, тестування та апробація розробленого АПЗ, аналіз результатів тестування	08.02.2026	12.04.2026	Виконано
8	Оформлення КБР та презентації	18.05.2026	21.05.2026	Виконано
9	Перший передній захист КБР	22.05.2026	22.05.2026	Виконано
10	Другий передній захист КБР	05.06.2026	05.06.2026	Виконано
11	Завершення оформлення КБР та презентації	08.06.2026	10.06.2026	Виконано
12	Перевірка на академічний плагіат, фальсифікацію та списування	11.06.2026	11.06.2026	Виконано
13	Відгук керівника КБР	12.06.2026	12.06.2026	Виконано
14	Подання КБР рецензенту та рецензування КБР	13.06.2026	14.06.2026	Виконано
15	Подання КБР, її електронної копії та інших документів (відгуку, рецензії)	15.06.2026	17.06.2026	Виконано
16	Захист кваліфікаційної бакалаврської роботи	24.06.2026	24.06.2026	Виконано

Керівник роботи _____ Іван БУРЛАЧЕНКО

Особистий підпис *Власне ім'я ПРІЗВИЩЕ*

Здобувач _____ Владислав ЄВЛАМПІЄВ

Особистий підпис *Власне ім'я ПРІЗВИЩЕ*

«____» _____ 20__ р.

АНОТАЦІЯ

до кваліфікаційної бакалаврської роботи
«Інтелектуальна система виявлення аномалій мережевого трафіку»
Здобувач: Євлампієв Владислав Юрійович
Керівник: ст. викладач Бурлаченко Іван Сергійович

Кваліфікаційна бакалаврська робота присвячена дослідженню методів і засобів моніторингу мережевої безпеки, використанню мікросервісної архітектури як інструменту здійснення інтелектуального аналізу трафіку, а також розробці програмно-апаратної платформи для виявлення аномалій і своєчасного апаратного сповіщення. Актуальність бакалаврської роботи підтверджується спостеріганням тенденції переходу до мікросервісних архітектур при розробці систем моніторингу.

Об’єктом дослідження є процес моніторингу та пошуку аномальної активності у мережевих потоках. Предметом дослідження є апаратно-програмні засоби та алгоритми виявлення аномалій на базі мікросервісів. **Практична цінність системи полягає** у можливості виявляти мережеві атаки на ранніх етапах. На додаток, адміністратори зможуть швидко отримувати повідомлення через різні канали зв'язку. Отримані результати можуть бути використані IT-підрозділами підприємств, центрами обробки даних та адміністраторами невеликих мережевих інфраструктур для автоматизованого моніторингу кіберзагроз.

Кваліфікаційна робота пройшла апробацію під час XVII Міжнародної науково-практичної конференції «Free and Open Source Software» (м. Харків, 11–12 лютого 2026 р.); має відповідні публікації матеріалів тез.

Пояснювальна записка кваліфікаційної бакалаврської роботи складається зі вступу, трьох розділів, висновків та переліку джерел посилання. У вступі роботи визначено актуальність тематики, сформульовані мета, об’єкт, предмет, практичне значення результатів і завдання для виконання.

У першому розділі проведено аналіз предметної області, пов'язаної з інформаційною безпекою, досліджено існуючі методи виявлення аномалій у трафіку та виконано порівняльний аналіз популярних рішень на ринку. У другому розділі було розглянуто та здійснено підбір апаратних компонентів, а саме маршрутизатора MikroTik та мікроконтролера ESP32. Також було обґрунтовано вибір програмних засобів, підібрано системи керування базами даних та спроєктовано мікросервісну архітектуру системи. У третьому розділі кваліфікаційної роботи здійснено опис етапів конструювання апаратно-мережевої частини, програмну реалізацію мікросервісів та алгоритмів детекції загроз за

допомогою аналізу текстових журналів брандмауера за протоколом Syslog, а також проведено тестування платформи в умовах імітації аномального трафіку.

У висновках наведено загальний аналіз виконаної роботи та отриманих результатів проведеного дослідження, виконання поставлених задач і розроблення платформи. У додатку А представлена блок-схема алгоритму функціонування апаратної частини пристрою. У додатку Б наведено лістинг програмного коду для керування мікроконтролером ESP32. У додатку В представлено лістинг програмного коду ядра системи виявлення мережевих аномалій. Додаток Г містить матеріали апробації результатів кваліфікаційної роботи на науково-практичній конференції.

У цілому, кваліфікаційна бакалаврська робота, містить 66 с. (без додатків), 35 рис., 14 табл., 23 джерел посилання та 4 додатки.

Ключові слова: *обробка мережевого трафіку, виявлення аномалій, мікросервісна архітектура, мікроконтролер ESP32, маршрутизатор MikroTik, Інтернет речей, кібербезпека.*

ABSTRACT

of the Bachelor’s Thesis
“Intelligent network traffic anomaly detection system”
Applicant: Vladyslav Yevlampiiev
Supervisor: Senior Lecturer Ivan Burlachenko

The bachelor's thesis is devoted to the study of methods and tools for network security monitoring, the use of microservice architecture as a tool for intelligent traffic analysis, as well as the development of a software-hardware platform for anomaly detection and timely hardware notification. The relevance of the bachelor's thesis is confirmed by the observed trend of transition to microservice architectures in the development of monitoring systems.

The object of the study is the process of monitoring and searching for anomalous activity in network flows. The subject of the study is the hardware-software tools and anomaly detection algorithms based on microservices. The practical value of the system lies in the ability to detect network attacks at an early stage. In addition, administrators will be able to quickly receive notifications through various communication channels. The obtained results can be used by corporate IT departments, data centers, and administrators of small network infrastructures for automated cyber threat monitoring.

The qualification work was tested during the XVII International Scientific and Practical Conference "Free and Open Source Software" (Kharkiv, February 11–12, 2026); it has corresponding publications of abstracts.

The explanatory note of the bachelor's thesis consists of an introduction, three chapters, conclusions, and a list of references. The introduction of the work defines the relevance of the topic, formulates the purpose, object, subject, practical significance of the results, and tasks to be performed.

The first chapter analyses the subject area related to information security, examines existing methods for detecting anomalies in traffic, and carries out a comparative analysis of popular solutions on the market. The second chapter examines and selects the hardware components, namely the MikroTik router and the ESP32 microcontroller. It also justifies the choice of software tools, selects database management systems, and designs the system’s microservice architecture. The third chapter of the thesis describes the stages of designing the hardware and network components, the software implementation of microservices and and threat detection algorithms using firewall log analysis via the Syslog protocol, and the testing of the platform under simulated abnormal traffic conditions.

The conclusions provide a general analysis of the work performed and the results of the study, the achievement of the objectives, and the development of the platform. Appendix A presents a block diagram and the operational algorithm of the device’s hardware. Appendix B contains the source code for controlling the ESP32 microcontroller. Appendix C contains the source code listing of the core network anomaly detection system. Appendix D contains materials relating to the presentation of the results of this thesis at a scientific and practical conference.

In general, the bachelor's thesis, contains 66 pages (excluding appendices), 35 figures, 14 tables, 23 references and 4 appendices.

Keywords: *network traffic processing, anomaly detection, microservices architecture, ESP32 microcontroller, MikroTik router, Internet of Things, cybersecurity.*

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

АПЗ – апаратно-програмне забезпечення
БД – база даних
ЄКТС – Європейська кредитна трансферна накопичувальна система
ІТ – інформаційні технології
ЦОД – центр обробки даних

API – Application Programming Interface
DDoS – Distributed Denial of Service
DoS – Denial of Service
IDS – Intrusion Detection System
IoT – Internet of Things
IP – Internet Protocol
IPS – Intrusion Prevention System
JDK – Java Development Kit
JWT – JSON Web Token
ML – Machine Learning
OLED – Organic Light-Emitting Diode
SOC – Security Operations Center
SSH – Secure Shell
UML – Unified Modeling Language

ВСТУП

Наразі спостерігається як сфера IT розвивається надзвичайно швидко. Обсяги даних у мережах постійно зростають. Разом з цим архітектура корпоративних систем стає все більш важкою, що створює нові виклики для галузі кібербезпеки. Звичайні засоби захисту, які працюють за встановленими правилами, часто не здатні вчасно розпізнати нові або складні атаки. Використання жорстко заданих алгоритмів стає недостатнім, що вимагає необхідність впровадження інтелектуальних підходів. Вони мають аналізувати як саме поводяться мережеві вузли і робити це в режимі реального часу.

Актуальність цієї роботи підтверджується спостеріганням тенденції переходу до мікросервісних архітектур при розробці систем моніторингу. Це дає системі хорошу масштабованість і робить її стійкішою до збоїв. Для того щоб забезпечити цілісність та доступність в сучасних IT-інфраструктурах, важливо створити систему, яка б поєднувала інтелектуальну детекцію, асинхронну обробку подій через брокери повідомлень та гнучку візуалізацію результатів. Якщо інтегрувати все це з професійним обладнанням, наприклад маршрутизатор MikroTik та додати периферію у вигляді мікроконтролера ESP32, то вийде повноцінне програмно-апаратне середовище. Воно дозволить дуже оперативно знаходити мережеві загрози і реагувати на них.

Мета роботи це розробка та тестування програмної системи на базі мікросервісної архітектури для вчасного знаходження аномалій в мережевому трафіку, та сповіщення користувача у вигляді фізичної індикації станів.

Для досягнення поставленої мети необхідно вирішити такі **задачі**:

- проаналізувати, що відбувається з безпекою мережевих архітектур, та дослідити існуючі методи виявлення аномалій;
- обґрунтувати вибір підходів і зібрати технологічний стек для надійності та гнучкості роботи системи;

– сформулювати специфікацію вимог до розроблюваного програмно-апаратного забезпечення, визначивши ключові функціональні та нефункціональні характеристики системи;

– спроєктувати архітектуру системи, продумавши основні модулі автоматизації, збору стастистики і сповіщень;

– реалізувати програмний засіб для роботи з мережевими даними, використовуючи реляційні та повнотекстові сховища;

– впровадити систему миттєвих сповіщень через WebSockets та механізм Webhooks;

– протестувати програмний комплекс в умовах імітації шкідливого мережевого трафіку для перевірки алгоритмів детекції та швидкості спрацьовування апаратних сповіщень.

Об’єктом дослідження є процес моніторингу та пошуку аномальної активності у мережевих потоках.

Предметом дослідження виступають апаратно-програмні засоби та алгоритми виявлення аномалій на базі мікросервісів.

Сфера застосування розробленої системи охоплює IT-підрозділи підприємств, центри обробки даних та невеликі мережеві інфраструктури, де існує потреба в автоматизованому моніторингу кіберзагроз.

Практична цінність системи полягає у можливості виявляти мережеві атаки на ранніх етапах. На додаток, адміністратори зможуть швидко отримувати повідомлення через різні канали зв'язку.

Апробація та публікація: основні теоретичні положення та технічні рішення кваліфікаційної роботи були представлені у формі доповіді та опубліковані у збірнику матеріалів XVII Міжнародної науково-практичної конференції «Free and Open Source Software» : патерни оркестрування у мультиагентних системах аналізу мережевого трафіку» (м. Харків, 11–12 лютого, 2026) [1].

1 АНАЛІЗ ТА ОГЛЯДОГЛЯД ІСНУЮЧИХ РІШЕНЬ ІНТЕЛЕКТУАЛЬНОЇ СИСТЕМИ АНАЛІЗУ МЕРЕЖЕВОГО ТРАФІКУ

1.1 Огляд предметної області

Розвиток сучасного IT призвів до того, що обсяги даних у комп'ютерних мережах зростають ступенево. Мережі стають складнішими, а їхні функціональні можливості розширюються. Але разом із цим пропорційно зростає і кількість загроз інформаційній безпеці. Зараз для будь-якої організації критично важливим завданням є забезпечення конфіденційності, цілісності та нормального доступу до своїх ресурсів. Це регламентується основними положеннями Закону України «Про основні засади забезпечення кібербезпеки України» [2]. Саме тому системи IDS та IPS відіграють ключову роль у захисті мережевого периметра. Проєктування та впровадження таких засобів має здійснюватися відповідно до встановленого порядку проведення робіт з технічного захисту інформації [3].

Раніше було достатньо поставити звичайний міжмережевий екран Firewall із жорстко заданими правилами фільтрації. Проте на сьогодні традиційні підходи є недостатніми, оскільки методи атак постійно вдосконалюються. Зловмисники постійно покращують свої атаки. Вони використовують шифрування, розподілені ботнети і знаходять нові техніки обходу стандартних засобів захисту. Це створює необхідність переходити від статичних методів до більш динамічних систем. Вимагається потреба в розробці рішень, які зможуть виявляти раніше невідомі атаки лише аналізуючи відхилення у поведінці мережі.

1.1.1 Поняття аномалії в мережевому трафіку

Аномалією у мережевому трафіку вважають будь-яку активність, характеристику чи шаблон поведінки, який суттєво відрізняється від того, як мережа зазвичай має працювати в нормі. Такі аномалії можуть бути викликані експлуатаційними або зловмисними факторами.

До небезпечних аномалій відносяться DDoS-атаки, коли обсяг трафіку різко збільшується, щоб виснажити ресурси цільового вузла. У контексті розроблюваної

системи такі атаки фіксуються шляхом аналізу розміру пакетів або їхньої інтенсивності за одиницю часу. Також існує мережеве сканування, коли хтось систематично перебирає порти цільової системи, намагаючись знайти вразливі сервіси для подальших атак. Ще однією загрозою є брутфорс-атаки. Це багаторазові спроби підібрати пароль для автентифікації, що створює специфічні патерни в трафіку. На додаток, варто згадати про витік даних, що являє собою нетипову передачу великих обсягів інформації з внутрішньої мережі на невідомі зовнішні IP-адреси. Наведено приклад графіку аномальної активності в мережевому трафіку (рис. 1.1).

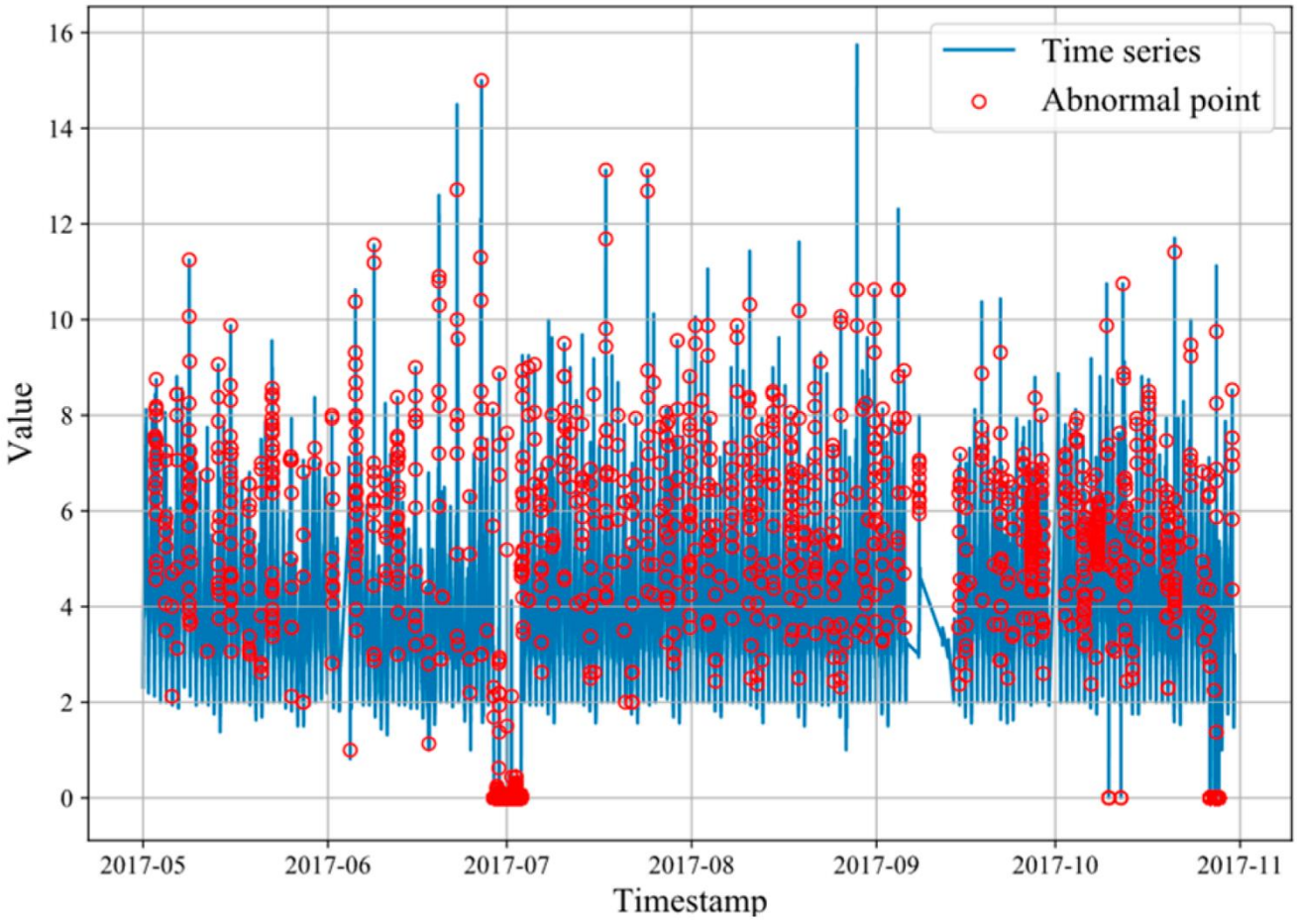


Рисунок 1.1 – Приклад візуального представлення аномальної активності в мережевому трафіку [4]

Як видно з наведеного графіка, візуалізація часових рядів дозволяє чітко відокремити звичайний трафік від раптових аномальних сплесків. Своєчасна

фіксація відхилень є базовим етапом для автоматизованого аналізу та оперативного реагування на потенційні загрози.

1.1.2 Класифікація систем виявлення вторгнень

Системи моніторингу безпеки поділяються на дві групи: сигнатурні та ті, що працюють на основі аномалій.

Сигнатурний аналіз працює доволі просто. Він бере вміст мережевих пакетів і порівнює його з базою даних, де лежать відомі шаблони атак. Це високоефективно для тих загроз, які ми вже знаємо, і такий метод дає мінімальну кількість хибних спрацьовувань. Проте його недоліком є низька ефективність проти нових загроз. Системане може помічати нові типи атак, для яких шаблони ще не були створені. Крім того, зловмисники можуть легко модифікувати код, щоб змінити сигнатуру пакета, залишаючи при цьому саму його шкідливу суть такою ж.

З іншого боку, є аналіз аномалій, який працює за принципом побудови профілю нормальної поведінки мережі. Протягом певного часу система фіксує статистику трафіку та дивиться на середню пропускну здатність. Також аналізується, які порти зазвичай використовуються, які типи протоколів та IP-адреси фігурують між відправниками та отримувачами. Коли профілювання завершено, будь-яке суттєве відхилення від цієї базової моделі одразу сприймається як аномалія. Головний плюс цього підходу це здатність знаходити атаки, про які ще ніхто раніше не знав (табл. 1.1).

Таблиця 1.1 – Порівняльна характеристика методів виявлення мережевих вторгнень

Характеристика	**Сигнатурний аналіз (Signature-based)**	**Аналіз на основі аномалій (Anomaly-based)**
Принцип детекції	Порівняння вмісту пакетів із базою відомих шкідливих шаблонів	Порівняння поточного трафіку з базовим профілем нормальної поведінки мережі

Характеристика	**Сигнатурний аналіз (Signature-based)**	**Аналіз на основі аномалій (Anomaly-based)**
Виявлення нових атак	Неможливо (система не бачить загрози, доки не оновиться база сигнатур)	Можливо (будь-яка нетипова активність фіксується як потенційна атака)
Рівень хибних спрацьовувань	Дуже низький (спрацьовує лише на точний збіг)	Високий (легітимні сплески трафіку можуть розпізнаватися як атака)
Вимоги до обчислювальних ресурсів	Помірні (переважно швидкість доступу до бази даних та оперативна пам'ять)	Високі (потребує постійного статистичного аналізу та профілювання в реальному часі)
Обслуговування	Потребує регулярного оновлення баз сигнатур від постачальників	Потребує періодичного перенавчання моделі нормальної поведінки при зміні топології мережі

Але у систем на основі аномалій є головний мінус, це високий рівень хибних спрацьовувань. Наприклад, легітимне завантаження великого оновлення може бути розпізнане системою як відхилення від звичайної поведінки. Це може бути розпізнано як DDoS-атака.

1.1.3 Інтелектуальні підходи та мікросервісна архітектура в сучасних системах

Щоб виправити недоліки старих класичних методів, сучасні системи детекції все частіше переходять на концепцію інтелектуального аналізу та розподіленої обробки даних. Великі монолітні рішення зазвичай дуже важко масштабувати при зростанні навантаження на мережу. Тому зараз перевагу надають мікросервісній архітектурі. Вона дає можливість чітко розділити ролі. Один сервіс збирає дані, інший їх аналізує, а третій зберігає ці дані. Наприклад, для генерації звітів доцільно використовувати реляційні бази типу Postgres, а для швидкого повнотекстового пошуку та миттєвих сповіщень краще підходить Elasticsearch [5]. Такий підхід до побудови інфраструктури дозволяє дотримуватися вимог Закону України «Про захист інформації в інформаційно-комунікаційних системах» [6] та відповідати

оновленим міжнародним стандартам управління безпекою ДСТУ ISO/IEC 27001:2023 [7].

Варто зазначити, що інтелектуалізація процесу – це не просто використання статистичних порогів. Необхідно, щоб система вміла агрегувати події та аналізувати контекст. Сучасна інфраструктура збору трафіку в системах безпеки найчастіше спирається на експорт текстових журналів подій брандмауера або спеціалізовані протоколи агрегації бінарної телеметрії. У даній роботі маршрутизатор MikroTik працює у ролі сенсора. Він транслює текстові повідомлення брандмауера про транзитні пакети, що суттєво розвантажує обчислювальні ресурси, бо відсутня потреба здійснювати повне копіювання чи глибокий аналіз вмісту кожного пакета. (рис. 1.2) [8].

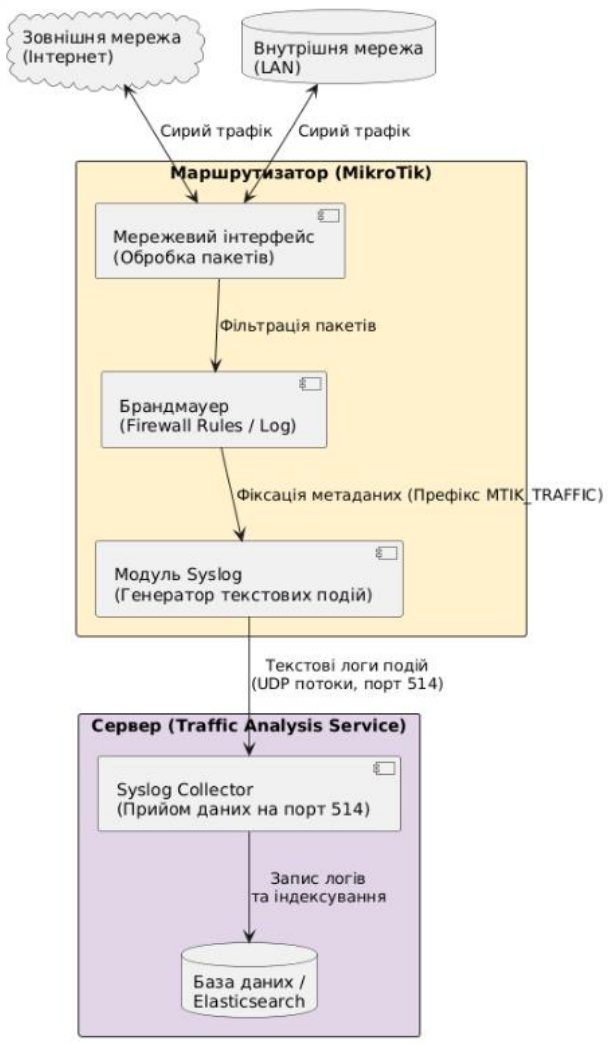


Рисунок 1.2 – Принцип збору та експорту статистики трафіку з маршрутизатора на сервер аналізу

Така архітектурна модель дозволяє суттєво розвантажити мережеве обладнання, оскільки на сервер аналізу передаються лише метадані, а не самі пакети. Це забезпечує високу швидкодію збору статистики та дозволяє системі легко масштабуватися при зростанні обсягів трафіку.

1.1.4 Роль апаратної індикації в системах моніторингу

Окрім програмних інструментів типу вебдашбордів на WebSockets, дуже важливою складовою для швидкої реакції є фізична індикація. Системні адміністратори або оператори SOC не завжди мають змогу здійснювати безперервний моніторинг. Через це було вирішено інтегрувати інтелектуальну систему з IoT-пристроями, зокрема мікроконтролерами серії ESP32 [9]. Це дозволяє вивести сигнал про аномалію в мережі прямо через світлову чи звукову тривогу або через виведення повідомлення на OLED-екран пристрою. Така стратегія створює систему оповіщення, яка значно стійкіша до звичайного людського фактора.

Підсумовуючи, сфера виявлення аномалій у трафіку пройшла шлях від простих брандмауерів до складних мікросервісних систем, так як вони поєднують у собі глибокий аналіз даних, апаратні можливості MikroTik та периферійні пристрої на базі ESP32. Сучасні виклики вимагають саме таких комплексних рішень. Вони мають не тільки вміти помічати загрозу, а й гарантовано та швидко сповістити відповідальний персонал.

1.2 Аналіз існуючих аналогів та рішень для моніторингу мережевої безпеки

Для обґрунтування розробки власної системи детекції аномалій необхідно розглянути існуючі на ринку рішення. На сьогоднішній день сегмент систем виявлення вторгнень представлений як дуже потужними комерційними платформами, так і гнучкими open-source продуктами. Кожне з цих рішень має свою унікальну філософію обробки даних. Варіації йдуть від звичайних порівнянь сигнатур до складних моделей машинного навчання [10;11].

1.2.1 Аналіз традиційних, гібридних систем виявлення вторгнень та платформ з використанням машинного навчання

Одним із найстаріших та найвідоміших представників систем аналізу мережі є система Snort, яка працює переважно на сигнатурному аналізі. Адміністратор створює правила, що описують шкідливу активність, і система порівнює кожен вхідний пакет із цією базою. Головна перевага Snort – це велика спільнота, яка дуже швидко оновлює правила при появі нових вразливостей. Але у системи є серйозні недоліки. Наприклад, архітектурно Snort довго залишався однопотоковим. Це сильно обмежує його продуктивність у сучасних мережах, де швидкості легко перевищують 1 Гбіт/с.

Наступним великим кроком стала поява Suricata (рис. 1.3) [12], яку розробили як більш сучасну альтернативу. Вона вже з коробки підтримує багатопотоковість на рівні ядра. Це дозволяє ефективно розподіляти навантаження по всіх ядрах процесора. Suricata є ідеальним вибором для великих підприємств. Окрім звичайних сигнатур, вона вміє розпізнавати протоколи прикладного рівня і витягувати файли прямо з трафіку, щоб потім їх проаналізувати. Також важливою особливістю є можливість використовувати відеокарти для прискорення обробки пакетів.

Ще однією вагомою перевагою Suricata є її глибока інтеграція з сучасними системами аналітики завдяки стандартизованому формату виведення подій EVE JSON. На відміну від застарілих систем, які генерують неструктуровані текстові логи, Suricata формує деталізовані JSON-об'єкти, що містять вичерпну інформацію про кожен HTTP-запит, DNS-запит чи встановлене TLS-з'єднання. Це дозволяє безшовно інтегрувати її з такими потужними аналітичними платформами, як Elastic Stack або Splunk, для подальшої візуалізації та миттєвого повнотекстового пошуку. Крім того, архітектура рушія підтримує не лише режим пасивного виявлення, але й режим активного запобігання вторгненням, що дає змогу системі автоматично відкидати (англ. drop) шкідливі мережеві пакети на рівні ядра операційної системи ще до того, як вони досягнуть цільового вузла.

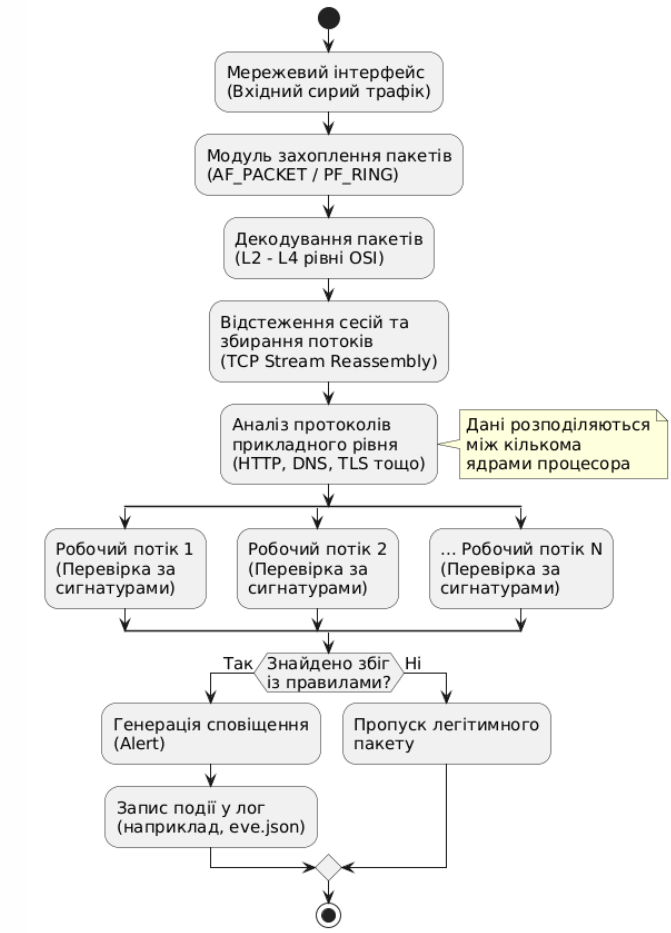


Рисунок 1.3 – Візуалізація розподілу навантаження
в багатопотоковій архітектурі Suricata

Окремо варто виділити систему Zeek, яка раніше називалася Bro. Вона відрізняється від попередніх систем тим, що не просто шукає збіги за правилами, а перетворює сирий мережевий трафік на структуровані логи подій. Це генерує детальні звіти про кожне HTTP-з'єднання чи DNS-запит. У Zeek є своя скриптова мова, яка дозволяє програмувати дуже складні сценарії. Наприклад, можна налаштувати так, щоб система вважала аномалією ситуацію, коли працівник раптом починає завантажувати гігабайти даних у неробочий час на закордонні адреси. Саме цей підхід було взято за основу для архітектури розроблюваної системи аналізу трафіку через її гнучкість та стабільність.

Якщо ж говорити про максимальну інтелектуальність, то існує система Darktrace. Це дорогий комерційний продукт, який позиціонує себе як імунна система мережі. На відміну від безкоштовних рішень, він повністю покладається на машинне навчання. Darktrace не треба писати правила, адже система сама вивчає,

як нормально працює кожен вузол. Коли вона помічає хоч найменше відхилення від цього патерну, одразу сприймає це як загрозу, що є високим показником інтелектуальності. Але ціна такого рішення є надвисокою, плюс закрита архітектура не дає змоги підключити власні апаратні модулі сповіщення.

1.2.2 Порівняльна характеристика аналогів та синтез вимог до розроблюваної системи

Щоб наочно порівняти всі ці системи і зрозуміти, де місце нашої розробки, було створено порівняльну таблицю (табл. 1.2).

Таблиця 1.2 – Порівняльна характеристика систем мережевого моніторингу

Критерій порівняння	**Snort**	**Suricata**	**Zeek (Bro)**	**Darktrace**	**Розроблювана система**
Основний метод детекції	Сигнатурний (Signature-based)	Гібридний (Правила, App Layer)	Поведінковий (Орієнтований на логи)	Машинне навчання	Детермінований (На основі правил та порогових значень)
Архітектура	Монолітна (Однопотокова)	Багатопотокова	Модульна (Script-based)	Пропрієтарна (Closed-source)	Мікросервісна (Docker, Spring Boot)
Обробка великих даних	Середня (масштаб малих офісів , гігабайти на добу)	Висока (масштаб корпоративного сектора, терабайти на добу)	Висока (масштаб магістральних провайдері	Дуже висока (масштаб глобальних хмарних платформ,	Висока (масштаб локальних підприємств, сотні гігабайт

Критерій порівняння	**Snort**	**Suricata**	**Zeek (Bro)**	**Darktrace**	**Розроблювана система**
			в, десятки терабайт)	петабайти на добу)	мережевих логів)
Можливість розширення	Обмежена (Лише правила)	Висока (C/Rust плагіни)	Дуже висока (Zeek скрипти)	Відсутня (Закрита екосистема)	Необмежена (API, Мікросервіси)
Інтеграція з IoT (ESP32)	Складно (Зовнішні скрипти)	Через плагіни (Lua скрипти)	Складно (Зовнішні інтеграції)	Неможливо (Vendor lock-in)	Рідна (Webhooks, WebSockets)

Як видно з таблиці, наявні рішення, хоч і дуже потужні, але вони часто страждають від монолітності або ж є повністю закритими. Оскільки більшість із них просто скидає дані в текстові логи, побудова гнучких екосистем навколо них є складною задачею. Головним недоліком наявних аналогів у контексті даного проєкту є відсутність нативної інтеграції з апаратними компонентами нижнього рівня. Для активації апаратної сигналізації через ESP32 у Snort виникла б необхідність розробки складних зовнішніх скриптів-аналізаторів, тоді як у розроблюваній системі завдяки мікросервісу сповіщень та Webhooks це працює прямо з коробки.

Крім того, використовувати Elastic Stack (Elasticsearch, Logstash, Kibana) для аналогів вимагає значних обчислювальних ресурсів. Тоді як розроблювана мікросервісна модель дозволяє оптимізувати використання пам'яті і пришвидшити реакцію. Завдяки прямій роботі з MikroTik через експорт статистики відсутня потреба у встановленні додаткових програм на комп'ютери користувачів. Ще однією перевагою є візуалізація через WebSockets, де адміністратор бачить кожну аномалію миттєво в браузері, без необхідності постійно перезавантажувати

сторінку. Це є критично важливим для швидких атак типу DDoS, тому розробка власної системи є цілком виправданою.

1.3 Формування вимог до апаратно-програмного забезпечення та опис його поведінки

Для забезпечення ефективності розроблюваного АПЗ необхідно визначити цільову аудиторію та її потреби. Оскільки цільова аудиторія – це мережеві адміністратори та SOC, у кожного з них свої задачі. Адміністратору треба бачити загальний стан трафіку, а спеціаліст з кібербезпеки орієнтований на виявлення цілеспрямованих атак.

1.3.2 Системні вимоги до апаратних та програмних компонентів

З інженерної точки зору, розроблюване АПЗ є комплексом взаємопов'язаних мікросервісів, де вимоги до апаратної частини включають три основні рівні. Першим є маршрутизатор MikroTik, який повинен підтримувати гнучке налаштування правил брандмауера з функцією віддаленого логування подій і бути достатньо потужним, щоб аналізувати пакети і при цьому не знизити швидкість мережі. Другим компонентом виступає сервер з мінімум 8 Гбайт оперативної пам'яті, оскільки Elasticsearch і Java-мікросервіси споживають багато ресурсів. Процесор повинен забезпечувати стабільну роботу Docker-середовища. Третім елементом є мікроконтролер ESP32 з WiFi-модулем, до якого підключена периферія. В розроблюваній системі світлодіодний датчик та OLED-дисплей для виведення сповіщень вже є вбудованими у мікроконтролер (рис. 1.4). Використовуваний периферійний пристрій повинен відповідати нормам безпеки та експлуатації, що визначені у Технічному регламенті радіообладнання [13].

Рисунок 1.4 – Мікроконтролер ESP32 та маршрутизатор MikroTik

Щодо програмних вимог, використовується JDK 21. Вибір стабільної версії Java обумовлений її довготривалою підтримкою та зручністю роботи. Також було використано брокер RabbitMQ для асинхронної передачі повідомлень [14] та Docker Compose для зручної оркестрації інфраструктури.

1.3.3 Функціональні вимоги та опис поведінки системи

Функціональні вимоги описують специфіку поведінки системи, коли отримує вхідні дані, і всі основні операції зведені у табл. 1.3.

Таблиця 1.3 – Специфікація функціональних вимог до АПЗ

ID	**Назва функції**	**Опис поведінки системи**	**Пріоритет (критичність розробки)**
F01	Захоплення даних	Прийом та парсинг статистики трафіку з маршрутизатора MikroTik або симулятора	Високий
F02	Детекція аномалій	Порівняння параметрів пакетів (розмір, порт) з встановленими інтелектуальними порогами	Високий
F03	Збереження звітів	Запис виявлених інцидентів у БД Postgres та індексування в Elasticsearch для пошуку	Середній
F04	Вебвізуалізація	Відображення списку аномалій у браузері в реальному часі через WebSockets	Високий

ID	Назва функції	Опис поведінки системи	Пріоритет (критичність розробки)
F05	Апаратне сповіщення	Надсилання команди на ESP32 для активації фізичної сигналізації при критичній загрозі	Середній
F06	Авторизація	Перевірка прав доступу користувачів до системи за допомогою JWT токенів	Високий

Система повинна працювати передбачувано. Наприклад, якщо фіксується спроба перебору паролів по порту SSH, мікросервіс аналізу генерує подію. Вона через чергу RabbitMQ одночасно надсилається на вебдашборд і на мікроконтролер. Завдяки цьому користувачу взагалі не треба оновлювати сторінку.

1.3.4 Нефункціональні вимоги

Окрім функціональних обов'язків, важливо визначити обмеження. Наприклад, система не повинна автоматично блокувати трафік, залишаючи ухвалення рішень за адміністратором, щоб уникнути ризику випадково відключити легітимних користувачів. Основні нефункціональні вимоги включають цілодобову роботу. У разі виходу з ладу одного з мікросервісів RabbitMQ має зберегти дані в черзі до моменту відновлення його працездатності. Наступною вимогою є затримка від виявлення аномалії до спрацювання сигналу на ESP32 максимум 2 с. Крім того, архітектура має дозволяти додавати нові детектори аномалій без зупинки всієї інфраструктури (рис. 1.5).

Рисунок 1.5 – Схема прецедентів взаємодії користувача з АПЗ

Тестувати створений комплекс потрібно модульно, а також необхідно дати навантаження, згенерувати багато штучного шкідливого трафіку, щоб подивитися, чи витримають алгоритми детекції такий потік. Також важливою є безпека, де використання JWT гарантує, що тільки авторизовані особи зможуть переглядати звіти.

Висновки до розділу 1

У першому розділі було проведено детальний огляд предметної області, пов'язаної з інформаційною безпекою та методами виявлення аномалій у трафіку. Було досліджено, як саме еволюціонували підходи до побудови IDS. Також було проаналізовано різницю між застарілим сигнатурним пошуком та сучасним аналізом поведінкових відхилень. Стало очевидно, що сьогодні вже недостатньо просто мати жорсткі правила фільтрації, оскільки мережі настільки динамічні, що єдиний вихід це перехід на мікросервісну архітектуру.

Також було виконано глибокий порівняльний аналіз популярних рішень на ринку, таких як Snort, Suricata, Zeek та Darktrace. Вони є дуже ефективними у вирішенні своїх вузькоспеціалізованих завдань. Проте більшість із них має закриту або монолітну структуру. Головна проблема полягає в тому, що вони дуже складно

налаштовуються під нестандартні потреби і майже не підтримують нативної інтеграції з периферійними IoT-пристроями нижнього рівня.

Спираючись на недоліки існуючих рішень та реальні потреби цільової аудиторії, було сформовано детальну специфікацію вимог до апаратно-програмного забезпечення. Як апаратну базу розроблюваної системи виявлення аномалій мережевого трафіку було обрано використання маршрутизатора MikroTik як основного сенсора даних та мікроконтролера ESP32 з вбудованим OLED-екраном та WiFi-модулем для фізичної індикації аномалій. Також було сформовано вичерпний перелік ключових функціональних та нефункціональних вимог, які жорстко регламентують поведінку системи.

Запропонований концепт архітектурної моделі є комплексною системою, де незалежні мікросервіси забезпечують паралельну обробку мережевих метаданих, що надходять від маршрутизатора MikroTik. Інтеграція алгоритмів детекції багатовекторних загроз разом із механізмами асинхронної маршрутизації сповіщень через WebSockets та Webhooks формує цілісну, стійку до високих навантажень платформу моніторингу.

2 ПРОЄКТУВАННЯ АПАРАТНО-ПРОГРАМНОГО КОМПЛЕКСУ ВИЯВЛЕННЯ АНОМАЛІЙ МЕРЕЖЕВОГО ТРАФІКУ

2.1 Обґрунтування вибору апаратного забезпечення системи

Для повноцінного функціонування інтелектуальної системи виявлення аномалій мережевого трафіку необхідна наявність апаратного модуля, який виконуватиме роль фізичного термінала індикації та сповіщення. Через те, що програмна частина комплексу побудована на базі мікросервісної архітектури з окремим сервісом сповіщень, мікроконтролер повинен в режимі реального часу отримувати дані від сервера та відображати критичні попередження, наприклад, виявлення DDoS-атаки або сканування портів.

2.1.1 Аналіз та вибір мікроконтролера для фізичної індикації станів

Відповідно до поставленої задачі, мікроконтролер має відповідати таким базовим вимогам:

– наявність вбудованого бездротового модулю Wi-Fi для підключення до локальної мережі та зв’язку з бекенд-сервером;

– достатній обсяг оперативної пам’яті та швидкодія для парсингу JSON-відповідей та роботи з мережевими протоколами (HTTP/REST, WebSockets);

– можливість зручної інтеграції із дисплеями для виведення текстової інформації про аномалію;

– низьке енергоспоживання та компактні розміри.

Для вибору оптимального рішення було проведено порівняльний аналіз п’яти популярних плат розробника. Для порівняння було обрано моделі Arduino Uno R3, STM32F103C8T6 (Blue Pill), NodeMCU (ESP8266), Raspberry Pi Pico W та ESP32 у модифікації з вбудованим OLED-дисплеєм.

Arduino Uno R3 – це найвідоміша у світі платформа для початкового вивчення робототехніки. Вона побудована на базі 8-бітного мікроконтролера Atmega328P. Її головною перевагою є велика спільнота підтримки та простота використання. Проте для розроблюваної системи вона має критичні недоліки. По-

перше, це відсутність вбудованих мережевих інтерфейсів. Вона потребує додаткових модулів, таких як Ethernet Shield або ESP-01. Також ця плата має малий обсяг оперативної пам’яті, що унеможливлює повноцінну обробку складних JSON-пакетів або роботу з шифрованими з’єднаннями HTTPS. Характеристики даного мікроконтролера вказані в табл. 2.1.

Таблиця 2.1 – Короткі характеристики Arduino Uno R3

Характеристика	**Значення**
Архітектура та розрядність	AVR, 8-біт
Тактова частота, МГц	16
Оперативна пам’ять (SRAM), кбайт	2
Флеш-пам’ять, кбайт	32
Вбудований Wi-Fi / Bluetooth	Відсутній / Відсутній
Робоча напруга, В	5

Зважаючи на низьку тактову частоту та критичну нестачу оперативної пам’яті, плата Arduino Uno не може бути використана як повноцінний мережевий термінал. Залучення додаткових зовнішніх модулів зв’язку лише ускладнить розробку, збільшить габарити пристрою та знизить загальну відмовостійкість системи.

Плата STM32F103C8T6 (Blue Pill) використовує 32-бітний процесор з архітектурою ARM Cortex-M3. Вона значно перевершує сімейство Arduino за обчислювальною потужністю та має велику кількість інтерфейсів. Ця плата добре підходить для завдань, що вимагають складної математики або обробки сигналів. Її характеристики вказано в табл. 2.2.

Таблиця 2.2 – Короткі характеристики STM32 (Blue Pill)

Характеристика	**Значення**
Архітектура та розрядність	ARM Cortex-M3, 32-біт
Тактова частота, МГц	72
Оперативна пам’ять (SRAM), кбайт	20

Характеристика	Значення
Флеш-пам’ять, кбайт	64/128
Вбудований Wi-Fi / Bluetooth	Відсутній / Відсутній
Робоча напруга, В	3.3

Як видно з наведених характеристик, незважаючи на потужне 32-бітне обчислювальне ядро, STM32 не має вбудованих засобів для роботи з бездротовими мережами. Це робить її використання недоцільним для реалізації компактного IoT-пристрою індикації. На додаток, вона потребуватиме розробки додаткових плат розширення для Wi-Fi-модулів.

NodeMCU на базі чипа ESP8266 свого часу здійснила революцію у сфері IoT. Дана плата запропонувала інтегрований модуль Wi-Fi та повноцінний 32-бітний процесор на одній компактній платі за дуже низькою ціною. Цей мікроконтролер здатний підключатися до сервера, працювати як вебсервер та виконувати HTTP-запити. Проте його основним недоліком є наявність лише одного обчислювального ядра (табл. 2.3). Під час інтенсивного мережевого обміну, встановлення захищених з’єднань типу TLS/SSL та одночасного оновлення зображення на зовнішньому дисплеї можуть виникати затримки.

Таблиця 2.3 – Короткі характеристики NodeMCU v3 (ESP8266)

Характеристика	Значення
Архітектура та розрядність	Tensilica L106, 32-біт (Одне ядро)
Тактова частота, МГц	80/160
Оперативна пам’ять (SRAM), КБ	~50 доступно для користувача
Флеш-пам’ять, Мбайт	4
Вбудований Wi-Fi / Bluetooth	802.11 b/g/n / Відсутній
Робоча напруга, В	3.3

Аналізуючи дані, наведені в таблиці, можна зробити висновок, що ESP8266 є непоганим бюджетним варіантом для базових систем телеметрії. Проте одноядерна архітектура та обмежений обсяг оперативної пам’яті можуть стати слабким місцем на етапі масштабування розроблюваної системи безпеки, де вимагатиметься швидка реакція на аномалії без втрати продуктивності інтерфейсу користувача.

Сучасний мікроконтролер від Raspberry Pi Foundation побудований на власному чипі RP2040. Модифікація «W» включає надійний бездротовий модуль Infineon CYW43439 з підтримкою Wi-Fi 4. Це потужний двоядерний контролер, який відрізняється унікальною системою програмованого вводу/виводу (англ. Programmed input/output), що дозволяє створювати власні апаратні інтерфейси (табл. 2.4). Плата підтримує розробку на MicroPython, C/C++ або через звичне середовище Arduino IDE, що робить її серйозним конкурентом на ринку сучасних IoT-рішень [15].

Таблиця 2.4 – Короткі характеристики Raspberry Pi Pico W

Характеристика	**Значення**
Архітектура та розрядність	ARM Cortex-M0+, 32-біт (Два ядра)
Тактова частота, МГц	133
Оперативна пам’ять (SRAM), кбайт	264
Флеш-пам’ять, Мбайт	2
Вбудований Wi-Fi / Bluetooth	802.11 b/g/n / Bluetooth 5.2
Робоча напруга, В	3.3

Дані, наведені у таблиці, свідчать про високий технічний потенціал плати Raspberry Pi Pico W для розробки мережевих пристроїв. Однак незважаючи на потужне апаратне забезпечення, інфраструктура сторонніх C++ бібліотек для взаємодії з вебпротоколами та специфічними OLED-дисплеями все ще поступається за рівнем стабільності екосистемі від Espressif.

ESP32 є більш потужним розвитком чипа ESP8266. Він оснащений високопродуктивним двоядерним процесором із тактовою частотою до 240 МГц, має понад 500 кбайт оперативної пам’яті та апаратне прискорення криптографії

(AES, SHA, RSA). Це ідеально підходить для парсингу великих обсягів даних таких як логів аномалій, та забезпечення швидкого шифрованого зв’язку [16].

Для потреб розроблюваної системи було обрано плату розробки ESP32 з інтегрованим 0.96-дюймовим OLED-дисплеєм на базі контролера SSD1306, підключеного по шині I2C та мостом USB-TTL CH340 (рис. 2.1). Наявність дисплея безпосередньо на друкованій платі є вирішальною перевагою (табл. 2.5). Це значно мінімізує кількість необхідних з'єднань для базової візуалізації, залишаючи можливість використання макетної плати для підключення додаткових апаратних буферів та промислової периферії. Характеристики даної плати вказано в табл. 2.5.

Рисунок 2.1 – Зовнішній вигляд обраної плати розробника ESP32

Таблиця 2.5 – Короткі характеристики ESP32 з OLED-дисплеєм

Характеристика	**Значення**
Архітектура та розрядність	Xtensa LX6, 32-біт (Два ядра)
Тактова частота, МГц	160/240
Оперативна пам’ять (SRAM), кбайт	520
Флеш-пам’ять, Мбайт	4

Характеристика	Значення
Вбудований Wi-Fi / Bluetooth	802.11 b/g/n / Bluetooth v4.2 BR/EDR та BLE
Вбудована периферія	OLED екран 0.96» (резолюція 128x64)
Робоча напруга, В	3.3

Характеристики підтверджують, що архітектура ESP32 забезпечує оптимальний баланс між продуктивністю, обсягом пам’яті та мережевими можливостями. Інтегрований дисплей дозволяє розробнику миттєво перейти до написання програмного коду для візуалізації мережевих загроз, уникаючи проблем із апаратним узгодженням компонентів.

Для систематизації наведених даних та остаточного обґрунтування вибору, основні технічні параметри розглянутих мікроконтролерів було зведено до загальної порівняльної таблиці (табл. 2.6).

Таблиця 2.6 – Зведене порівняння мікроконтролерів для фізичної індикації станів

Параметр	Arduino Uno R3	STM32 (Blue Pill)	ESP8266 (NodeMCU)	Rpi Pico W	ESP32 + OLED
Архітектура обчислювача	AVR (8-біт)	ARM (32-біт)	Tensilica (32-біт)	ARM (32-біт)	Xtensa (32-біт)
Кількість ядер, шт.	1	1	1	2	2
Тактова частота, МГц	16	72	80/160	133	240
Оперативна пам’ять (SRAM), кбайт	2	20	50	264	520
Флеш-пам’ять, Мбайт	0.032	0.064	4	2	4

Параметр	Arduino Uno R3	STM32 (Blue Pill)	ESP8266 (NodeMCU)	Rpi Pico W	ESP32 + OLED
Робоча напруга логіки, В	5.0	3.3	3.3	3.3	3.3
Вбудований Wi-Fi	Відсутній	Відсутній	Присутній	Присутній	Присутній
Вбудований Bluetooth	Відсутній	Відсутній	Відсутній	Присутній	Присутній
Наявність інтегрованого дисплея	Відсутній	Відсутній	Відсутній	Відсутній	Присутній (OLED 0.96»)
Апаратне прискорення криптографії	Відсутнє	Відсутнє	Відсутнє	Відсутнє	Присутнє
Цільове призначення платформи	Навчання та базове макетування	Робототехніка, складна математика	Бюджетні датчики розумного дому	Універсальні IoT-застосунки	Складні IoT-системи, мережеві дашборди

На основі проведеного комплексного аналізу для реалізації апаратної підсистеми сповіщення було обрано плату розробника на базі ESP32 з інтегрованим OLED-дисплеєм. Дане рішення покриває усі архітектурні вимоги системи. Наявність вбудованого Wi-Fi для опитування серверного модуля нотифікацій та великого обсягу оперативної пам’яті для стабільної роботи мережевого стеку робить дану плату оптимальним вибором. Іншим ключовим фактором вибору стала наявність двох високочастотних обчислювальних ядер та апаратного прискорення криптографії. Одне ядро мікроконтролера може відповідати за постійне фонове опитування захищених кінцевих точок API, в той час як інше ядро буде без блокувань та затримок оновлювати графічний інтерфейс. Також воно буде на вбудованому дисплеї виводити адміністратору тип виявленої аномалії, IP-адресу джерела загрози, та розмір отриманого пакета в реальному часі.

2.1.2 Аналіз та вибір маршрутизатора в якості мережевого сенсора

У розроблюваній системі виявлення аномалій маршрутизатор виконує функцію мережевого сенсора. Його задача полягає не лише в забезпеченні доступу до Інтернету, а й у безперервному зборі метаданих про весь транзитний трафік з подальшим їх відправленням на сервер аналізу. У маршрутизатор має відповідати таким вимогам:

– підтримка механізмів фільтрації та віддаленого експорту системних журналів безпеки брандмауера без необхідності перепрошивки пристрою, наявність гнучкої операційної системи для налаштування правил брандмауера, дзеркалювання портів та скриптинг-автоматизації;

– достатня обчислювальна потужність процесора для генерації потоків NetFlow без суттєвого зниження загальної пропускної здатності мережі.

Маршрутизатор TP-Link TL-WR840N є одним із найпопулярніших та найдоступніших маршрутизаторів для домашнього використання. Він забезпечує базову маршрутизацію та WiFi-з’єднання на швидкості до 300 Мбіт/с (табл. 2.7). Проте його стандартна прошивка має закритий вихідний код і обмежений функціонал. У ньому повністю відсутня підтримка протоколів збору статистики NetFlow. Для реалізації функцій мережевого сенсора цей пристрій довелося б перепрошивати на альтернативну ОС, що скасовує гарантію та може призвести до нестабільної роботи через малий обсяг оперативної пам’яті.

Таблиця 2.7 – Короткі характеристики TP-Link TL-WR840N

Характеристика	**Значення**
Процесор та частота	MediaTek MT7628NN, 580 МГц
Оперативна пам’ять (RAM), Мбайт	32
Операційна система	TP-Link Firmware
Наявність Wi-Fi	Так (802.11 b/g/n)
Кількість LAN-портів	4 x 10/100 Мбіт/с

За даними, наведеними у таблиці, даний маршрутизатор розроблений виключно для забезпечення базового доступу до мережі. Відсутність вбудованих інструментів для глибокого аналізу трафіку робить його непридатним для використання як сенсор у системах кібербезпеки без суттєвих програмних модифікацій.

Наступне розглянуте рішення – це маршрутизатор Ubiquiti EdgeRouter X (ER-X). Це потужний маршрутизатор корпоративного рівня, який працює під управлінням операційної системи EdgeOS на базі Debian Linux. Він оснащений гігабітними портами та продуктивним процесором, що дозволяє йому легко обробляти великі обсяги даних. Пристрій має нативну підтримку протоколу NetFlow, що робить його чудовим кандидатом для збору статистики. Проте його головним недоліком для даного проєкту є відсутність вбудованого модуля Wi-Fi, так як для бездротової мережі необхідно докуповувати окрему точку доступу, а також значно вища вартість порівняно з конкурентами (табл. 2.8).

Таблиця 2.8 – Короткі характеристики Ubiquiti EdgeRouter X

Характеристика	**Значення**
Процесор та частота	MediaTek MT7621AT (Два ядра), 880 МГц
Оперативна пам’ять (RAM), Мбайт	256
Операційна система	EdgeOS
Наявність Wi-Fi	Відсутня
Кількість LAN-портів	5 x 10/100/1000 Мбіт/с

EdgeRouter X є високопродуктивним рішенням, яке задовольняє вимоги до мережевого моніторингу. Але необхідність використання зовнішніх точок доступу Wi-Fi та загальна висока вартість інфраструктури роблять його використання економічно надлишковим для цілей прототипування або використання в невеликих локальних мережах.

Останнім маршрутизатором та основним вибором для даної роботи є MikroTik RB941-2nD (hAP lite). Модель hAP lite побудована на базі процесора з частотою 650 МГц. Найбільшою перевагою цього пристрою є операційна система

RouterOS. Ця ОС має вбудовану підсистему логування дій брандмауера (англ. Firewall Logging) та нативну підтримку протоколу віддаленого журналювання Syslog. Це дозволяє фіксувати ключові текстові метадані про кожен транзитний пакет і миттєво відправляти їх на віддалений сервер для подальшого аналізу на предмет аномалій (рис. 2.2). Крім того, пристрій містить вбудований WiFi-модуль, що дозволяє контролювати як дротових, так і бездротових клієнтів з одного пристрою (табл. 2.9).

Рисунок 2.2 – Зовнішній вигляд обраного маршрутизатора MikroTik RB941-2nD

Таблиця 2.9 – Короткі характеристики MikroTik RB941-2nD

Характеристика	**Значення**
Процесор та частота	QCA9533, 650 МГц
Оперативна пам’ять (RAM), Мбайт	32
Операційна система	RouterOS (Ліцензія L4)
Наявність Wi-Fi	Так (802.11 b/g/n)
Кількість LAN-портів	4 x 10/100 Мбіт/с

Дані підтверджують, що незважаючи на бюджетні апаратні характеристики, операційна система RouterOS надає цьому пристрою функціонал маршрутизаторів корпоративного класу, що робить його ідеальним сенсором для збору телеметрії.

Для наочної демонстрації переваг обраного пристрою ключові параметри розглянутих маршрутизаторів зведено до загальної таблиці (табл. 2.10).

Таблиця 2.10 – Зведене порівняння маршрутизаторів

Параметр	**TP-Link TL-WR840N**	**Ubiquiti EdgeRouter X**	**MikroTik hAP lite**
Операційна система	Закрита (TP-Link)	EdgeOS (Linux)	RouterOS
Тактова частота процесора, МГц	580	880	650
Обсяг RAM, Мбайт	32	256	32
Вбудований Wi-Fi	Присутній	Відсутній	Присутній
Підтримка скриптів та автоматизації	Відсутня	Присутня	Присутня
Цільове призначення	Базовий домашній доступ	Корпоративні мережі	SOHO мережі, гнучкий моніторинг

На основі проведеного аналізу для виконання ролі мережевого сенсора було обрано маршрутизатор MikroTik RB941-2nD (hAP lite). Вибір зумовлений наявністю професійної операційної системи RouterOS. Це дозволяє без додаткових прошивок конфігурувати пристрій для експорту потоків даних на сервер інтелектуальної системи виявлення аномалій. Крім того, наявність вбудованого модуля Wi-Fi та доступна ціна роблять його оптимальним рішенням для побудови тестового стенда та демонстрації роботи розроблюваного АПЗ.

2.2 Обґрунтування вибору програмних засобів та технологічного стеку

Розробка інтелектуальної системи виявлення аномалій мережевого трафіку передбачає створення складного програмно-апаратного комплексу, який поділяється на два рівні. Першим є рівень периферійних пристроїв, що включає мікроконтролер для фізичної індикації, другим є рівень ядра – це високонавантажений сервер обробки даних та аналітики. Такий поділ вимагає підбору оптимальних мов програмування для кожного з вузлів.

2.2.1 Аналіз та вибір мов програмування для серверної частини та мікроконтролера

Для розробки та прошивки мікропрограмного забезпечення обраного раніше мікроконтролера ESP32 індустріальним стандартом є мова C/C++, разом із застосуванням фреймворків Arduino Core або нативного ESP-IDF. Вибір C++ є безальтернативним з огляду на необхідність прямого та швидкого доступу до апаратних регістрів, керування мережевим стеком Wi-Fi та шиною I2C для виведення інформації на OLED-дисплей. Враховуючи невеликий обсяг оперативної пам’яті мікроконтролера, використання інтерпретованих мов є недоцільним. Все це через високі накладні витрати системних ресурсів та можливі затримки при обробці мережевих пакетів від сервера.

Серверна частина є основною частиною та найважливішим аспектом розробленої системи. Її головні завдання включають безперервний прийом текстових логів брандмауера за протоколом Syslog від маршрутизаторів MikroTik, парсинг мережевих пакетів, аналіз поведінкових патернів для виявлення аномалій, а також маршрутизацію сповіщень. До мови програмування серверної частини висуваються жорсткі вимоги щодо підтримки справжньої багатопотоковості, високої швидкодії та наявності розвиненої екосистеми для побудови мікросервісів.

Для обґрунтування вибору було проведено порівняльний аналіз чотирьох популярних технологій: Python, Node.js, C++ та Java. Порівняння були наведені в таблиці (табл. 2.11).

Таблиця 2.11 – Зведене порівняння мов програмування для серверної частини

Критерій порівняння	**Python (Cpython)**	**Node.js (V8)**	**C++ (GCC/Clang)**	**Java (JVM HotSpot)**
Парадигма виконання коду	Динамічна інтерпретація рядок за рядком (оверхед на типізацію).	JIT-компіляція коду в процесі виконання.	AOT-компіляція (попередня збірка безпосередньо в машинний код).	Компіляція в байт-код + агресивна динамічна JIT-оптимізація під час роботи.
Модель багатопотоковості	Блокується механізмом GIL (Global Interpreter Lock). Паралельні обчислення (CPU-bound) на одному ядрі неможливі.	Однопотоковий Event Loop. Тяжкі обчислення блокують обробку всього іншого мережевого I/O.	Нативні потоки ОС (pthreads). Максимальний контроль, але складна ручна синхронізація.	Нативна потоковість (Pools). Віртуальні потоки (Project Loom) для обробки мільйонів I/O з’єднань одночасно.
Керування оперативною пам’яттю	Підрахунок посилань. Базовий збирач сміття (GC) може викликати затримки при очищенні.	V8 GC. При великих обсягах даних виникають паузи типу «Stop-the-World».	Повністю ручне керування (вказівники). Високий ризик витоків пам’яті (memory leaks) та вразливостей.	Автоматичне, високопродуктивне (G1 GC або ZGC). Дозволяє працювати з купою (Heap) без відчутних пауз.
Екосистема для мікросервісів та брокерів	Flask/FastAPI. Слабко підходить для суворих	Express/NestJS. Добре для легких API, але бракує стандартизованих	Відсутні єдині стандарти для створення	Spring Boot / Spring Cloud. Нативна інтеграція з

Критерій порівняння	Python (Cpython)	Node.js (V8)	C++ (GCC/Clang)	Java (JVM HotSpot)
	корпоративних архітектур через динамічну типізацію.	рішень для складної маршрутизації.	REST API та інтеграції з AMQP (RabbitMQ). Тривалий час розробки.	RabbitMQ, Eureka Server, стандартизовані ORM (Hibernate).
Швидкодія (парсинг трафіку)	Найнижча (через накладні витрати інтерпретатора).	Висока для введення/виведення, падає при CPU-навантаженнях.	Максимальна швидкодія та прямий доступ до мережевого стека.	Близька до нативного C++ завдяки оптимізації JVM для довготривалих серверних процесів.

На основі проведеного аналізу розробки серверної частини системи було обрано мову Java 21. Даний вибір є найбільш обґрунтованим у контексті архітектури розроблюваного проєкту. Відсутність GIL та Event Loop дозволяє Java повноцінно утилізувати всі ядра серверного процесора. Це важливо, оскільки система повинна паралельно прослуховувати порти та одночасно проводити обчислювальний аналіз трафіку без взаємного блокування. Водночас, парсинг великих масивів текстових Syslog-повідомлень брандмауера за секунду створює велике навантаження. Сучасні алгоритми збирання сміття в Java (G1 GC) забезпечують стабільну роботу сервера в режимі 24/7 без витоків пам’яті, на відміну від C++, та без критичних пауз на очищення.

Узагальнюючи, обраний концепт системи безпеки передбачає жорстку розподілену архітектуру. В ній окремо працює модуль виявлення аномалій, модуль авторизації, модуль нотифікацій та сервер виявлення Eureka Server. Фреймворк Spring Boot та екосистема Spring Cloud є індустріальним стандартом Java, що надає готові абстракції для створення захищених кінцевих точок [17]. Наостанок, у розроблюваній системі сервіс аналізу повинен миттєво передавати критичні дані

до сервісу сповіщень. Екосистема Java пропонує надійні бібліотеки для безшовної інтеграції з брокерами повідомлень типу RabbitMQ, що гарантує доставку сповіщень про аномалії навіть при пікових мережевих навантаженнях.

2.2.2 Вибір системи керування базами даних

Проєктування інтелектуальної системи виявлення аномалій мережевого трафіку вимагає особливого підходу до організації збереження даних. Специфіка функціонування платформи полягає в забезпеченні транзакційної надійності при реєстрації поточних мережевих подій та забезпеченні миттєвого повнотекстового пошуку. Також є необхідність проведення складної аналітичної фільтрації серед великої кількості історичних записів про інциденти безпеки. Якщо спробувати використати єдину універсальну базу даних для обох типів навантаження, то це призведе до виникнення критичних архітектурних обмежень. Процеси аналітики блокуватимуть операції запису нових логів трафіку. Для подолання цієї проблеми в архітектурі мікросервісного комплексу реалізовано концепцію гібридного зберігання даних (англ. Polyglot Persistence), яка поєднує можливості об’єктно-реляційної системи керування базами даних Postgres та аналітичного рушія Elasticsearch.

СКБД Postgres виступає як первинне сховище і головне джерело даних для структурованої системної інформації, операційних логів та транзакційних записів. Для функціонування сервісу авторизації, управління обліковими записами користувачів та базового логування параметрів трафіку важливою частиною є повна підтримка вимог ACID [18] (укр. атомарність, узгодженість, ізольованість, довговічність). Postgres забезпечує жорстку типізацію, контроль цілісності даних на рівні схеми через зовнішні ключі та унікальні обмеження. Це робить появу неузгоджених або пошкоджених записів при управлінні профілями користувачів, ролями та сесіями доступу неможливим. Використання інструментів автоматизованої міграції дозволяє контролювати еволюцію реляційної схеми. Це гарантує передбачуване та безпечне розгортання платформи. Уся поточна статистика мережевої активності, яка надходить від сенсорів та підлягає первинній

реєстрації, фіксується у вигляді структурованих об’єктів у реляційних таблицях, що гарантує високу швидкість лінійного запису та безперешкодну інтеграцію із фреймворками об’єктно-реляційного відображення на бекенді.

Надійна обробка мережевої телеметрії у реляційній базі даних досягається за рахунок оптимізації структур зберігання під конкретні типи даних, такі як часові мітки та IP-адреси. Завдяки вбудованим механізмам індексування, Postgres дозволяє сервісу аналізу трафіку швидко записувати та зчитувати поточні агреговані показники для виявлення аномалій, таких як DDoS. Реляційне сховище мінімізує накладні витрати на збереження метаданих кожного мережевого пакета. Це забезпечує високу щільність пакування даних на фізичних носіях. Крім того, наявність надійних транзакційних механізмів гарантує, що у разі виникнення раптових збоїв під час генерації або обробки аналітичних звітів стан системи залишиться абсолютно узгодженим.

Завдання збереження, індексації та аналізу виявлених інцидентів безпеки покладено на аналітичну систему Elasticsearch. Специфіка аномалій мережевого трафіку полягає в тому, що звіти про них мають напівструктурну форму, представляючи собою складні об’єкти з великою кількістю динамічних параметрів. Ці об’єкти можуть бути переліками атакованих портів, сигнатурами загроз, географічними даними джерела або поведінковими метриками. Використання класичних реляційних баз даних для пошуку за такими критеріями вимагало б створення великої кількості таблиць та складних операцій об’єднання, що призводило б до падіння швидкодії при збільшенні обсягів даних. Elasticsearch, працює як документоорієнтована SQL. Вона зберігає інформацію у вигляді JSON-документів та автоматично будує інвертований індекс за всіма наявними полями [19]. Це дозволяє адміністратору безпеки виконувати миттєві складні пошукові запити, здійснювати фільтрацію за довільними комбінаціями параметрів або аналізувати часові ряди інцидентів з мінімальною затримкою.

2.2.3 Вибір додаткових інструментів взаємодії

Сучасна мікросервісна архітектура вимагає надійних механізмів взаємодії між ізольованими компонентами. Для забезпечення високої доступності та швидкодії системи виявлення аномалій було обрано спеціалізовані інфраструктурні інструменти та мережеві протоколи. Розділення бекенду на незалежні модулі створює потребу в гарантованій доставці повідомлень. Використання прямих синхронних HTTP-запитів між ними є ризикованим, бо тимчасова недоступність модуля сповіщень може призвести до втрати даних про загрозу або заблокувати роботу основного аналізатора.

Для вирішення цієї проблеми як єдиний брокер повідомлень було обрано RabbitMQ (рис. 2.3). Він реалізує розширений протокол AMQP та забезпечує надійну передачу подій. Коли модуль аналізу фіксує аномалію, він миттєво публікує дані у чергу і продовжує обробку наступних пакетів. Сервіс сповіщень виступає як споживач, та асинхронно зчитує ці події. Це гарантує швидку маршрутизацію інцидентів безпеки з мінімальною затримкою.

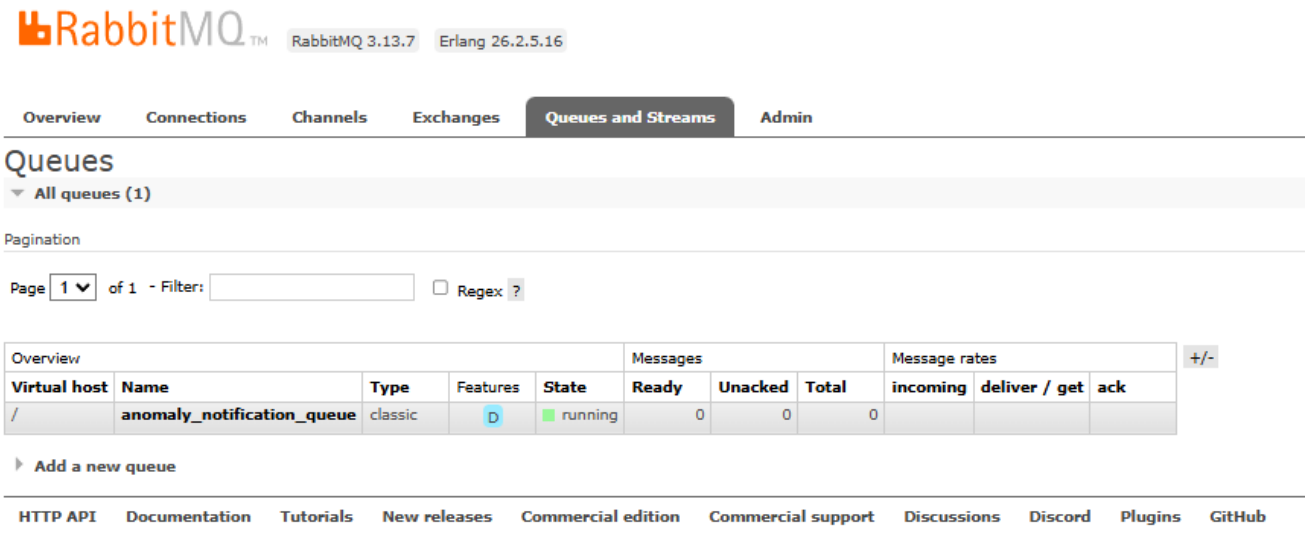


Рисунок 2.3 – Панель керування та моніторингу черг повідомлень RabbitMQ

Кожен мікросервіс може динамічно змінювати свою IP-адресу та порт під час роботи в середовищі Docker, тому жорстке конфігурування мережевих параметрів

буде неефективним. Для забезпечення динамічної маршрутизації було інтегровано сервіс Spring Cloud Netflix Eureka [20].

Цей інструмент працює як централізований реєстр (рис. 2.4). Під час запуску кожен модуль автоматично реєструється на сервері Eureka та регулярно надсилає сигнали перевірки працездатності. Це дозволяє сервісам знаходити один одного за логічними іменами та надає адміністратору графічний інтерфейс для візуального моніторингу активних вузлів.


nstances currently registered with Eureka

Application	AMIs	Availability Zones	Status
AUTH-SERVICE	n/a (1)	(1)	UP (1) - localhost:auth-service:8081
NOTIFICATION-SERVICE	n/a (1)	(1)	UP (1) - localhost:notification-service:8083
TRAFFIC-SERVICE	n/a (1)	(1)	UP (1) - localhost:traffic-service:8082

Рисунок 2.4 – Інтерфейс моніторингу мікросервісів у панелі Eureka Server

Взаємодія бекенду з адміністратором та мікроконтролером ESP32 базується на комбінації протоколів HTTP/REST та WebSockets. Класичний REST API використовується для виконання базових операцій авторизації користувачів, управління налаштуваннями та отримання історичних звітів з бази даних.

Для швидшої передачі критичних сповіщень було впроваджено протокол WebSockets. Він встановлює постійне з’єднання між сервісом нотифікацій та кінцевим пристроєм. Як тільки система фіксує нову атаку, вона миттєво відправляє push-повідомлення на OLED-дисплей ESP32, забезпечуючи реакцію на аномалію за лічені мілісекунди.

2.3 Формування концепції та архітектури розроблюваної системи

2.3.1 Загальна концепція та логіка функціонування платформи

На першому етапі головною дією виступає маршрутизатор MikroTik. Він функціонує як граничний шлюз локальної мережі, через який проходить весь вхідний та вихідний трафік користувачів та IoT-пристроїв. Завдяки активованій функції Firewall Logging, маршрутизатор не тільки пересилає пакети, а й логує

базові метадані про кожне мережеве з'єднання. Ці дані автоматично формуються у текстові повідомлення журналу безпеки і безперервно відправляються на IP-адресу центрального сервера системи за протоколом UDP. Такий підхід гарантує, що процес моніторингу не впливатиме на пропускну здатність самого маршрутизатора і не створюватиме затримок для легітимних користувачів.

Другий етап включає в себе інтелектуальний аналіз. Мережеві метадані потрапляють до мікросервісного бекенду. В бекенді, безперервний потік розбивається на окремі об’єкти та аналізується. Концепція системи передбачає виявлення мінімум двох типів класичних аномалій. Це можуть бути об’ємні аномалії, такі як DDoS-атаки або флуд, та розвідувальні, як сканування портів, коли фіксується спроба підключення до великої кількості різних портів за короткий проміжок часу. Усі отримані дані каталогізуються у реляційній базі даних для можливості подальшого ретроспективного аудиту.

Останнім процесом розробленої системи є візуальна індикація помилки на мікроконтролері ESP32. У разі підтвердження аномалії система генерує сповіщення. Щоб уникнути ситуації, коли адміністратор пропускає критичний інцидент, концепція передбачає використання апаратного індикатора на базі мікроконтролера ESP32. Отримавши сигнал через WebSocket, мікроконтролер миттєво виводить на вбудований OLED-дисплей тип виявленої загрози, IP-адресу зловмисника, та розмір отриманого пакета, привертаючи увагу фахівця у режимі реального часу.

2.3.2 Проєктування мікросервісної архітектури програмного комплексу

Для забезпечення високої відмовостійкості та можливості незалежного масштабування компонентів серверну частину системи спроєктовано за класичною мікросервісною архітектурою. Весь програмний комплекс розбито на чотири ізольовані сервіси. Кожен з цих сервісів виконує суворо визначену функцію і взаємодіє з іншими через стандартизовані інтерфейси або брокери повідомлень. Концептуальну схему мікросервісної архітектури розроблюваної системи наведено на рис. 2.5.

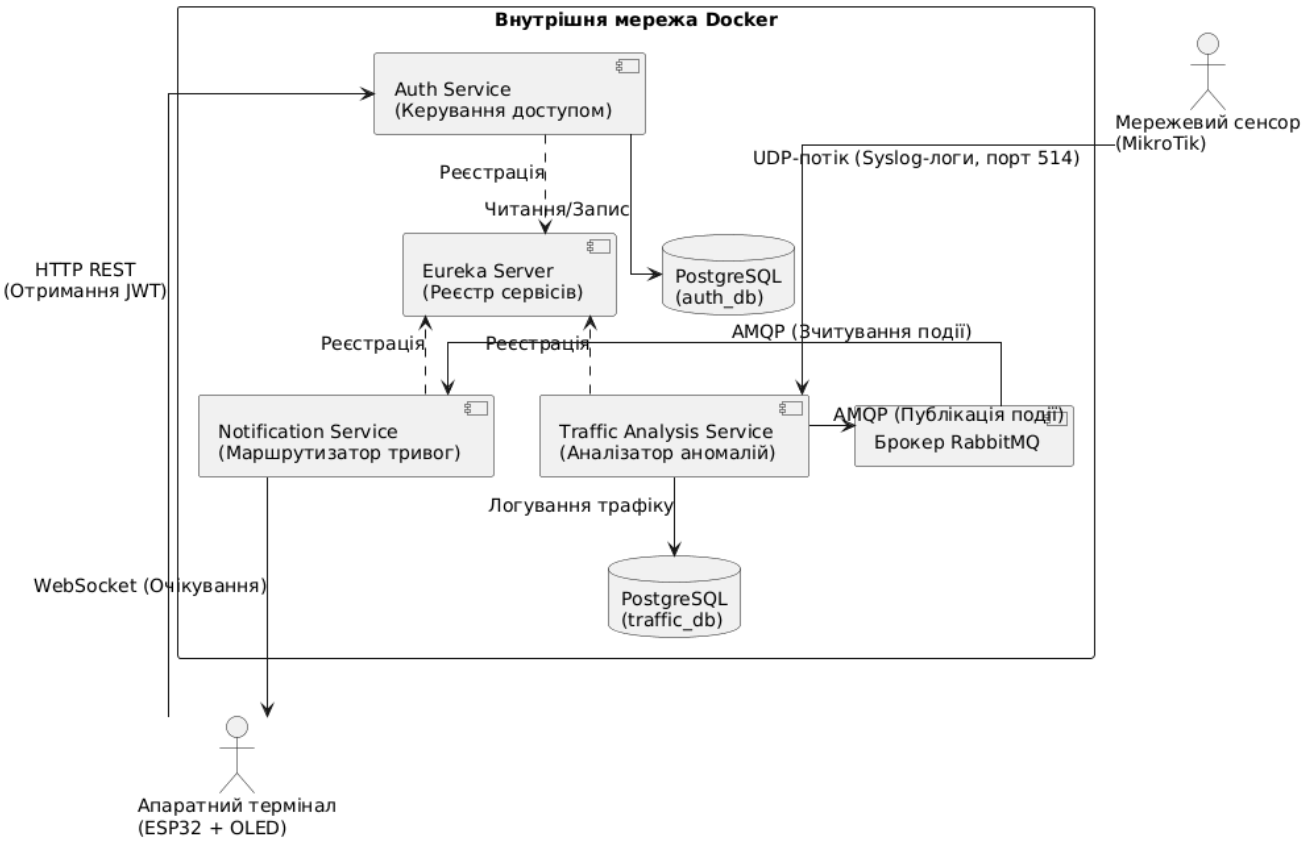


Рисунок 2.5 – Загальна мікросервісна архітектура системи

Сервіс реєстрації та виявлення виконує роль динамічної адресної книги. Усі інші мікросервіси при старті реєструються в ньому, що дозволяє їм знаходити один одного в Docker-середовищі за логічними іменами, а не за статичними IP-адресами. Сервіс авторизації відповідає за безпеку кінцевих точок. Він обробляє запити від клієнтів, перевіряє облікові дані в базі даних автентифікації користувачів, та генерує криптографічно захищені JWT-токени. Сервіс аналізу трафіку є найбільш навантаженим вузлом системи. Він постійно слідкує за портом, очікуючи пакети від маршрутизатора MikroTik. Сервіс розбирає UDP-пакети, проводить аналіз за заданими алгоритмами виявлення аномалій, зберігає статистику в БД та публікує події про інциденти у чергу брокера RabbitMQ. Сервіс сповіщень – це ізольований модуль, який працює як підписник черги RabbitMQ. Його головним призначенням є керування активними WebSocket-з’єднаннями з клієнтами та розсилка їм сповіщень при появі нової аномалії у брокері повідомлень.

2.3.3 Взаємодії компонентів

Зовнішній зловмисник або скомпрометований пристрій усередині мережі починає генерувати шкідливий трафік, після цього маршрутизатор MikroTik формує запис про цю активність і відправляє його у вигляді датаграми на порт сервісу аналізу трафіку. Сервіс аналізу розпаковує датаграму і проганяє метадані через детектори аномалій. Якщо пороги перевищено, подія фіксується у базі даних, а об’єкт відправляється у чергу брокера RabbitMQ. Брокер миттєво передає цю інформацію сервісу нотифікацій, який утримує постійний зв’язок з апаратним індикатором ESP32. Мікроконтролер ESP32, який попередньо авторизувався та відкрив WebSocket, отримує JSON-повідомлення, а мікропрограма ESP32 розбирає JSON і виводить критичну інформацію на OLED-дисплей, сигналізуючи про небезпеку.

Висновки до розділу 2:

У другому розділі було проведено комплексний аналіз та проєктування програмно-апаратної архітектури інтелектуальної системи виявлення аномалій мережевого трафіку. На основі аналізу апаратних засобів для реалізації підсистеми візуальної індикації було обрано мікроконтролер ESP32 із вбудованим OLED-дисплеєм, що забезпечує необхідний баланс між продуктивністю та компактністю. Для організації збору мережевої статистики обрано використання маршрутизатора MikroTik RB941-2nD з підтримкою віддаленого Syslog-експорту логів брандмауера. У ході обґрунтування програмного стеку доведено доцільність використання мови програмування Java та фреймворку Spring Boot для побудови бекенду. Обрано гібридну модель збереження даних. Транзакційна база PostgreSQL для збереження конфігурацій та первинних логів. На завершення сформовано загальну концепцію та розроблено структурні схеми взаємодії, які визначають логіку руху даних від моменту виникнення мережевої аномалії до її відображення на екрані апаратного термінала. Спроєктована архітектура є стійкою до високих навантажень і готова до етапу безпосередньої програмної реалізації.

3 ПРАКТИЧНА РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ АПАРАТНО-ПРОГРАМНОГО КОМПЛЕКСУ ВИЯВЛЕННЯ АНОМАЛІЙ МЕРЕЖЕВОГО ТРАФІКУ

3.1. Вибір та обґрунтування компонентної бази системи

Як основне джерело даних про стан мережевого трафіку було обрано маршрутизатор MikroTik на базі операційної системи RouterOS (рис. 3.1). Обґрунтуванням такого вибору можливості вбудованого брандмауера, який дозволяє налаштовувати правила маршрутизації, сегментувати локальну мережу за допомогою віртуальних точок доступу та фіксувати транзитний трафік. Важливою перевагою обраного обладнання є підтримка протоколу Syslog [21], що дозволяє маршрутизатору в реальному часу експортувати системні журнали у вигляді UDP-пакетів на сервер аналітики без додаткового навантаження на мережевий канал. Живлення маршрутизатора здійснюється зовнішнім акумулятором Qinetiq RQ-80.

Рисунок 3.1 – Маршрутизатор MikroTik з зовнішнім акумулятором

Для реалізації фізичного модуля візуалізації та сповіщення було обрано мікроконтролерну платформу ESP32 з вбудованим OLED-дисплеєм (рис. 3.2). Даний мікроконтролер має вбудований модуль Wi-Fi, що дозволяє підключити його до ізольованої підмережі, створеної маршрутизатором, забезпечуючи тим

самим високий рівень кібербезпеки. ESP32 володіє обчислювальною потужністю з частотою до 240 МГц для підняття власного вебсервера, швидкого прийому HTTP POST-запитів та десеріалізації JSON-об'єктів.

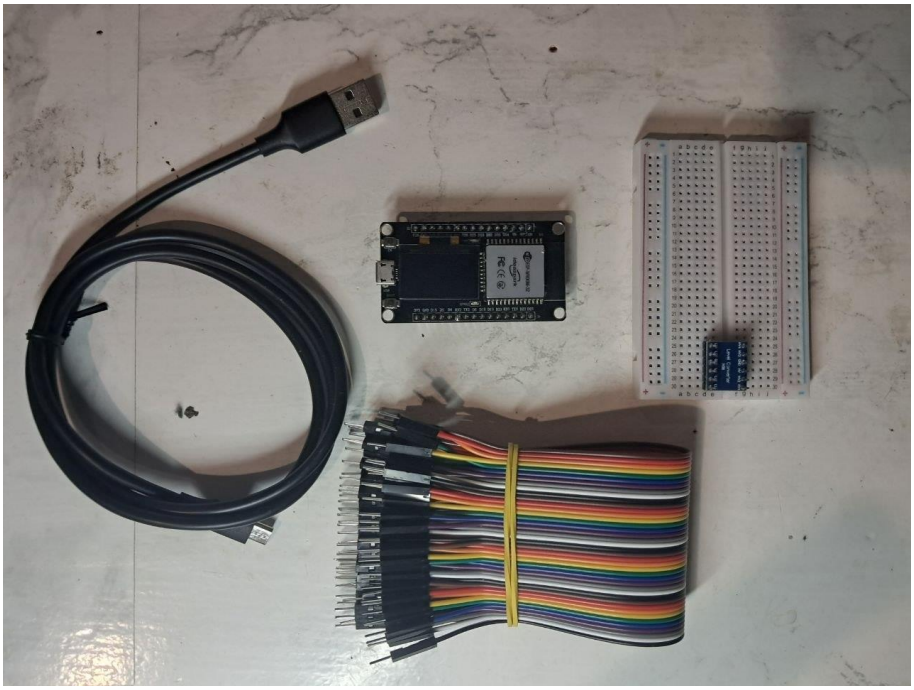


Рисунок 3.2 – Апаратний модуль сповіщень на базі мікроконтролера ESP32 та перетворювач логічних рівнів

Оскільки мікроконтролер ESP32 функціонує на логічному рівні 3,3 В, для забезпечення безпечного сполучення з периферійними пристроями, зовнішніми датчиками або джерелами живлення, що використовують 5-вольтову логіку, до апаратної бази системи включено двонаправлений перетворювач логічних рівнів (рис.3.3).

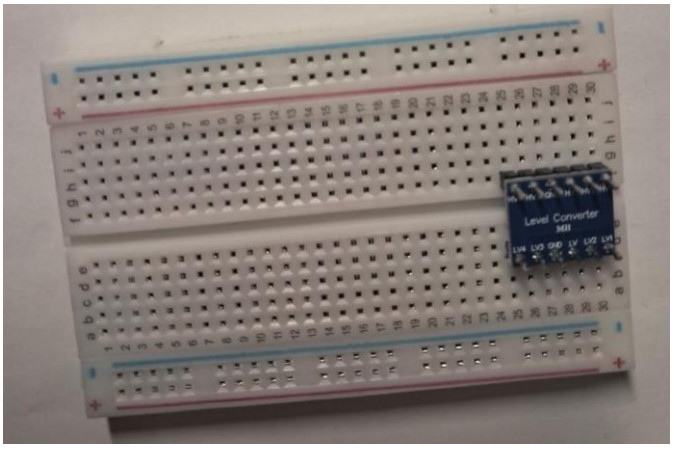


Рисунок 3.3 – Перетворювач логічних рівнів на макетній платі

Використання цього модуля електричного узгодження запобігає ймовірному виходу з ладу портів вводу-виводу мікроконтролера внаслідок перенапруги та гарантує стабільну передачу цифрових сигналів між різними компонентами комплексу. На додаток, наявність перетворювача даватиме змогу масштабувати систему та підключити більше апаратних компонентів без ризику перегорання плати.

3.2 Конструювання та налаштування апаратно-мережевої частини

3.2.1 Розробка та налаштування топології локальної мережі

Для забезпечення безпеки та стабільності роботи апаратно-програмного комплексу та уникнення конфліктів IP-адрес було прийнято рішення про логічне сегментування мережі. Мікроконтролер ESP32, який виконує роль вузла візуального сповіщення, винесено в окрему ізольовану бездротову підмережу (рис. 3.4).

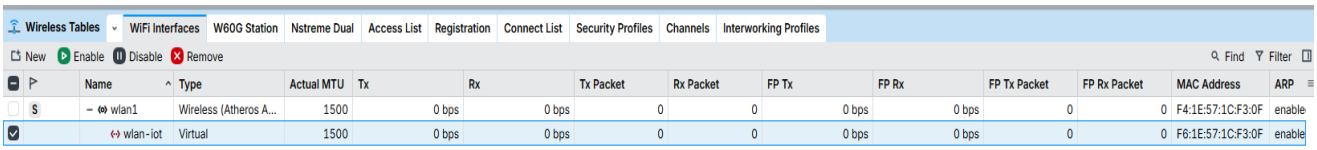


Рисунок 3.4 – Створення віртуальної точки доступу для ізоляції IoT-пристроїв у середовищі WinBox

На маршрутизаторі MikroTik засобами інтерфейсу WinBox було створено віртуальну точку доступу Virtual AP [22] з унікальним ідентифікатором SSID. Такий підхід унеможливлює несанкціонований доступ до плати з основної користувацької мережі та дозволяє застосовувати до неї окремі політики безпеки.

3.2.2 Налаштування маршрутизатора MikroTik та підсистеми логування

Наступним етапом конструювання є конфігурація мережевих служб маршрутизатора. Для забезпечення безперебійного зв’язку між сервером та модулем сповіщень мікроконтролеру необхідно мати незмінну мережеву адресу. З цією метою було налаштовано DHCP-сервер для створеної підмережі та здійснено

статичну прив'язку MAC-адреси мікроконтролера ESP32 до виділеної IP-адреси (рис. 3.5).

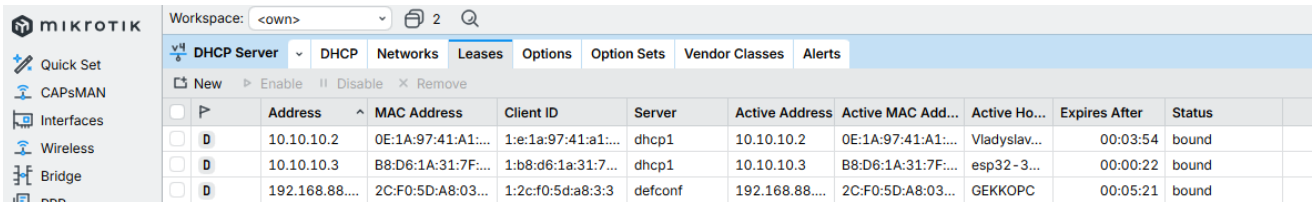


Рисунок 3.5 – Резервування статичної IP-адреси мікроконтролера на DHCP-сервері

Ключовим завданням маршрутизатора у розробленій системі є фіксація мережевих пакетів та їх передача на сервер аналітики. Для того, щоб реалізувати механізм у брандмауері, було створено правило для ланцюжка forward, яке обробляє весь транзитний трафік. Дії правила призначено значення log. Щоб сервер аналітики міг легко відфільтрувати потрібні події від системного сміття, до записів додається унікальний префікс MTIK_TRAFFIC (рис. 3.6).

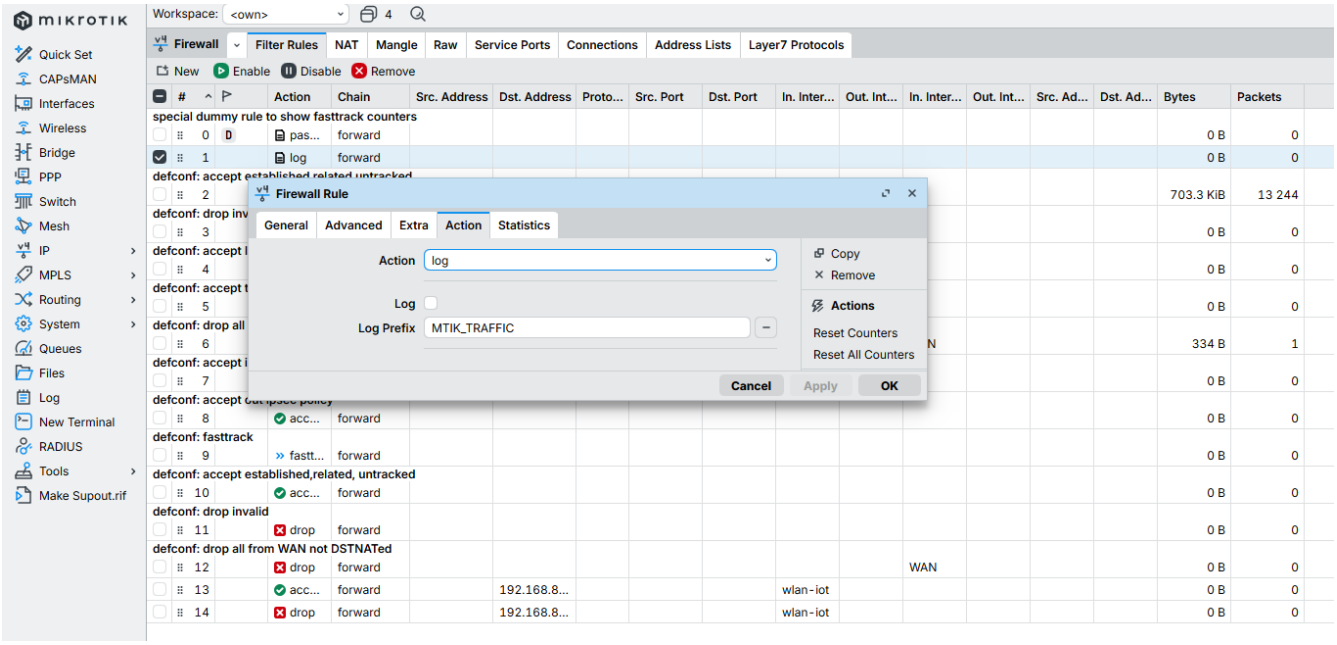


Рисунок 3.6 – Налаштування правила брандмауера для фіксації мережевого трафіку

Для експорту зібраних пакетних даних на сервер, де розгорнуто додаток Spring Boot, було налаштовано підсистему логування Syslog. У налаштуваннях

системи створено нову дію типу remote. У якості цільового призначення вказано IP-адресу комп'ютера-сервера та стандартний порт протоколу Syslog (UDP 514). Після цього створено правило маршрутизації логів, яке перенаправляє всі події з категорії firewall на створений віддалений сервер (рис. 3.7).

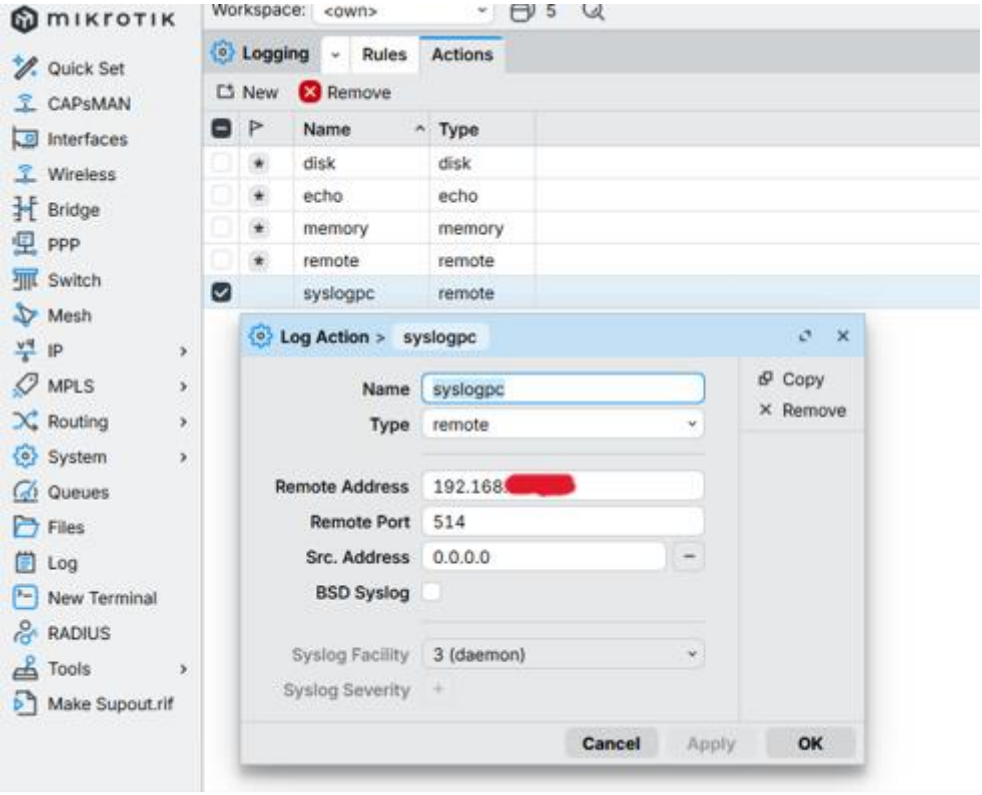


Рисунок 3.7 – Конфігурація віддаленого логування для передачі телеметрії на сервер аналітики

У результаті виконання конфігурацій маршрутизатор MikroTik починає виконувати функцію активного мережевого сенсора. Зібрана телеметрія безперервно та з мінімальними затримками транслюється по протоколу UDP на сервер обробки для подальшого програмного аналізу трафіку та виявлення аномалій у режимі реального часу.

3.2.3 Конструктивно-технічне виконання модуля сповіщень

Конструювання та відладка апаратного прототипу модуля сповіщень здійснювалися за допомогою безпаячної макетної плати. ESP32 було встановлено таким чином, що один ряд його виводів зафіксовано в матриці макетної плати із

залишенням вільного ряду отворів для підключення, тоді як протилежний ряд ніжок конструктивно залишався за межами плати у підвішеному стані. Комутація з виводами, що знаходяться у підвішеному стані, реалізована за допомогою комбінованих Dupont-провідників, утворених шляхом щільного з'єднання між собою дротів типу «мама-мама» та «тато-тато» (рис. 3.8).

Для забезпечення апаратної масштабованості, відмовостійкості та захисту мікроконтролера у схему комплексу інтегровано чотириканальний двоспрямований перетворювач логічних рівнів. Він виконує роль апаратного буфера між низьковольтною логікою мікроконтролера та потенційною зовнішньою промисловою периферією з високовольтною логікою, такою, як виконавчі реле або звукові сирени оповіщення.

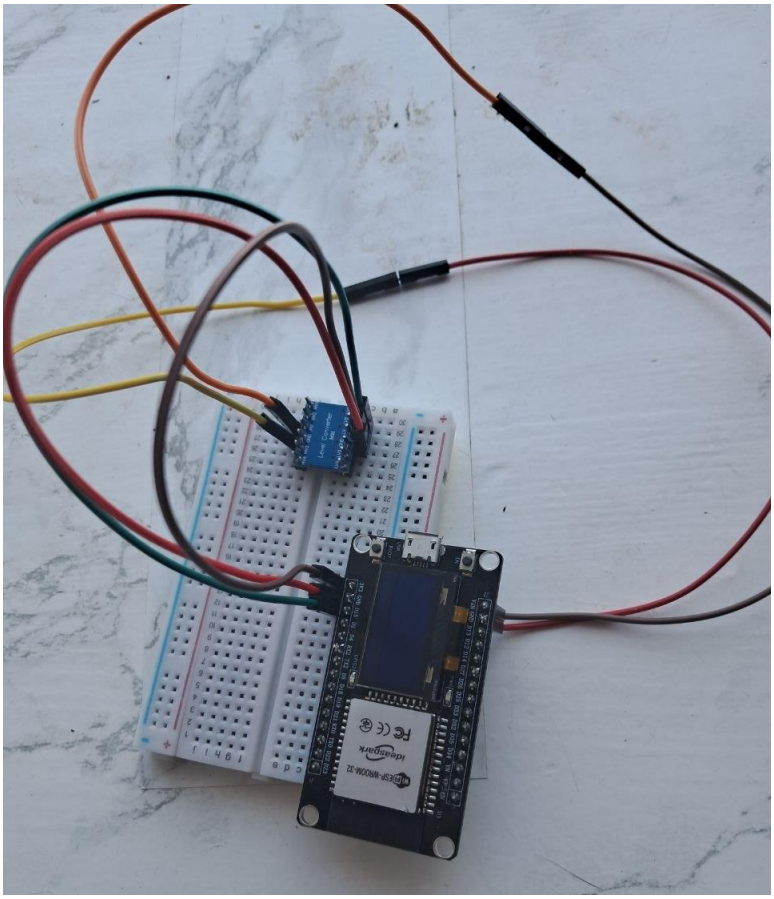


Рисунок 3.8 – Інтегрований апаратний модуль сповіщень на базі плати ESP32 з вбудованим OLED-дисплеєм

Схематичне рішення комутації елементів, базується на забезпеченні спільних опорних напруг для обох контурів двоспрямованого перетворювача логічних рівнів.

Низьковольтна сторона пристрою (LV) та відповідний вивід заземлення (GND) підключені до пінів 3V3 та GND мікроконтролера ESP32, що формують потенціал 3,3 В для інтерфейсної логіки. Високовольтна сторона (HV) разом із власним заземленням під’єднана до виводу VIN плати мікроконтролера, який виступає джерелом напруги 5 В, отриманої через інтерфейс USB. Комутація сигнальних ліній реалізована шляхом трансляції цифрового сигналу з виводу загального призначення (GPIO) ESP32 на низьковольтний вхід LV1.

3.2.4. Підготовка середовища розробки та ініціалізація плати

Останнім етапом налаштування апаратно-мережевої частини є підготовка інтегрованого середовища розробки та первинна ініціалізація мікроконтролера. Для компіляції та завантаження мікропрограми на ESP32 було використано платформу Arduino IDE.

Для успішної комунікації між комп'ютером та мікроконтролером у налаштуваннях середовища обрано відповідну модель плати ESP32 Dev Module та визначено віртуальний COM-порт. Також параметри швидкості передачі даних було встановлено на рівні 115200 бод для забезпечення стабільного прошивання (рис. 3.9).

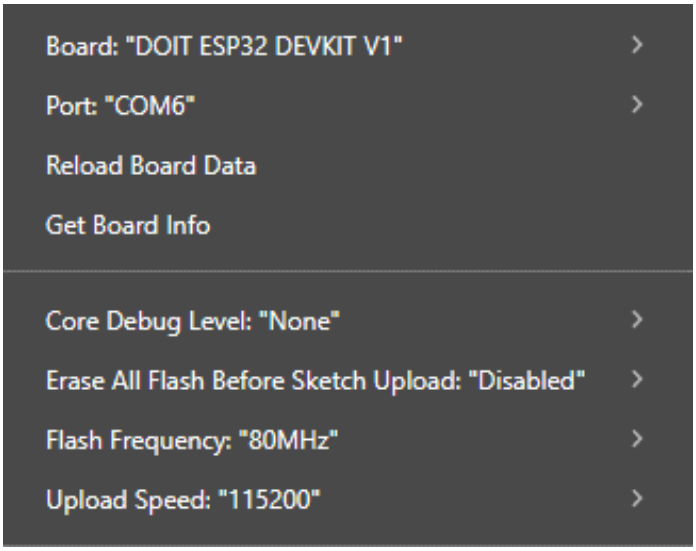


Рисунок 3.9 – Налаштування параметрів плати та COM-порту в середовищі Arduino IDE

Успішна конфігурація середовища розробки та завантаження базової мікропрограми дозволила мікроконтролеру ініціалізувати підключення до створеної віртуальної мережі Wi-Fi. Мікроконтролер отримав статичну IP-адресу від маршрутизатора (рис. 3.10), що підтверджує готовність апаратної частини до інтеграції з програмними мікросервісами для прийому запитів з даними про аномалії.

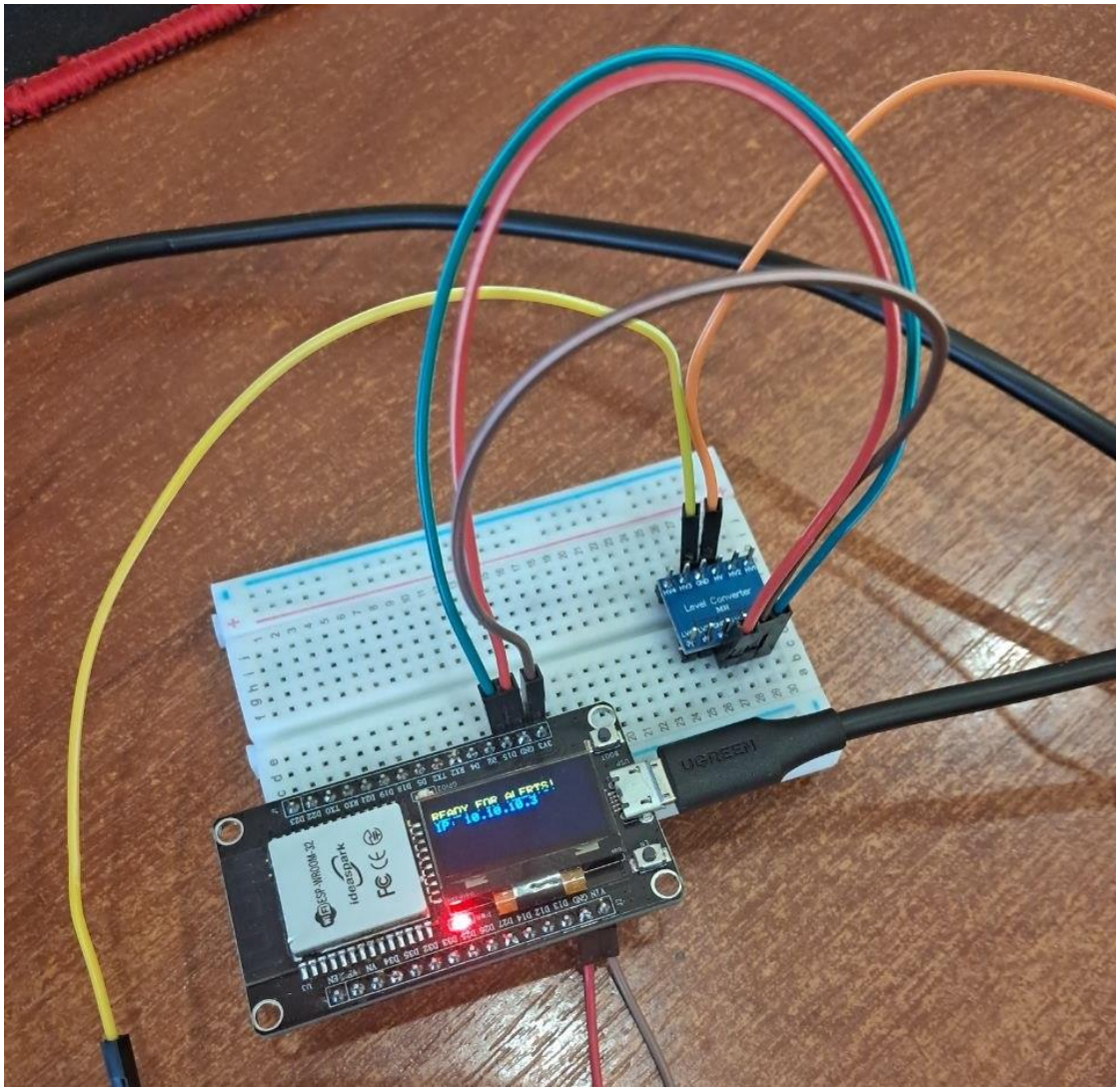


Рисунок 3.10 – Мікроконтролер підключений до IP-адреси маршрутизатора

Поточна збірка апаратного модуля є повністю функціональним прототипом, який дозволяє відпрацьовувати логіку взаємодії між сервером та кінцевим IoT-пристроєм. Використання прямого USB-підключення до комп'ютера на етапі тестування забезпечує як стабільне живлення платформи напругою 5 В, так і зручний канал для завантаження мікропрограми.

3.3. Програмна реалізація мікросервісної архітектури та IoT-модуля

3.3.1. Розробка архітектури програмного забезпечення

Основою взаємодії є сервіс виявлення на базі Spring Cloud Netflix Eureka. Він діє як динамічний реєстр, де кожен мікросервіс під час запуску реєструє свою IP-адресу та порт. Це дозволяє іншим компонентам звертатися один до одного за логічними іменами, наприклад, TRAFFIC-SERVICE, не прив'язуючись до статичних мережевих адрес (рис. 3.11).

Безпека доступу до API системи для перегляду звітів чи налаштування вебхуків забезпечується модулем Auth-Service. Він генерує та валідує JWT-токени, гарантуючи, що лише авторизовані користувачі або внутрішні сервіси мають доступ до захищених кінцевих точок.

Instances currently registered with Eureka

Application	AMIs	Availability Zones	Status
AUTH-SERVICE	n/a (1)	(1)	UP (1) - host.docker.internal:auth-service:8081
NOTIFICATION-SERVICE	n/a (1)	(1)	UP (1) - host.docker.internal:notification-service:8083
TRAFFIC-SERVICE	n/a (1)	(1)	UP (1) - host.docker.internal:traffic-service:8082

Рисунок 3.11 – Панель керування сервісом виявлення Eureka із зареєстрованими модулями системи

Головним обчислювальним ядром є TrafficAnalysisService. Його завдання полягає у прийомі сирих даних, їх парсингу, перевірці на відповідність правилам безпеки. Також він має зберігати результати у базах даних Postgres для структурованих звітів та Elasticsearch для швидкого пошуку. У разі виявлення аномалії TrafficAnalysisService виступає в ролі «Видавця» (англ. Publisher) і відправляє подію про загрозу до брокера повідомлень RabbitMQ. З іншого боку черги знаходиться NotificationService – «Підписник» (англ. Subscriber). Він асинхронно зчитує ці події і паралельно транслює їх на клієнтський вебфронтенд через WebSockets та формує HTTP POST-запит за допомогою Webhook для відправки на апаратний модуль ESP32.

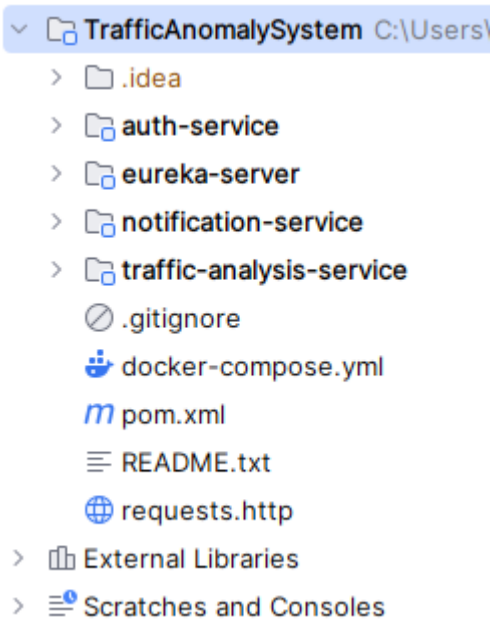


Рисунок 3.12 – Структура мікросервісного проєкту в середовищі розробки

Підхід проєктування програмної частини системи у вигляді окремих мікросервісів забезпечує високу відмовостійкість. Падіння одного з модулів не призводить до зупинки процесу збору та аналізу мережевого трафіку (рис. 3.12).

3.3.2. Програмна реалізація детекторів аномалій

Процес виявлення аномалій розпочинається з отримання телеметрії від маршрутизатора MikroTik. Для цього в ядрі TrafficAnalysisService реалізовано клас MikrotikSyslogServer, який ініціалізує безперервне прослуховування UDP-порту у окремому обчислювальному потоці (рис. 3.13). Оскільки лог маршрутизатора надходить у вигляді неструктурованого тексту, система використовує механізм регулярних виразів (Regex) для вилучення ключових метрик, як IP-адреси джерела, IP-адреси та порту призначення, а також розміру пакета в байтах. Після успішного парсингу ці розрізнені параметри миттєво об’єднуються у єдиний структурований об'єкт програмного пакета. Такий підхід дозволяє перетворювати сирі текстові логи на зручну для обробки інформацію без значних накладних витрат процесорного часу.

```java
private final Pattern ipPortPattern = Pattern.compile( regex: "(\\d+\\.\\d+\\.\\d+\\.\\d+):(\\d+)->(\\d+\\.\\d+\\.\\d+\\.\\d+):(\\d+)");
private final Pattern lenPattern = Pattern.compile( regex: "len (\\d+)");  1 usage

@PostConstruct  Vladyslav Yevlampiev
public void startListening() {
    new Thread(() -> {
        try (DatagramSocket socket = new DatagramSocket( port: 514)) {
            System.out.println(">>> [UDP SERVER] Listening and Parsing MikroTik logs on port 514...");
            byte[] buffer = new byte[2048];

            while (true) {
                DatagramPacket packet = new DatagramPacket(buffer, buffer.length);
                socket.receive(packet);
                String message = new String(packet.getData(),  offset: 0, packet.getLength());

                if (message.contains("MTIK_TRAFFIC")) {
                    Matcher ipMatcher = ipPortPattern.matcher(message);
                    Matcher lenMatcher = lenPattern.matcher(message);

                    if (ipMatcher.find() && lenMatcher.find()) {
                        String srcIp = ipMatcher.group(1);
                        String destIp = ipMatcher.group(3);
                        int destPort = Integer.parseInt(ipMatcher.group(4));
                        int sizeBytes = Integer.parseInt(lenMatcher.group(1));

                        TrafficPacket trafficPacket = new TrafficPacket(srcIp, destIp, sizeBytes, destPort);

                        trafficService.processRealPacket(trafficPacket);
                    }
                }
            }
        } catch (Exception e) {
            System.err.println(">>> [UDP SERVER] Error: " + e.getMessage());
```

Рисунок 3.13 – Програмна реалізація UDP-сервера та парсингу логів маршрутизатора за допомогою регулярних виразів

Після парсингу сформований об'єкт пакета передається до методу processRealPacket класу TrafficService. Тут реалізовано логіку мультивекторної детекції загроз, яка працює в режимі реального часу і включає три основні правила (рис. 3.14):

– виявлення DDoS-атак: система використовує хеш-таблицю ConcurrentHashMap для підрахунку кількості отриманих пакетів з кожної унікальної IP-адреси. Якщо кількість пакетів від одного джерела перевищує встановлений поріг протягом однієї секунди, система генерує попередження про аномальну інтенсивність трафіку;

– виявлення сканування портів: детектор аналізує порт призначення кожного пакета. Спроба доступу до критичних або закритих портів класифікується як потенційне порушення безпеки та спроба несанкціонованого втручання;

– виявлення аномального об'єму: Перевіряється розмір кожного окремого транзитного пакета. Якщо об'єм корисного навантаження перевищує заданий ліміт, це свідчить про передачу важких файлів або нестандартний трафік, що також викликає спрацювання тригера безпеки.

```
public void processRealPacket(TrafficPacket packet) {  1 usage   Vladyslav Yevlampiev
    long now = System.currentTimeMillis();

    if (now - lastResetTime > 1000) {
        packetCounts.clear();
        lastResetTime = now;
    }

    int count = packetCounts.getOrDefault(packet.getSourceIp(),  defaultValue: 0) + 1;
    packetCounts.put(packet.getSourceIp(), count);

    if (count == DDOS_THRESHOLD) {
        triggerAlert(packet,  type: "DDoS Attack",  description: "High packet rate detected: " + count + " pkts/sec");
    }

    if (packet.getDestPort() == 22) {
        triggerAlert(packet,  type: "Security Breach",  description: "Attempted access to closed port 22 (SSH)");
    }

    if (packet.getSizeBytes() > 1000) {
        triggerAlert(packet,  type: "Real High Volume",  description: "Heavy packet detected! Size: " + packet.getSizeBytes() + " bytes");
    }
}
```

Рисунок 3.14 – Алгоритм виявлення мережевих аномалій на базі правил

Для уникнення дублювання коду, процес реєстрації інциденту винесено в окремий метод triggerAlert. Він відповідає за формування об'єкта AnomalyReport,його каскадне збереження у реляційну базу PostgreSQL, реплікацію даних в Elasticsearch та остаточне формування події AnomalyEvent для відправки у брокер RabbitMQ.

3.3.3. Реалізація прошивки мікроконтролера

Програмне забезпечення для мікроконтролера ESP32 розроблено мовою C++ з використанням програмного середовища Arduino IDE. Прошивка має забезпечувати стабільне підключення до бездротової мережі, розгортання локального вебсервера та швидке оброблення вхідних даних про загрози для їх візуалізації.

Під час ініціалізації пристрою виконується підключення OLED-дисплея через I2C-інтерфейс за допомогою бібліотек Adafruit_GFX та Adafruit_SSD1306.

Після цього мікроконтролер встановлює з'єднання з ізольованою точкою доступу маршрутизатора MikroTik та запускає об'єкт класу WebServer на порту. Для прослуховування вхідних сповіщень зареєстровано маршрут alert, який обробляє HTTP POST-запити. Ключовим елементом прошивки є функція обробки сповіщень handleAlert. Оскільки NotificationService відправляє дані у форматі JSON, для їх розбору на мікроконтролері використовується бібліотека ArduinoJson. Спочатку сервер зчитує тіло запиту, після чого виділяє блок пам'яті для парсингу отриманого рядка (рис. 3.15).

```
43  void handleAlert() {
44    String jsonBody = server.arg("plain");
45
46    if (jsonBody.length() == 0 && server.args() > 0) {
47      jsonBody = server.arg(0);
48    }
49
50    Serial.println("Received Body: " + jsonBody);
51
52    if (jsonBody.length() == 0) {
53      server.send(400, "text/plain", "Body is completely empty");
54      return;
55    }
56
57    DynamicJsonDocument doc(1024);
58    DeserializationError error = deserializeJson(doc, jsonBody);
59
60    if (error) {
61      Serial.print("JSON Parse Error: ");
62      Serial.println(error.c_str());
63      server.send(400, "text/plain", "Invalid JSON format");
64      return;
65    }
66
67    String anomalyType = doc["anomalyType"].as<String>();
68    String sourceIp = doc["sourceIp"].as<String>();
69    int sizeBytes = doc["sizeBytes"].as<int>();
70
71    updateScreen(anomalyType, sourceIp, sizeBytes);
72
73    server.send(200, "text/plain", "Alert Successfully Displayed!");
74  }
```

Рисунок 3.15 – Програмна реалізація прийому та розбору JSON-об'єктів на базі мікроконтролера ESP32

Після успішного розбору JSON-структури мікроконтролер вилучає значення типу аномалії, IP-адреси джерела та розміру пакета. Ці дані передаються до функції оновлення дисплея updateScreen, яка очищає попередній вміст екрана та виводить нове критичне сповіщення. Наприкінці обробки ESP32 повертає серверу HTTP-статус 200 OK та підтвердження успішного отримання та візуалізації алерта.

3.3.4. Опис інтерфейсів АПЗ

Коли модуль аналізу трафіку виявляє загрозу, він публікує повідомлення у RabbitMQ (рис. 3.16). Після цього брокер автоматично маршрутизує його до підписаного модуля сповіщень. Завдяки такому інтерфейсу гарантується збереження алертів навіть у випадку тимчасових мережевих збоїв.

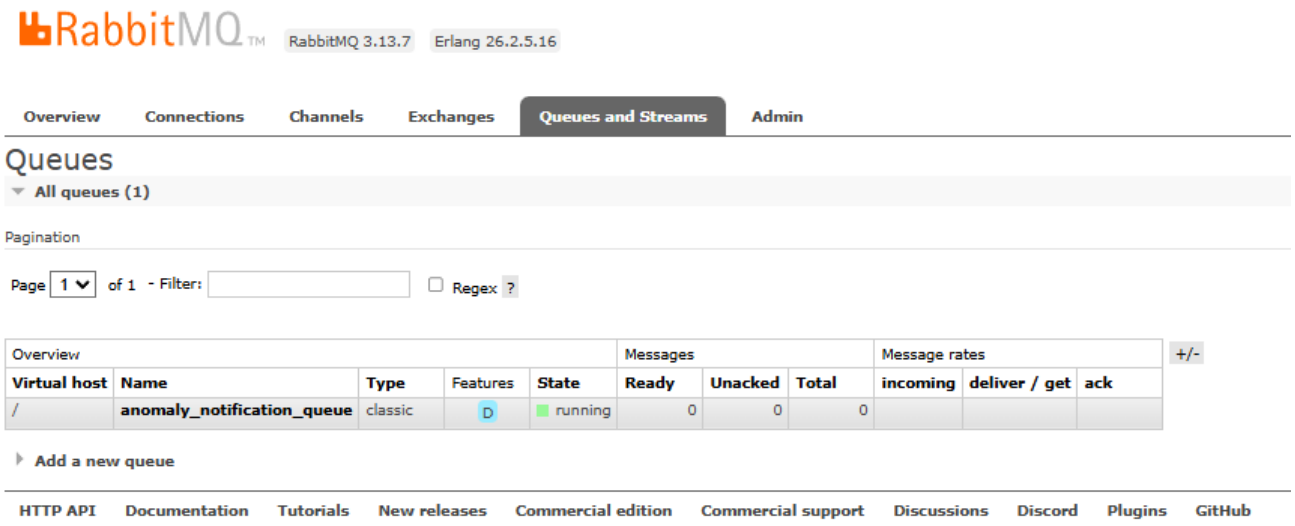


Рисунок 3.16 – Внутрішній інтерфейс асинхронного обміну повідомленнями на базі RabbitMQ

Користувацький програмний інтерфейс АПЗ представлено вебпанеллю оператора, яка інтегрована у NotificationService. На відміну від класичних вебсторінок, інтерфейс оператора працює в режимі реального часу, що досягається завдяки використанню протоколу WebSockets.

Як тільки NotificationService отримує повідомлення з RabbitMQ, він транслює його через відкритий WebSocket-канал у браузер клієнта. Фронтенд-частина, реалізована за допомогою HTML, CSS та JavaScript, миттєво перехоплює цю подію та динамічно додає новий запис про виявлену аномалію до таблиці інцидентів на екрані оператора (рис. 3.17).

Рисунок 3.17 – Інтерфейс вебпанелі оператора для моніторингу мережевих загроз

Така архітектура інтерфейсів дозволяє адміністратору мережі отримувати миттєві візуальні сповіщення як на апаратному IoT-модулі, так і у розгорнутій вебпанелі.

3.4. Експериментальні дослідження, тестування та настанова користувача

3.4.1. Інтеграційне тестування та налагоджування роботи системи

Для перевірки коректності функціонування розробленого АПЗ було обрано методику системного тестування шляхом імітаційного моделювання мережевих загроз у контрольованому ізольованому середовищі. Метою тестування є підтвердження здатності програмного забезпечення безпомилково ідентифікувати аномалії в реальному трафіку та оцінка швидкості реакції апаратного модуля сповіщень. Використовуваний програмний засіб для тестування Nmap [23].

Сценарій 1. Імітація DDoS-атаки. Метою тесту була перевірка механізму виявлення аномальної інтенсивності запитів. З комп'ютера-хоста за допомогою утиліти Nping було згенеровано серію швидких послідовних TCP-пакетів зі швидкістю 100 пакетів на секунду (рис. 3.18).

```
C:\Users\gekko>ping 10.10.10.3 -l 1200

Pinging 10.10.10.3 with 1200 bytes of data:
Reply from 10.10.10.3: bytes=1200 time=107ms TTL=63
Reply from 10.10.10.3: bytes=1200 time=41ms TTL=63
Reply from 10.10.10.3: bytes=1200 time=132ms TTL=63
Reply from 10.10.10.3: bytes=1200 time=41ms TTL=63

Ping statistics for 10.10.10.3:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 41ms, Maximum = 132ms, Average = 80ms
```

Рисунок 3.18 – Імітація DDoS-атаки за допомогою Nping

Система успішно підрахувала кількість пакетів з IP-адреси джерела та, зафіксувавши перевищення встановленого порогу в 50 пакетів за секунду, ідентифікувала активність як загрозу. Апаратний модуль миттєво відреагував на подію (рис. 3.19).

Рисунок 3.19 – Фіксація системою DDoS-атаки та виведення тривоги на екран

Сценарій 2. Спроба несанкціонованого доступу. Метою експерименту була перевірка здатності системи реагувати на цілеспрямоване сканування критичних портів. Імітація полягала у використанні сканера Nmap для перевірки стану портів цільового вузла (рис. 3.20).

```
C:\Windows\System32>nmap -p 22,80,443,3306 10.10.10.3
Starting Nmap 7.99 ( https://nmap.org ) at 2026-06-09 12:24 +0300
Nmap scan report for 10.10.10.3
Host is up (0.056s latency).

PORT     STATE  SERVICE
22/tcp   closed ssh
80/tcp   open   http
443/tcp  closed https
3306/tcp closed mysql

Nmap done: 1 IP address (1 host up) scanned in 1.17 seconds
```

Рисунок 3.20 – Процес сканування мережевих портів за допомогою утиліти Nmap

Алгоритм парсингу логів успішно виділив цільовий порт із Syslog-повідомлення маршрутизатора, а детектор згенерував попередження про порушення політики доступу. На OLED-дисплеї відобразилася інформація про інцидент (рис. 3.21).

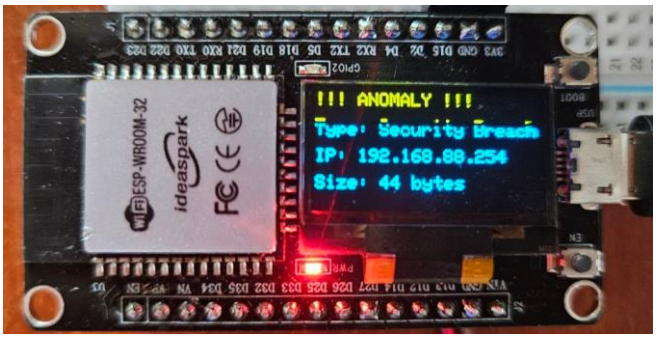


Рисунок 3.21 – Апаратна індикація спроби несанкціонованого доступу

Сценарій 3. Передача пакета аномального об'єму. Для перевірки механізму визначення аномального об'єму корисного навантаження було проведено вдосконалення парсера системи для підтримки безпортових протоколів ICMP. Завдяки цьому з'явилася можливість імітувати атаку за допомогою стандартної утиліти Ping, відправивши пакет із корисним навантаженням у 1200 байт (рис. 3.22) Було використано команду ping 10.10.10.3 -l 1200.

```
RCVD (7.7830s) TCP 10.10.10.3:80 > 192.168.88.254:19096 SA ttl=63 id=1521 iplen=44  seq=2355655808 win=5760 <mss 1436>
SENT (7.7830s) TCP 192.168.88.254:19096 > 10.10.10.3:80 S ttl=64 id=31878 iplen=40  seq=2310611908 win=1480
RCVD (7.7970s) TCP 10.10.10.3:80 > 192.168.88.254:19096 SA ttl=63 id=1522 iplen=44  seq=2355655808 win=5760 <mss 1436>
SENT (7.7970s) TCP 192.168.88.254:19096 > 10.10.10.3:80 S ttl=64 id=31878 iplen=40  seq=2310611908 win=1480
SENT (7.8140s) TCP 192.168.88.254:19096 > 10.10.10.3:80 S ttl=64 id=31878 iplen=40  seq=2310611908 win=1480
SENT (7.8270s) TCP 192.168.88.254:19096 > 10.10.10.3:80 S ttl=64 id=31878 iplen=40  seq=2310611908 win=1480
RCVD (7.8390s) TCP 10.10.10.3:80 > 192.168.88.254:19096 SA ttl=63 id=1525 iplen=44  seq=2355655808 win=5760 <mss 1436>
SENT (7.8390s) TCP 192.168.88.254:19096 > 10.10.10.3:80 S ttl=64 id=31878 iplen=40  seq=2310611908 win=1480
RCVD (7.8510s) TCP 10.10.10.3:80 > 192.168.88.254:19096 SA ttl=63 id=1526 iplen=44  seq=2355655808 win=5760 <mss 1436>
SENT (7.8510s) TCP 192.168.88.254:19096 > 10.10.10.3:80 S ttl=64 id=31878 iplen=40  seq=2310611908 win=1480
RCVD (7.8670s) TCP 10.10.10.3:80 > 192.168.88.254:19096 SA ttl=63 id=1527 iplen=44  seq=2355655808 win=5760 <mss 1436>
SENT (7.8670s) TCP 192.168.88.254:19096 > 10.10.10.3:80 S ttl=64 id=31878 iplen=40  seq=2310611908 win=1480
RCVD (7.8800s) TCP 10.10.10.3:80 > 192.168.88.254:19096 SA ttl=63 id=1528 iplen=44  seq=2355655808 win=5760 <mss 1436>

Max rtt: 28.000ms | Min rtt: 12.000ms | Avg rtt: 12.278ms
Raw packets sent: 500 (27.000KB) | Rcvd: 479 (22.034KB) | Lost: 21 (4.20%)
Nping done: 1 IP address pinged in 7.88 seconds
```

Рисунок 3.22 – Відправка аномально великого ICMP-пакета

Система класифікувала такий об'єм як відхилення від норми. ESP32 вивів відповідне повідомлення про перехоплення пакета розміром понад 1000 байт (рис. 3.23).

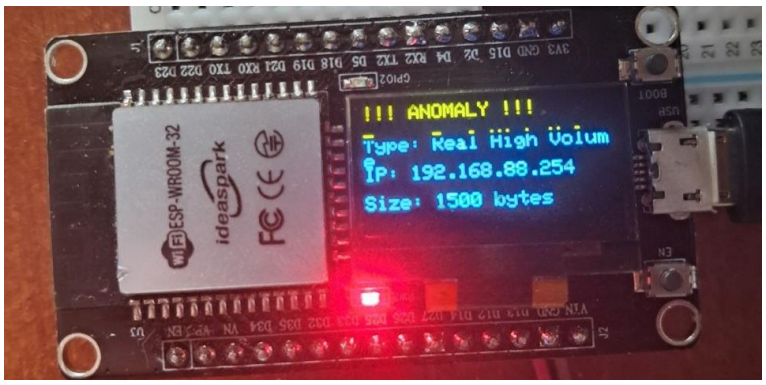


Рисунок 3.23 – Фіксація аномалії розміру пакета на мікроконтролері

Усі згенеровані під час тестування аномалії не лише відображалися на апаратному модулі, а й паралельно маршрутизувалися брокером RabbitMQ до вебпанелі оператора. Завдяки технології WebSockets адміністратор міг спостерігати хронологію багатовекторних атак у браузері в режимі реального часу (рис. 3.24).

Рисунок 3.24 – Відображення перехоплених інцидентів у вебпанелі оператора

Проведені експериментальні дослідження підтвердили високу надійність розробленої системи. Усі три тестові сценарії були успішно перехоплені програмними детекторами. Затримка реакції від моменту генерації трафіку до індикації становила менше 200 мілісекунд, що повністю доводить ефективність застосування мікросервісного підходу.

3.4.2 Настанова користувача

Запуск серверної інфраструктури. Перед початком роботи оператор повинен переконатися, що запущені бази даних Postgres, Elasticsearch та брокер RabbitMQ. Після цього в інтегрованому середовищі розгортаються мікросервіси системи. Першим обов'язково запускається EurekaServer для маршрутизації, після чого запускаються AuthService, TrafficAnalysisService та NotificationService (рис. 3.25).

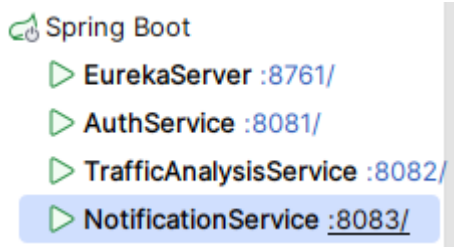


Рисунок 3.25 – Запуск та ініціалізація серверної мікросервісної архітектури

Ініціалізація апаратного модуля сповіщень. Мікроконтролер ESP32 необхідно підключити до джерела живлення. Протягом перших секунд пристрій автоматично підключається до виділеної бездротової мережі маршрутизатора MikroTik. Успішна ініціалізація підтверджується появою на OLED-дисплеї повідомлення про готовність приймати сповіщення та виділеної IP-адреси пристрою.

Для розширеного моніторингу оператор відкриває веббраузер та переходить за адресою локального фронтенд-сервера. Інтерфейс автоматично встановлює WebSocket-з'єднання з бекендом. У разі виникнення загрози, на екрані миттєво з'являється новий запис із зазначенням типу атаки, IP-адреси зловмисника, цільового порту та точного часу інциденту, що дозволяє оператору оперативно заблокувати підозрілу адресу на маршрутизаторі.

Висновки до розділу 3

У третьому розділі кваліфікаційної роботи виконано повний цикл практичної реалізації, налаштування та тестування апаратно-програмного комплексу для виявлення аномалій мережевого трафіку. Процес конструювання системи охопив

усі рівні розробки, від фізичного підключення електронних компонентів та конфігурації мережевого обладнання до написання мікросервісного бекенду та прошивки мікроконтролера.

Обґрунтовано вибір та налаштовано апаратну базу комплексу. Використання маршрутизатора MikroTik дозволило реалізувати апаратну ізоляцію IoT-сегмента шляхом створення віртуальної точки доступу та налаштувати безперервний експорт журналів транзитного трафіку за протоколом Syslog. В якості кінцевого вузла сповіщень успішно ініціалізовано мікроконтролер ESP32 з OLED-дисплеєм SSD1306. Конструктивне виконання вузла гарантує його автономність та мінімальне енергоспоживання, а шина I2C гарантує надійність обміну даними.

Програмну частину системи реалізовано на базі мікросервісної архітектури з використанням мови Java 21 та фреймворку Spring Boot. Такий підхід забезпечив високу модульність рішення. Сервіси авторизації, аналізу трафіку та сповіщень функціонують незалежно один від одного, координуючись через сервер виявлення Eureka. Використання брокера повідомлень RabbitMQ для асинхронної комунікації між модулями дозволило розв'язати проблему можливих мережевих затримок і гарантувати збереження алертів у разі тимчасової недоступності мікроконтролера.

Для зберігання даних розгорнуто гібридну модель баз даних. Було використано PostgreSQL для збереження структурованих звітів та конфігурацій, а також пошуковий рушій Elasticsearch для аналізу великих масивів логів трафіку. Візуалізація даних для адміністратора реалізована через вебпанель, яка оновлюється в режимі реального часу завдяки технології WebSockets.

У рамках експериментального дослідження програмно реалізовано та протестовано алгоритми детекції трьох типів багатовекторних загроз. DDoS-атака, сканування цільових портів шляхом звернення до закритих портів та передачі пакетів великого об'єму. Результати тестування у контрольованому середовищі підтвердили безпомилкову роботу парсера логів та детектора. Система продемонструвала високу швидкодію із затримкою реакції менше 200 мілісекунд від моменту генерації трафіку до індикації на екрані ESP32 та відсутність хибних спрацювань.

ВИСНОВКИ

У кваліфікаційній бакалаврській роботі вирішено актуальну мету, яка полягає у розробці та тестуванні програмної системи на базі мікросервісної архітектури для вчасного знаходження аномалій у мережевому трафіку та сповіщення користувача у вигляді фізичної індикації станів. Проєктування та програмно-апаратна реалізація комплексу дозволили створити гнучке, масштабоване та відмовостійке рішення для автоматизованого моніторингу мережевого периметра в режимі реального часу.

У ході виконання роботи відповідно до поставленої мети було повністю розв'язано такі задачі:

– проаналізовано сучасний стан безпеки мережевих архітектур та детально досліджено існуючі методи виявлення аномалій у трафіку; теоретичний аналіз дозволив чітко розмежувати обмеження сигнатурного пошуку та обґрунтувати переваги детермінованого поведінкового аналізу відхилень від порогових значень;

– обґрунтовано вибір інженерних підходів і зібрано високонавантажений технологічний стек для забезпечення максимальної надійності та гнучкості роботи системи; як апаратну основу обрано маршрутизатор MikroTik та мікроконтролер ESP32, а програмний комплекс розгорнуто на базі сучасної платформи Java 21 та фреймворку Spring Boot;

– сформульовано вичерпну специфікацію функціональних і нефункциональних вимог до розроблюваного програмно-апаратного забезпечення, що дозволило чітко визначити ключові експлуатаційні характеристики системи, такі як механізми захоплення даних, правила детекції аномалій, швидкість реакції та криптографічний захист кінцевих точок;

– спроєктовано розподілену мікросервісну архітектуру системи та детально продумано логіку взаємодії її основних автономних модулів автоматизації, збору статистики і сповіщень; інтеграція оркестратора Eureka Server та асинхронного брокера повідомлень RabbitMQ дозволила повністю усунути ризики взаємного блокування сервісів під час пікових навантажень;

– реалізовано програмний засіб для ефективної роботи з мережевими метаданими, у якому успішно впроваджено концепцію гібридного зберігання даних (Polyglot Persistence); транзакційне реляційне сховище PostgreSQL забезпечує надійність збереження профілів користувачів та операційних логів, тоді як документоорієнтований рушій Elasticsearch гарантує миттєву індексацію та швидкий повнотекстовий пошук історичних записів про інциденти кібербезпеки;

– впроваджено інтелектуальну систему миттєвих сповіщень через WebSockets для динамічного оновлення інтерактивного вебдашборду оператора та гнучкий асинхронний механізм Webhooks для трансляції тривожних сигналів безпеки на віддалені IoT-компоненти;

– протестовано створений програмний комплекс в умовах імітації шкідливого мережевого трафіку шляхом генерації DDoS-флуду, сканування портів за допомогою Nmap/Nping та передачі аномальних об'ємів даних; експеримент повністю підтвердив технічну безпомилковість розроблених алгоритмів детекції, стабільність роботи регулярних виразів парсера та високу швидкість реакції апаратного модуля.

Практичне значення отриманих результатів полягає у створенні повністю працездатного апаратно-програмного комплексу, який може бути впроваджений системними адміністраторами для моніторингу локальних мереж. Використання ізольованого апаратного модуля разом із вебпанеллю дозволяє миттєво виявляти кіберзагрози та оперативно реагувати на інциденти інформаційної безпеки, зводячи до мінімуму ризики компрометації мережі.

Основні положення та результати кваліфікаційної роботи доповідались та були опубліковані у збірнику матеріалів XVII Міжнародної науково-практичної конференції «Free and Open Source Software» : патерни оркестрування у мультиагентних системах аналізу мережевого трафіку» (м. Харків, 11–12 лютого, 2026).

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Бурлаченко І. С., Євлампієв В. Ю. Патерни оркестрування у мультиагентних системах аналізу мережевого трафіку. FOSS – 2026 : тези доп. XVII Міжн. наук.-практ. конф. Харків, 11–12 лют. 2026 р. Харків : Харківський нац. економ. ун-т ім. Семена Кузнеця, 2026. С. 135–137. URL: https://foss.kn-it.info/uploads/foss-2026-theses.pdf (дата звернення: 17.12.2025).

2. Про основні засади забезпечення кібербезпеки України : Закон України від 19 жовт. 2025 р. № 2163-VIII. URL: https://zakon.rada.gov.ua/laws/show/2163-19#Text (дата звернення: 17.12.2025).

3. ДСТУ 3396.1-96. Захист інформації. Технічний захист інформації. Порядок проведення робіт. Київ : Держстандарт України, 1997. 12 с. URL: https://online.budstandart.com/ua/catalog/doc-page.html?id_doc=69173 (дата звернення: 27.12.2025).

4. Yang D., Liu Z., Wei S. Interactive Learning for Network Anomaly Monitoring and Detection with Human Guidance in the Loop. Sensors. 2023. Vol. 23. № 18. P. 7803. DOI: 10.3390/s23187803.

5. What is Elasticsearch? URL: https://www.elastic.co/what-is/elasticsearch (Last accessed: 07.01.2026).

6. Про захист інформації в інформаційно-комунікаційних системах : Закон України від 05 лип. 1994 р. № 80/94-ВР. URL: https://zakon.rada.gov.ua/laws/show/80/94-%D0%B2%D1%80#Text (дата звернення: 17.01.2026).

7. ДСТУ ISO/IEC 27001:2023 (ISO/IEC 27001:2022, IDT). Інформаційна безпека, кібербезпека та захист конфіденційності. Системи керування інформаційною безпекою. Вимоги. Київ : ДП «УкрНДНЦ», 2023. URL: https://online.budstandart.com/ua/catalog/doc-page.html?id_doc=104398 (дата звернення: 28.01.2026).

8. Kasongo S. M., Sun Y. A deep learning method with wrapper based feature extraction for wireless intrusion detection system. Computers & Security. 2020. Vol. 92. P. 101752. DOI: 10.1016/j.cose.2020.101752.

9. ESP32 Series Documentation. URL: https://docs.espressif.com/projects/esp-idf/en/latest/esp32/index.html (Last accessed: 13.02.2026).

10. Bahlali A. R., Bachir A. Machine Learning Anomaly-Based Network Intrusion Detection: Experimental Evaluation. In book: Advanced Information Networking and Applications. Switzerland : Springer Nature, 2023. P. 392–403. DOI: 10.1007/978-3-031-28451-9_34.

11. Aljammal A. H., Al-Oqily I., Obiedat M., Qawasmeh A., Taamneh S., Wedyan F. I. Anomaly intrusion detection using machine learning- IG-R based on NSL-KDD dataset. Bulletin of Electrical Engineering and Informatics. 2024. Vol. 13. № 6. P. 4466–4474. DOI: 10.11591/eei.v13i6.7308.

12. Suricata: Open Source IDS / IPS / NSM engine. URL: https://docs.suricata.io/en/suricata-8.0.4/ (Last accessed: 17.02.2026).

13. Про затвердження Технічного регламенту радіообладнання : постанова КМУ від 24 трав. 2017 р. № 355. URL: https://zakon.rada.gov.ua/laws/show/355-2017-%D0%BF/ed20180401#Text (дата звернення: 17.02.2026).

14. Messaging that just works — RabbitMQ. URL: https://www.rabbitmq.com/docs (Last accessed: 15.03.2026).

15. Raspberry Pi Pico W. *Raspberry Pi Documentation*. URL: https://www.raspberrypi.com/documentation/microcontrollers/raspberry-pi-pico.html (Last accessed: 26.03.2026).

16. ESP32 Wi-Fi & Bluetooth MCU. Espressif Systems. URL: https://www.espressif.com/en/products/socs/esp32 (Last accessed: 14.04.2026).

17. Spring Boot Reference Documentation. Spring Docs. URL: https://docs.spring.io/spring-boot/index.html (Last accessed: 16.04.2026).

18. PostgreSQL: About. PostgreSQL Official Website. URL: https://www.postgresql.org/about/ (Last accessed: 23.05.2026).

19. Elasticsearch Guide. Elastic Documentation. URL: https://www.elastic.co/guide/en/elasticsearch/reference/current/index.html (Last accessed: 23.05.2026).

20. Spring Cloud Netflix. Spring Cloud Features. URL: https://docs.spring.io/spring-cloud-netflix/reference/spring-cloud-netflix.html (Last accessed: 25.05.2026).

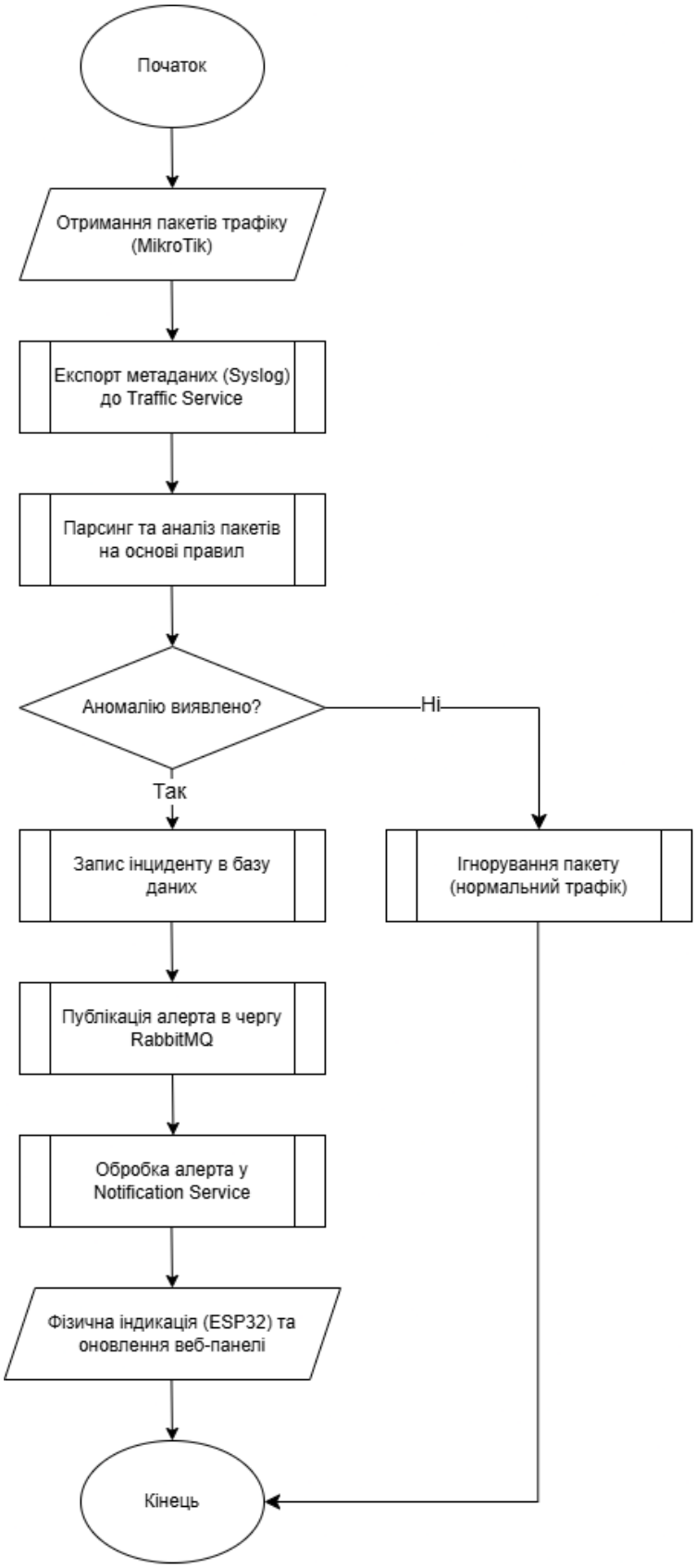
21. RFC 5424 – The Syslog Protocol. Internet Engineering Task Force (IETF), 2009. URL: https://datatracker.ietf.org/doc/html/rfc5424 (Last accessed: 26.05.2026).

22. Wireless Access Point Configuration. MikroTik Documentation. URL: https://help.mikrotik.com/docs/spaces/ROS/pages/1409138/Wireless (Last accessed: 04.06.2026).

23. Nmap: the Network Mapper – Free Security Scanner. Nmap Official Site. URL: https://nmap.org/book/man.html (Last accessed: 07.06.2026).

ДОДАТОК А

Блок-схема алгоритму роботи апаратної частини



ДОДАТОК Б

Код програми IoT-модуля

```
#include <WiFi.h>
#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>
#include <WebServer.h>
#include <ArduinoJson.h>

#define SCREEN_WIDTH 128
#define SCREEN_HEIGHT 64
#define OLED_SDA 21
#define OLED_SCL 22

Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, -1);
WebServer server(80);

const char* ssid = "Traffic_IoT_Net";
const char* password = "esp32_secure123";

void updateScreen(String type, String ip, int size) {
  display.clearDisplay();

  display.setTextSize(1);
  display.setTextColor(WHITE);
  display.setCursor(0, 0);
  display.println("!!! ANOMALY !!!");

  display.setCursor(0, 15);
  display.print("Type: ");
  display.println(type);

  display.setCursor(0, 30);
  display.print("IP: ");
  display.println(ip);

  display.setCursor(0, 45);
  display.print("Size: ");
  display.print(size);
  display.println(" bytes");
```

```
  display.display();
}

void handleAlert() {
  String jsonBody = server.arg("plain");

  if (jsonBody.length() == 0 && server.args() > 0) {
    jsonBody = server.arg(0);
  }

  Serial.println("Received Body: " + jsonBody);

  if (jsonBody.length() == 0) {
    server.send(400, "text/plain", "Body is completely empty");
    return;
  }

  DynamicJsonDocument doc(1024);
  DeserializationError error = deserializeJson(doc, jsonBody);

  if (error) {
    Serial.print("JSON Parse Error: ");
    Serial.println(error.c_str());
    server.send(400, "text/plain", "Invalid JSON format");
    return;
  }

  String anomalyType = doc["anomalyType"].as<String>();
  String sourceIp = doc["sourceIp"].as<String>();
  int sizeBytes = doc["sizeBytes"].as<int>();

  updateScreen(anomalyType, sourceIp, sizeBytes);

  server.send(200, "text/plain", "Alert Successfully Displayed!");
}

void setup() {
  Serial.begin(115200);

  Wire.begin(OLED_SDA, OLED_SCL);
```

```
  display.begin(SSD1306_SWITCHCAPVCC, 0x3C);

  display.clearDisplay();
  display.setTextColor(WHITE);
  display.setCursor(0, 10);
  display.println("Connecting...");
  display.display();

  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
  }

  display.clearDisplay();
  display.setCursor(0, 10);
  display.println("READY FOR ALERTS!");
  display.print("IP: ");
  display.println(WiFi.localIP());
  display.display();

  server.on("/alert", HTTP_POST, handleAlert);
  server.begin();
}

void loop() {
  server.handleClient();
}
```

ДОДАТОК В

Код ядра системи виявлення мережевих аномалій

```
private final Pattern ipPortPattern =
Pattern.compile("(\\d+\\.\\d+\\.\\d+\\.\\d+)(?::(\\d+))?-
>(\\d+\\.\\d+\\.\\d+\\.\\d+)(?::(\\d+))?");
private final Pattern lenPattern = Pattern.compile("len (\\d+)");

@PostConstruct
public void startListening() {
    new Thread(() -> {
        try (DatagramSocket socket = new DatagramSocket(514)) {
            byte[] buffer = new byte[2048];
            while (true) {
                DatagramPacket packet = new DatagramPacket(buffer, buffer.length);
                socket.receive(packet);
                String msg = new String(packet.getData(), 0, packet.getLength());

                if (msg.contains("MTIK_TRAFFIC")) {
                    Matcher ipMatcher = ipPortPattern.matcher(msg);
                    Matcher lenMatcher = lenPattern.matcher(msg);

                    if (ipMatcher.find() && lenMatcher.find()) {
                        String srcIp = ipMatcher.group(1);
                        String destIp = ipMatcher.group(3);
                        String portStr = ipMatcher.group(4);
                        int destPort = (portStr != null) ?
Integer.parseInt(portStr) : 0;
                        int sizeBytes = Integer.parseInt(lenMatcher.group(1));

                        TrafficPacket trafficPacket = new TrafficPacket(srcIp,
destIp, sizeBytes, destPort);
                        trafficService.processRealPacket(trafficPacket);
                    }
                }
            }
```

```
        } catch (Exception e) { System.err.println("UDP Error: " +
e.getMessage()); }
    }).start();
}

public void processRealPacket(TrafficPacket packet) {
    long now = System.currentTimeMillis();

    if (now - lastResetTime > 1000) {
        packetCounts.clear();
        lastResetTime = now;
    }

    int count = packetCounts.getOrDefault(packet.getSourceIp(), 0) + 1;
    packetCounts.put(packet.getSourceIp(), count);

    if (count == DDOS_THRESHOLD) {
        triggerAlert(packet, "DDoS Attack", "High packet rate detected: " + count
+ " pkts/sec");
    }

    if (packet.getDestPort() == 22) {
        triggerAlert(packet, "Security Breach", "Attempted access to closed port
22 (SSH)");
    }

    if (packet.getSizeBytes() > 1000) {
        triggerAlert(packet, "Real High Volume", "Heavy packet detected! Size: " +
packet.getSizeBytes() + " bytes");
    }
}
```

ДОДАТОК Г

Матеріали апробації роботи

Г.1 XVII Міжнародна науково-практичної конференції «Free and Open Source Software»

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ ЕКОНОМІЧНИЙ УНІВЕРСИТЕТ ІМЕНІ СЕМЕНА КУЗНЕЦЯ

КАФЕДРА КІБЕРБЕЗПЕКИ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

МАТЕРІАЛИ

XVII-ої МІЖНАРОДНОЇ НАУКОВО-ПРАКТИЧНОЇ КОНФЕРЕНЦІЇ
«FREE AND OPEN SOURCE SOFTWARE»

Дякуємо за підтримку

11-12 лютого 2026 р.
м. Харків

2026 р. Владислав Євлампієв