МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЧЕРКАСЬКИЙ ДЕРЖАВНИЙ ФАХОВИЙ БІЗНЕС-КОЛЕДЖ
Циклова комісія комп’ютерної інженерії та інформаційних технологій

КВАЛІФІКАЦІЙНА РОБОТА

на тему

ДОСЛІДЖЕННЯ МЕТОДІВ ЗАХИСТУ ВІД АТАК НА РІВНІ ПРОТОКОЛІВ IOT (MQTT, COAP)

Виконав: студент групи 1К-22
Спеціальності
123 Комп’ютерна інженерія
Олександр МЕСЕВРА
Керівник:
Павло РАТАЙЧУК

Черкаси 2026

ЧЕРКАСЬКИЙ ДЕРЖАВНИЙ ФАХОВИЙ БІЗНЕС-КОЛЕДЖ
Циклова комісія комп’ютерної інженерії та інформаційних технологій
Спеціальність 123 «Комп’ютерна інженерія»
Освітня програма Комп’ютерна інженерія

ЗАТВЕРДЖУЮ
Завідувачка циклової комісії КІ та ІТ
Наталя ФАЛЬЧЕНКО
«_____» _____2026 р.

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ
Месевра Олександр Олександрович

1. Тема кваліфікаційної роботи Дослідження методів захисту від атак на рівні протоколів IoT (MQTT, CoAP) Керівник роботи Ратайчук П. Є., викладач вищої категорії затверджені наказом закладу фахової передвищої освіти від «6» жовтня 2025 року № 84/1-у.

2. Строк подання студентом кваліфікаційної роботи 02.06.2026

3. Вихідні дані до кваліфікаційної роботи специфікації MQTT, CoAP, TLS 1.3, DTLS 1.3; література з кібербезпеки IoT; ПЗ: Docker, Eclipse Mosquitto, Wireshark, OpenSSL, утиліти пентестування.

4. Зміст кваліфікаційної роботи дослідження архітектури та безпеки IoT-систем; аналіз протоколів MQTT та CoAP; класифікація прикладних атак; вивчення методів шифрування TLS/DTLS та політик ACL; розгортання Docker-стенду; моделювання атак Connection Flooding, Sniffing та Replay; практична реалізація захисту брокера й сервера; аналіз мережевого трафіку та оцінка продуктивності мережі після впровадження засобів безпеки.

5. Дата видачі завдання 15.09.2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Термін виконання етапів	Примітка про виконання
1	Вступ	14.10.2025	
2	РОЗДІЛ 1 ТЕОРЕТИЧНІ ОСНОВИ ФУНКЦІОНУВАННЯ ТА БЕЗПЕКИ ПРОТОКОЛІВ ІОТ	09.12.2025	
3	РОЗДІЛ 2. АНАЛІЗ МЕТОДІВ ТА ЗАСОБІВ ЗАХИСТУ ВІД АТАК НА РІВНІ ПРОТОКОЛІВ MQTT ТА CoAP	10.03.2026	
4	РОЗДІЛ 3 ПРАКТИЧНА РЕАЛІЗАЦІЯ ТА МОДЕЛЮВАННЯ СИСТЕМИ ЗАХИСТУ MQTT ТА CoAP	28.04.2026	
5	Висновки	12.05.2026	
6	Оформлення кваліфікаційної роботи (чистовий варіант)	20.05.2026	
7	Перевірка кваліфікаційної роботи на наявність ознак плагіату (за 10 днів до захисту)	05.06.2026	
8	Подання кваліфікаційної роботи на затвердження завідувачу ЦК (за 7 днів до захисту)	08.06.2026	

Студент Олександр МЕСЕВРА

Керівник роботи Павло РАТАЙЧУК

АНОТАЦІЯ

Кваліфікаційна робота присвячена дослідженню методів захисту від атак на рівні протоколів Інтернету речей MQTT та CoAP, які широко використовуються для забезпечення взаємодії між пристроями в сучасних IoT-системах. Актуальність теми обумовлена стрімким розвитком технологій Інтернету речей та зростанням кількості підключених пристроїв, що призводить до збільшення ризиків виникнення кіберзагроз і несанкціонованого доступу до мережевих ресурсів. Особливості архітектури IoT-систем, обмежені обчислювальні ресурси пристроїв та специфіка протоколів передачі даних створюють додаткові виклики щодо забезпечення належного рівня інформаційної безпеки.

У процесі дослідження проаналізовано сучасні підходи до захисту IoT-мереж, включаючи використання криптографічних засобів, механізмів автентифікації та авторизації користувачів, технологій шифрування транспортного рівня, систем контролю доступу та моніторингу мережевого трафіку. Особливу увагу приділено впровадженню захищених протоколів TLS і DTLS для забезпечення безпечного обміну даними між компонентами мережі. Визначено переваги та обмеження різних методів захисту з урахуванням специфіки функціонування IoT-пристроїв.

Практична частина роботи передбачає моделювання мережі Інтернету речей із використанням протоколів MQTT та CoAP, відтворення типових сценаріїв атак і дослідження ефективності запропонованих механізмів захисту.

КЛЮЧОВІ СЛОВА: ІНТЕРНЕТ РЕЧЕЙ; MQTT; COAP; КІБЕРБЕЗПЕКА; ІНФОРМАЦІЙНА БЕЗПЕКА; МЕРЕЖЕВІ АТАКИ; АВТЕНТИФІКАЦІЯ; ШИФРУВАННЯ; TLS; DTLS; ЗАХИСТ IOT-МЕРЕЖ.

ABSTRACT

The qualification work is devoted to the study of methods of protection against attacks at the level of the Internet of Things protocols MQTT and CoAP, which are widely used to ensure interaction between devices in modern IoT systems. The relevance of the topic is due to the rapid development of Internet of Things technologies and the growth of the number of connected devices, which leads to an increase in the risks of cyber threats and unauthorized access to network resources. The features of the architecture of IoT systems, limited computing resources of devices and the specificity of data transmission protocols create additional challenges in ensuring the proper level of information security.

The research analyzed modern approaches to protecting IoT networks, including the use of cryptographic tools, user authentication and authorization mechanisms, transport layer encryption technologies, access control systems, and network traffic monitoring. Particular attention was paid to the implementation of secure TLS and DTLS protocols to ensure secure data exchange between network components. The advantages and limitations of various protection methods were determined, taking into account the specifics of the functioning of IoT devices.

The practical part of the work involves modeling the Internet of Things network using the MQTT and CoAP protocols, reproducing typical attack scenarios, and studying the effectiveness of the proposed protection mechanisms.

KEYWORDS: INTERNET OF THINGS; MQTT; COAP; CYBERSECURITY; INFORMATION SECURITY; NETWORK ATTACKS; AUTHENTICATION; ENCRYPTION; TLS; DTLS; IOT NETWORK PROTECTION.

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАЧЕНЬ

Скорочення	Повна назва (англ.)
ACL	Access Control List
AES	Advanced Encryption Standard
CA	Certificate Authority
CoAP	Constrained Application Protocol
DoS	Denial of Service
DTLS	Datagram Transport Layer Security
IoT	Internet of Things
MITM	Man-in-the-Middle
MQTT	Message Queuing Telemetry Transport
PKI	Public Key Infrastructure
QoS	Quality of Service
TLS	Transport Layer Security

ЗМІСТ

ВСТУП

Актуальність теми на сучасному етапі розвитку інформаційного суспільства спостерігається глобальна трансформація традиційних технологічних процесів у цифрову площину, що зумовлено стрімким впровадженням концепції Інтернету речей (Internet of Things, IoT). Технології IoT поступово охоплюють практично всі сфери людської діяльності: від побутових систем автоматизації та концепцій «розумного міста» до високотехнологічних галузей промисловості, медицини, логістики та енергетики. За оцінками провідних аналітичних агентств, кількість підключених до мережі інтелектуальних пристроїв зростає в геометричній прогресії, що створює передумови для формування всеохоплюючої глобальної цифрової екосистеми.

Проте, масштабна цифровізація та відкритість архітектур IoT-систем породжують критичні виклики у сфері інформаційної безпеки. Специфіка Інтернету речей полягає в тому, що більшість кінцевих вузлів (датчиків, сенсорів) мають суттєві обмеження щодо обчислювальної потужності, обсягу пам’яті та енергозабезпечення. Традиційні протоколи передачі даних, розроблені для стаціонарних комп'ютерних систем, виявляються занадто громіздкими та ресурсомістким для використання в умовах обмежених ресурсів. Як наслідок, для забезпечення ефективного обміну даними були розроблені та впроваджені спеціалізовані протоколи прикладного рівня, серед яких ключове місце посідають MQTT [1] та CoAP [2].

Незважаючи на високу ефективність цих протоколів у контексті мінімізації трафіку та споживання енергії, питання їхньої безпеки залишається одним із найбільш дискусійних у науковій спільноті. Орієнтація розробників на максимальну продуктивність часто призводила до ігнорування вбудованих засобів захисту. Передача критично важливої інформації у відкритому вигляді без використання стійких алгоритмів шифрування та механізмів суворої аутентифікації створює сприятливі умови для реалізації різноманітних

векторів кібератак.

Особливу небезпеку становлять атаки типу «відмова в обслуговуванні» (DoS), перехоплення даних (Man-in-the-Middle) та атаки повторного відтворення (Replay). Враховуючи, що IoT-пристрої часто керують фізичними об’єктами у реальному світі, успішна компрометація протоколів MQTT та CoAP може призвести не лише до витоку конфіденційних даних, але й до серйозних техногенних аварій, фінансових збитків та загрози життю людей. Таким чином, науковий пошук та практична реалізація методів організації захисту на рівні цих протоколів є критично важливим завданням, що визначає актуальність обраної теми дипломної роботи.

Мета дослідження полягає у комплексному дослідженні методів захисту від атак на рівні протоколів IoT (MQTT, CoAP) та їх практичній реалізації шляхом теоретичного аналізу вразливостей, моделювання типових мережевих загроз та оцінки ефективності сучасних криптографічних засобів протидії.

Для досягнення поставленої мети визначено наступні завдання:

- Провести глибокий системний аналіз архітектурних особливостей, принципів функціонування та сфер застосування протоколів прикладного рівня MQTT та CoAP.
- Здійснити класифікацію актуальних загроз інформаційній безпеці в середовищі Інтернету речей та ідентифікувати специфічні вразливості досліджуваних протоколів [5, 7].
- Спроектувати та розгорнути віртуальний дослідницький стенд на базі технологій контейнеризації для імітації роботи IoT-інфраструктури.
- Здійснити практичне моделювання атак типу DoS, MITM та Replay на незахищені вузли мережі та проаналізувати отримані результати за допомогою інструментів глибокого аналізу пакетів.
- Реалізувати та налаштувати сучасні механізми захисту, включаючи протоколи TLS 1.3 [3] та DTLS 1.3 [4], а також впровадити політики контролю доступу (ACL).

- Виконати порівняльний аналіз метрик продуктивності та рівня безпеки системи до і після впровадження засобів захисту для формування обґрунтованих рекомендацій [6].

Об’єкт дослідження процеси інформаційного обміну та забезпечення кібербезпеки в мережевих інфраструктурах Інтернету речей.

Предмет дослідження вразливості, методи виявлення та механізми захисту прикладних протоколів MQTT та CoAP від деструктивних мережевих впливів.

У роботі проведено комплексне дослідження стійкості протоколів MQTT та CoAP до сучасних методів кібератак у реалістичному віртуальному середовищі. Сформовані у роботі практичні рекомендації щодо налаштування захищених конфігурацій брокерів та клієнтів можуть бути безпосередньо впроваджені при розробці систем автоматизації, моніторингу та телеметрії, що дозволить суттєво знизити ризики несанкціонованого втручання в роботу IoT-мереж.

РОЗДІЛ 1 ТЕОРЕТИЧНІ ОСНОВИ ФУНКЦІОНУВАННЯ ТА БЕЗПЕКИ ПРОТОКОЛІВ IOT

В кваліфікаційній роботі розглянуто базові архітектурні принципи побудови мереж Інтернету речей та здійснюється глибокий аналіз функціонування ключових протоколів прикладного рівня – MQTT та CoAP. Значну увагу приділено дослідженню теоретичних аспектів їхньої вразливості до сучасних мережевих кіберзагроз через відсутність суворих механізмів криптографічного захисту в базових специфікаціях. Також у розділі наведено класифікацію основних векторів атак на IoT-інфраструктуру, що формує необхідну теоретичну та методологічну базу для подальшого практичного моделювання ефективної системи захисту.

1.1 Архітектура IoT-систем та особливості їх побудови

Концепція Інтернету речей (Internet of Things, IoT) являє собою комплексну парадигму обчислювальної мережі, яка об'єднує фізичні об'єкти (речі), оснащені вбудованими технологіями для взаємодії один з одним та із зовнішнім середовищем. З інженерної точки зору, IoT-система не є монолітним рішенням, а складається з гетерогенного набору апаратних та програмних компонентів, які функціонують на різних рівнях мережевої моделі.

На сьогоднішній день не існує єдиного універсального стандарту архітектури IoT, проте більшість дослідників та міжнародних організацій (зокрема, ITU-T та IEEE) спираються на базову еталонну трирівневу модель[8]., показану на рис.1.1. Ця модель дозволяє чітко розмежувати процеси збору, передачі та обробки інформації, що є критично важливим для проєктування безпечних систем. Базова архітектура включає в себе рівень сприйняття, мережевий рівень (Network Layer) та прикладний рівень.

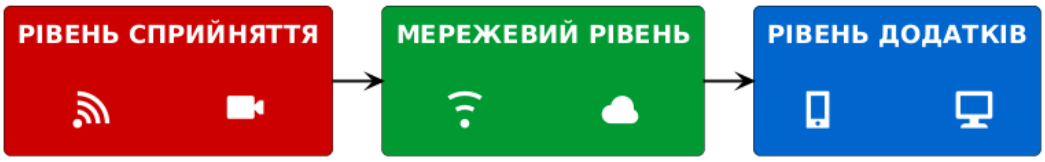

Рисунок 1.1 – Базова трирівнева еталонна архітектура систем Інтернету речей

Рівень сприйняття (Perception Layer) є найнижчим, фізичним рівнем архітектури. Його головна функція полягає у взаємодії з фізичним світом: зборі параметрів навколишнього середовища, ідентифікації об'єктів та виконанні фізичних дій. До апаратного забезпечення цього рівня належать різноманітні сенсори (температури, вологості, руху), RFID-мітки, мікроконтролери (MCU) та актуатори (виконавчі механізми, реле, сервоприводи). Саме на цьому рівні генерується первинна телеметрія. Специфікою цього рівня є те, що більшість вузлів функціонують на базі мікроконтролерів з жорсткими апаратними обмеженнями, часто працюючи від автономних джерел живлення (батарейок) роками.

Мережевий рівень (Network Layer) виконує роль транспортної магістралі. Він відповідає за маршрутизацію та безпечну передачу даних, зібраних на рівні сприйняття, до центрів обробки або хмарних платформ. Через гетерогенність кінцевих пристроїв, на цьому рівні працює широкий спектр технологій зв'язку: від дротових (Ethernet, PLC) до бездротових (Wi-Fi, Bluetooth Low Energy, Zigbee, LoRaWAN, 4G/5G). Ключовим елементом мережевого рівня часто виступає IoT-шлюз (Gateway), який виконує трансляцію протоколів та агрегацію трафіку перед його відправкою у глобальну мережу.

Прикладний рівень (Application Layer) є верхнім рівнем архітектури, який забезпечує кінцевого користувача специфічними послугами (розумний дім, моніторинг промислового обладнання, медична телеметрія). Тут відбувається збереження, аналітика (в тому числі з використанням методів машинного навчання), візуалізація даних та прийняття керівних рішень. Саме на цьому рівні функціонують комунікаційні протоколи MQTT [1] та CoAP [2],

які забезпечують оптимізований обмін повідомленнями між пристроями та серверами.

Проєктування та побудова IoT-систем суттєво відрізняється від традиційних IT-інфраструктур. Ця відмінність зумовлена низкою інженерних особливостей, які безпосередньо впливають на вибір методів організації кіберзахисту. До ключових особливостей побудови IoT-систем слід віднести:

Жорсткі апаратні обмеження кінцевих вузлів. Відповідно до класифікації IETF (зокрема RFC 7228)[9], більшість IoT-пристроїв належать до класу обмежених пристроїв (Constrained Devices). Вони мають обсяг оперативної пам'яті (RAM) на рівні десятків кілобайт та флеш-пам'яті (ROM) на рівні сотень кілобайт. Це унеможливлює використання повноцінних операційних систем (таких як Linux) та класичних мережевих стеків, а також робить неможливим обчислення складних криптографічних алгоритмів (наприклад, RSA з великим ключем) у реальному часі без критичних затримок.

Енергозалежність та робочий цикл. Багато пристроїв живляться від батарей і використовують режим глибокого сну (Deep Sleep), прокидаючись лише на кілька мілісекунд для відправки даних. Використання багатоетапних процедур встановлення захищеного з'єднання (як у класичному TLS) призводить до швидкого виснаження батареї.

Обмеженість пропускної здатності (Constrained Networks). Вузли часто працюють у мережах з високим рівнем втрат пакетів (Lossy Networks) та низькою пропускною здатністю. Застосування традиційних текстових протоколів (як HTTP) створює надмірні накладні витрати на заголовки пакетів, що перевищують розмір самого корисного навантаження (payload).

Масштабованість та динамічна топологія. Сучасні промислові IoT-мережі можуть налічувати десятки тисяч вузлів, які постійно підключаються або відключаються від мережі. Централізовані брокери (як у випадку з MQTT) повинні витримувати колосальне навантаження у вигляді одночасних з'єднань, що робить їх ідеальною мішенню для атак типу відмови в обслуговуванні (DoS).

Гетерогенність екосистеми. IoT-мережі об'єднують пристрої від різних виробників, які використовують різні стандарти та платформи. Це ускладнює впровадження єдиної уніфікованої політики безпеки.

Таким чином, архітектура систем Інтернету речей є компромісом між функціональністю та апаратними можливостями пристроїв. Саме ці обмеження змусили інженерів відмовитися від традиційного стеку TCP/IP (HTTP/TLS) на користь спеціалізованих легковигих протоколів прикладного рівня. Проте, оптимізація продуктивності призвела до послаблення механізмів безпеки, що формує широку поверхню для кібератак і вимагає розробки адаптивних методів захисту на рівні самих протоколів.

1.2 Протокол MQTT: принцип роботи, архітектура, рівні QoS

Протокол MQTT (Message Queuing Telemetry Transport) на сьогоднішній день де-факто є стандартом для обміну повідомленнями в системах Інтернету речей. Розроблений наприкінці 1990-х років Енді Стенфорд-Кларком (IBM) та Арленом Ніппером (Arcom), він створювався для вирішення специфічного завдання: моніторингу нафтопроводів через супутникові канали зв’язку з низькою пропускною здатністю та високою вартістю трафіку. Ці початкові вимоги визначили ключові характеристики протоколу: мінімальний розмір заголовка пакета (від 2 байт), робота поверх надійного транспортного протоколу TCP та використання асинхронної моделі передачі даних [1].

На відміну від традиційної клієнт-серверної моделі (як у HTTP), де клієнт робить прямий запит до сервера і чекає на відповідь, MQTT базується на архітектурі «публікація-підписка» (Publish-Subscribe). В цій моделі відсутній прямий зв'язок між відправником (Publisher) та отримувачем (Subscriber) інформації. Всіма процесами керує центральний вузол - брокер (Broker).

Рисунок 1.2 – Архітектура взаємодії компонентів у протоколі MQTT

Основними елементами архітектури є: (Клієнт, Брокер, Топік)

Клієнт (Client) будь-який пристрій (від мікроконтролера ESP32 до потужного сервера), що підключається до брокера. Клієнт може бути як видавцем повідомлень, так і підписником, або виконувати обидві ролі одночасно.

Брокер (Broker) серверне програмне забезпечення, яке відповідає за прийом усіх повідомлень, їхню фільтрацію та подальшу розсилку підписникам. Брокер також керує автентифікацією клієнтів та підтримує стан сесій.

Топік (Topic) символьний рядок, що використовується для маршрутизації повідомлень. Топіки мають ієрархічну структуру, де рівні розділяються символом косої риски (наприклад, home/livingroom/temperature).

Важливою особливістю MQTT є система фільтрації за допомогою шаблонів (wildcards). Підписник може підписатися на конкретний топік або використовувати спецсимволи: «+» для заміни одного рівня ієрархії та «#» для заміни всіх наступних рівнів. Наприклад, підписка на home/+/temperature дозволить отримувати дані про температуру з усіх кімнат. Така гнучкість спрощує масштабування систем, але водночас створює ризики: зловмисник, отримавши доступ до підписки через «#», може перехопити абсолютно весь трафік брокера, якщо не налаштовані списки контролю доступу (ACL) [5].

Процес взаємодії починається з відправки клієнтом пакету CONNECT. У цьому пакеті передаються ідентифікатор клієнта (ClientId), прапорець Clean Session (чи потрібно зберігати дані після відключення) та, за необхідності, логін і пароль. Важливою функцією є Last Will and Testament (LWT) -

повідомлення, яке брокер автоматично розішле підписникам, якщо зв'язок із клієнтом раптово перерветься.

Однією з найважливіших характеристик MQTT, що забезпечує надійність у нестабільних мережах, є механізм рівнів якості обслуговування (Quality of Service, QoS). Протокол підтримує три рівні доставки повідомлень, що відображено в табл.1.1.

Таблиця 1.1 – Характеристика рівнів якості обслуговування в MQTT

Рівень QoS	Назва	Опис механізму	Накладні витрати
QoS 0	At most once	«Вистрілив і забув». Повідомлення надсилається один раз без підтвердження доставки.	Мінімальні
QoS 1	At least once	Повідомлення гарантовано буде доставлено, але можливі дублікати. Очікується підтвердження PUBACK.	Середні
QoS 2	Exactly once	Найвищий рівень надійності. Гарантується доставка рівно один раз через чотирьохетапне підтвердження.	Високі

QoS 0 (Максимум один раз) використовується для некритичних даних телеметрії, де втрата окремого пакета (наприклад, поточного значення вологості повітря) не вплине на роботу всієї системи. Це найшвидший режим з мінімальним споживанням трафіку.

QoS 1 (Мінімум один раз) вимагає, щоб брокер або клієнт підтвердили отримання пакету. Якщо відправник не отримує PUBACK протягом визначеного часу, він надсилає повідомлення повторно. Це створює ризик появи дублікатів на стороні отримувача, що необхідно враховувати в логіці роботи додатку.

QoS 2 (Рівно один раз) – це найбільш складний режим, що використовує пакети PUBREC, PUBREL та PUBCOMP. Він виключає втрату або дублювання даних, що критично важливо для систем управління (наприклад, команда на відкриття замка або вимкнення обладнання). Однак, цей режим

створює найбільше навантаження на мережевий стек та процесор, а також збільшує затримки (latency).

Попри свою досконаліть у передачі даних, базовий протокол MQTT версії 3.1.1 або 5.0 має суттєву ваду - він не передбачає шифрування вмісту пакетів на рівні самого протоколу. Всі дані, включаючи логіни, паролі та корисне навантаження (payload), передаються у відкритому вигляді. Це означає, що будь-який вузол у мережі, через який проходить трафік, може здійснити атаку перехоплення або підміни даних. Саме тому організація захисту через зовнішні механізми (TLS) є обов'язковою умовою для безпечної експлуатації MQTT у відкритих мережах.

1.3 Протокол CoAP: модель взаємодії, методи запитів, особливості UDP

Протокол CoAP (Constrained Application Protocol) є спеціалізованим протоколом прикладного рівня, розробленим робочою групою CoRE (Constrained RESTful Environments) інженерної ради Інтернету (IETF) і стандартизованим у документі RFC 7252 [2]. Головною метою створення CoAP була адаптація архітектури класичного веб-інтернету (HTTP) для потреб пристроїв із суттєвими апаратними обмеженнями, які працюють у мережах з високим рівнем втрат пакетів та низькою пропускною здатністю (Lossy Networks).

На відміну від MQTT, який використовує асинхронну модель «публікація-підписка» та потребує центрального брокера, CoAP базується на класичній синхронній (або асинхронній) архітектурі «клієнт-сервер» та повністю підтримує парадигму REST (Representational State Transfer). Кожен фізичний або логічний об'єкт у мережі CoAP (датчик температури або реле) розглядається як ресурс, що має свій унікальний ідентифікатор (URI). Для звернення до ресурсів використовується схема на рисунку 1.3, що робить протокол семантично дуже схожим на HTTP і дозволяє легко інтегрувати IoT-пристрої у традиційні веб-сервіси через спеціальні проксі-сервери.

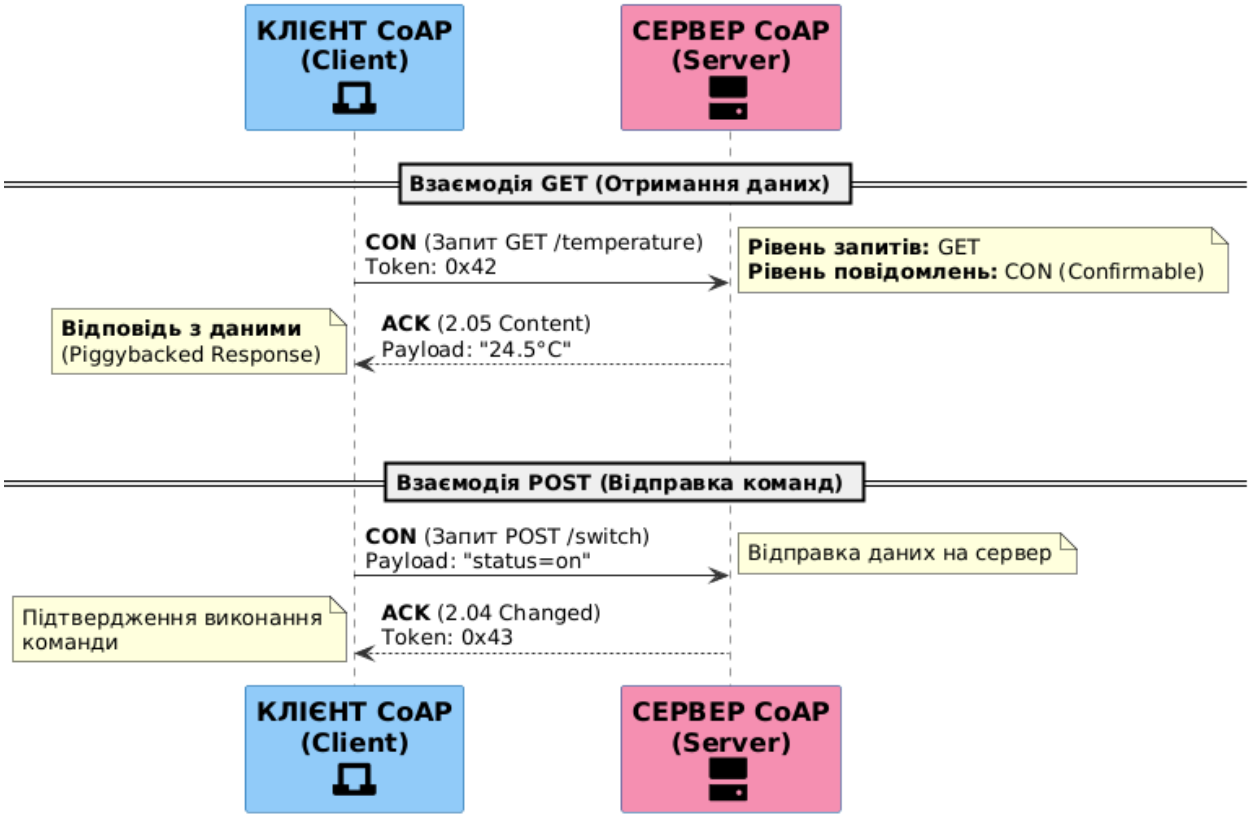

Рисунок 1.3– Модель взаємодії, обміну повідомленнями у протоколі CoAP [3]

Ключовою архітектурною відмінністю CoAP від HTTP та MQTT є використання транспортного протоколу UDP (User Datagram Protocol) замість TCP. Вибір UDP обґрунтований жорсткими вимогами до енергоефективності IoT-пристроїв. Протокол TCP є орієнтованим на з'єднання (connection-oriented): перед початком передачі даних він вимагає виконання процедури «потрійного рукостискання» (3-way handshake), постійно підтримує стан сесії та має громіздкі механізми контролю перевантаження. Для датчика, що прокидається на кілька мілісекунд, щоб передати 2 байти інформації про температуру, накладні витрати на встановлення TCP-з'єднання є критично високими.

Використання UDP дозволяє надсилати датаграми миттєво («fire and forget»), а розмір базового заголовка повідомлення CoAP складає всього 4 байти (порівняно з десятками байтів у HTTP/TCP). Проте, оскільки UDP є протоколом, що не гарантує доставку пакетів, розробникам CoAP довелося

реалізувати власний легковаговий механізм забезпечення надійності (QoS) безпосередньо на прикладному рівні.

Для керування надійністю доставки CoAP використовує чотири типи повідомлень:

Confirmable (CON) – Повідомлення, що потребують підтвердження. Якщо клієнт надсилає запит типу CON, сервер зобов'язаний надіслати у відповідь підтвердження. Якщо підтвердження не отримано за встановлений тайм-аут, клієнт виконує експоненційне збільшення часу очікування та повторює відправку.

Non-confirmable (NON) – Повідомлення без підтвердження. Використовуються для регулярної телеметрії, де втрата одного пакета не є критичною (аналог QoS 0 в MQTT).

Acknowledgment (ACK) – Підтвердження повідомлення, що генерується сервером або клієнтом у відповідь на отримання пакета типу CON. Воно містить той самий ідентифікатор повідомлення (Message ID), що й оригінальний запит.

Reset (RST) – Скидання та адсилається, якщо вузол отримав повідомлення CON, але не може його обробити (наприклад, через нестачу пам'яті або відсутність запитуваного ресурсу).

Взаємодія з ресурсами в CoAP здійснюється за допомогою стандартних RESTful-методів, які ідентичні методам HTTP, але оптимізовані для бінарної передачі:

GET запит на отримання поточного стану ресурсу (наприклад, зчитування показників датчика);

POST запит на створення нового ресурсу або ініціацію певної дії на пристрої;

PUT оновлення стану існуючого ресурсу (наприклад, зміна конфігурації мікроконтролера);

DELETE видалення ресурсу.

Відповіді сервера також маркуються спеціальними числовими кодами,

схожими на HTTP-статуси. Наприклад, успішне виконання запиту GET позначається кодом 2.05 Content (аналог 200 OK), а відсутність ресурсу - кодом 4.04 Not Found. Додатково CoAP підтримує розширення «Observe» (RFC 7641), яке дозволяє клієнту підписатися на зміни ресурсу і отримувати асинхронні сповіщення від сервера без необхідності постійно надсилати GET-запити.

Незважаючи на високу інженерну елегантність, використання UDP як основного транспортного протоколу породжує серйозні виклики у сфері кібербезпеки. По–перше, UDP вразливий до підміни IP-адреси відправника (IP Spoofing). Зловмисник може надіслати на CoAP-сервер невеликий запит з підробленою IP-адресою жертви. Сервер, обробивши запит, надішле жертві відповідь, розмір якої може в десятки разів перевищувати розмір оригінального запиту. Ця архітектурна вразливість робить CoAP ідеальним інструментом для проведення DoS-атак з плечем підсилення (Amplification Attacks) [7].

По-друге, оскільки UDP не підтримує встановлення захищених сесій за допомогою традиційного протоколу TLS, для забезпечення конфіденційності, цілісності та автентифікації вузлів у мережах CoAP необхідно використовувати його датаграмну версію – протокол DTLS (Datagram Transport Layer Security) [4]. Відсутність налаштованого DTLS за замовчуванням у багатьох IoT-рішеннях дозволяє зловмисникам вільно перехоплювати відкритий трафік та здійснювати атаки на інфраструктуру розумних пристроїв.

1.4 Порівняльний аналіз MQTT та CoAP

Незважаючи на те, що протоколи MQTT та CoAP були розроблені для вирішення спільної глобальної проблеми забезпечення зв'язку в умовах обмежених ресурсів та нестабільних мереж вони базуються на фундаментально різних архітектурних парадигмах. Вибір між цими протоколами під час проєктування IoT-інфраструктури залежить від специфіки конкретного завдання, топології мережі та апаратних

характеристик кінцевих вузлів. Для повноцінного розуміння їхніх переваг та вразливостей необхідно провести комплексний порівняльний аналіз за ключовими технічними метриками.

Транспортний рівень та накладні витрати. Найбільш суттєва відмінність полягає у виборі транспортного протоколу. MQTT працює поверх надійного протоколу TCP, що гарантує цілісність потоку даних та впорядкованість пакетів на рівні операційної системи. Однак підтримка TCP-з'єднання (Keep-Alive) вимагає постійного обміну службовими пакетами, що призводить до додаткових витрат енергії та мережевого трафіку. CoAP, навпаки, використовує протокол UDP, який працює за принципом передачі незалежних датаграм без попереднього встановлення з'єднання. Це робить CoAP значно енергоефективнішим пристроєм для сенсорів, що більшість часу перебувають у сплячому режимі і прокидаються лише для відправки одного пакету телеметрії. Розмір базового заголовка в MQTT становить 2 байти, тоді як у CoAP 4 байти, проте загальні накладні витрати на рівні транспорту (TCP vs UDP) роблять CoAP більш "легким" для радіоканалу.

Модель взаємодії та маршрутизація MQTT використовує асинхронну модель «публікація-підписка» (Publish-Subscribe) із централізованим брокером. Клієнти не знають про існування один одного, що забезпечує високий рівень просторової та часової розв'язки (decoupling). Ця модель ідеально підходить для сценаріїв маршрутизації «один до багатьох» (One-to-Many) або «багато до багатьох» (Many-to-Many). CoAP побудований на синхронній (з можливістю асинхронності через механізм Observe) REST-архітектурі «клієнт-сервер». Взаємодія відбувається безпосередньо (або через проксі) шляхом звернення до конкретних URI за допомогою методів GET, POST, PUT, DELETE. Це робить CoAP семантично сумісним із традиційним веб-стеком (HTTP), що суттєво спрощує інтеграцію IoT-пристроїв із хмарними веб-сервісами.

Таблиця 1.2 – Порівняльна характеристика протоколів прикладного рівня MQTT та CoAP

Характеристика	MQTT	CoAP
Повна назва	Message Queuing Telemetry Transport	Constrained Application Protocol
Транспортний рівень	TCP	UDP
Модель взаємодії	Publish-Subscribe (через брокер)	Client-Server (RESTful)
Архітектурний стиль	Подієво-орієнтований (Event-driven)	Орієнтований на стан (State-driven)
Надійність (QoS)	3 рівні: QoS 0, QoS 1, QoS 2	Повідомлення CON (Confirmable) та NON
Розмір заголовка	Від 2 байт	Фіксовано 4 байти
Тип повідомлень	Бінарні (довільне корисне навантаження)	Бінарні з підтримкою т ипів контенту (MIME)
Сумісність з HTTP	Низька (вимагає складних шлюзів)	Висока (крос-протокольні проксі-сервери)
Підтримка Multicast	Не підтримується (тільки через топіки)	Підтримується (рідна функція UDP)
Стандарт шифрування	TLS (Transport Layer Security)	DTLS (Datagram Transport Layer Security)

Механізми забезпечення надійності (QoS). Оскільки MQTT працює поверх TCP, його механізм QoS (рівні 0, 1, 2) відповідає виключно за доставку повідомлень на прикладному рівні від видавця до брокера і від брокера до підписника. CoAP, використовуючи ненадійний UDP, змушений реалізовувати логіку підтвердження доставки самостійно за допомогою повідомлень типу Confirmable (CON) та Acknowledgment (ACK) що відображено в табл 1.2.

Вимоги до безпеки. Обидва протоколи у своїх базових специфікаціях створювалися з акцентом на швидкість і мінімізацію трафіку, тому вони не включають вбудованих криптографічних алгоритмів шифрування корисного навантаження. Проте підходи до організації їхнього захисту різняться. Для MQTT загальноприйнятим стандартом є інкапсуляція трафіку в тунель TLS [3], що вимагає додаткових обчислювальних витрат на етапі встановлення з'єднання (TLS Handshake) та постійної підтримки TCP-сесії.

Для CoAP стандартом IETF передбачено використання DTLS [4] - датаграмної версії TLS, яка здатна працювати поверх UDP і справлятися з втратою або зміною порядку пакетів. Однак впровадження DTLS на мікроконтролерах із суворими обмеженнями пам'яті (класу 0 за RFC 7228) є складною інженерною задачею. Крім того, як вже зазначалося, використання UDP робить CoAP-вузли (без належної конфігурації автентифікації) потенційними учасниками розподілених атак типу «відмова в обслуговуванні» (DDoS Amplification).

Підсумовуючи порівняльний аналіз, можна зробити висновок: MQTT є оптимальним вибором для систем з постійним живленням і надійними каналами зв'язку, де важлива складна маршрутизація між великою кількістю пристроїв. CoAP є кращим рішенням для повністю автономних сенсорних мереж, що живляться від батарей і працюють через нестабільні радіоканали, де потрібна пряма інтеграція з веб-сервісами. Незалежно від обраного протоколу, критичним етапом проєктування є накладання додаткових рівнів безпеки для протидії типовим мережевим загрозам.

1.5 Типові загрози та вразливості IoT-протоколів

Безпека в системах Інтернету речей суттєво відрізняється від класичної інформаційної безпеки в корпоративних мережах[10]. Це зумовлено не лише апаратними обмеженнями кінцевих вузлів, які розглядалися раніше, але й специфікою середовища розгортання. IoT-пристрої (датчики, камери, контролери розумного міста) часто розміщуються у відкритому, фізично неконтрольованому середовищі, що робить їх доступними для безпосереднього втручання. Однак найбільший масив загроз криється саме на мережевому та прикладному рівнях, де функціонують протоколи MQTT та CoAP.

Головна архітектурна вразливість цих протоколів полягає у тому, що вони проєктувалися з акцентом на максимальну продуктивність, мінімальні затримки (latency) та економію заряду батареї. Питання безпеки в їхніх

базових специфікаціях було винесено за дужки, з розрахунком на те, що захист буде реалізовано на нижчих рівнях моделі OSI (наприклад, через VPN або ізольовані VLAN). На практиці ж IoT-пристрої часто підключаються через відкриті бездротові мережі або публічний Інтернет, оголюючи фундаментальні недоліки протоколів.

Дослідження безпеки прикладних IoT-протоколів дозволяє виділити кілька ключових категорій вразливостей, які експлуатуються зловмисниками:

1. Відсутність криптографічного захисту корисного навантаження (Plaintext Transmission) Базові версії MQTT та CoAP не містять вбудованих механізмів шифрування даних. Уся телеметрія, керуючі команди, імена топіків (в MQTT), URI ресурсів (в CoAP), а також облікові дані (логіни та паролі) передаються у відкритому текстовому або бінарному вигляді. Будь-який вузол, що знаходиться в одному широкомовному домені з пристроєм, або зловмисник, який отримав доступ до проміжного маршрутизатора, може використовувати мережеві аналізатори (сніфери) для повного перехоплення трафіку [5]. Ця вразливість повністю нівелює конфіденційність системи.

2. Слабка або відсутня автентифікація вузлів За замовчуванням брокери MQTT (наприклад, популярний Eclipse Mosquitto) у базовій конфігурації дозволяють анонімні підключення. Навіть якщо адміністратор вмикає парольну автентифікацію, паролі часто передаються у відкритому вигляді (пакет CONNECT), що робить їх легкою здобиччю при перехопленні. У протоколі CoAP автентифікація клієнта також не передбачена самою специфікацією протоколу і повністю покладається на зовнішній протокол DTLS. Відсутність суворої взаємної автентифікації (Mutual Authentication) дозволяє зловмиснику легко підключити несанкціонований пристрій до мережі та почати ін'єкцію хибних даних (Data Injection).

3. Вразливості механізмів авторизації та маршрутизації У протоколі MQTT маршрутизація здійснюється на основі топіків. Якщо на брокері не налаштовані суворі списки контролю доступу (Access Control Lists, ACL), будь-який автентифікований (або анонімний) клієнт може підписатися на

кореневий топік з використанням символу # (наприклад, підписка на # означає отримання абсолютно всіх повідомлень, що проходять через брокер). Це призводить до критичного витоку інформації. Крім того, відсутність ACL дозволяє звичайному датчику публікувати керуючі команди в топіки, призначені для адміністраторів або актуаторів.

4. Вразливість до вичерпання ресурсів (Resource Exhaustion) IoT-протоколи чутливі до перевантажень через обмеженість обчислювальних потужностей. Вразливість MQTT криється у використанні TCP. Зловмисник може ініціювати велику кількість напіввідкритих з'єднань (TCP SYN Flood) або надсилати безліч легітимних запитів CONNECT без подальшої передачі даних. Брокер виділяє оперативну пам'ять під кожну сесію, що швидко призводить до вичерпання RAM і відмови в обслуговуванні легітимних клієнтів. CoAP, своєю чергою, вразливий до вичерпання ресурсів через необхідність обробки повідомлень типу Confirmable (CON), які змушують сервер генерувати відповіді та зберігати стан сесії до отримання підтвердження.

5. Вразливість протоколу UDP до підміни адреси (IP Spoofing) Як зазначалося раніше, CoAP працює поверх UDP. Оскільки UDP не встановлює з'єднання, перевірити справжність IP-адреси відправника на рівні транспорту неможливо. Зловмисники масово експлуатують цю вразливість для генерації підроблених запитів, що є базисом для проведення руйнівних атак типу DDoS з підсиленням трафіку.

Виявлені вразливості демонструють, що протоколи MQTT та CoAP "з коробки" не відповідають сучасним вимогам кібербезпеки.

1.6 Класифікація атак на рівні прикладних протоколів

Експлуатація архітектурних вразливостей протоколів MQTT та CoAP дозволяє зловмисникам реалізовувати широкий спектр кібернетичних атак. Для побудови ефективної системи захисту необхідно систематизувати ці загрози. Традиційно в інформаційній безпеці атаки класифікують за їхнім

впливом на базові принципи тріади CIA конфіденційність, цілісність та доступність що відображено в табл 1.3.

Таблиця 1.3 – Класифікація мережевих атак на прикладні протоколи IoT

Клас атаки	Ціль впливу	Типові вектори реалізації	Вразливий протокол
Порушення доступності	Виведення з ладу вузлів мережі або каналів зв'язку	Connection Flooding (DoS), DDoS Amplification	MQTT, CoAP Продовження таблиці
Порушення конфіденційності та цілісності	Перехоплення, читання та несанкціонована модифікація даних	Man-in-the-Middle (MITM), Sniffing, Data Injection	MQTT, CoAP
Порушення автентичності	Імітація легітимного пристрою, несанкціоноване управління	Replay-атака, IP Spoofing, Brute-force	MQTT, CoAP

1. Атаки типу відмови в обслуговуванні (DoS/DDoS) Основною метою цього класу атак є вичерпання системних або мережевих ресурсів пристрою (найчастіше центрального брокера або сервера), що унеможливлює обробку легітимних запитів. У середовищі MQTT класичною є атака Connection Flooding. Зловмисник генерує тисячі TCP-з'єднань та надсилає пакети CONNECT, не закриваючи сесії. Брокер (наприклад, Mosquitto) змушений виділяти оперативну пам'ять та файлові дескриптори під кожне з'єднання. Після досягнення ліміту (у конфігурації ОС або самого брокера) нові підключення від легітимних датчиків відхиляються. Інший вектор – Topic Flooding, коли авторизований (або анонімний) зловмисник масово публікує великі обсяги даних у різні топіки (наприклад, з рівнем QoS 2), змушуючи брокер витрачати процесорний час на їхню маршрутизацію та збереження. У середовищі CoAP найбільш небезпечною є DDoS Amplification Attack (атака з підсиленням) [7]. Зловмисник формує UDP-датаграми із запитом типу GET /well-known/core (запит на отримання списку всіх ресурсів пристрою), підміняючи власну IP-адресу на IP-адресу жертви (IP Spoofing). CoAP-сервер,

отримавши запит розміром кілька десятків байтів, надсилає відповідь розміром у кілька кілобайтів на адресу жертви. Використання ботнету дозволяє згенерувати колосальний потік сміттєвого трафіку, який повністю блокує канал зв'язку атакованого вузла.

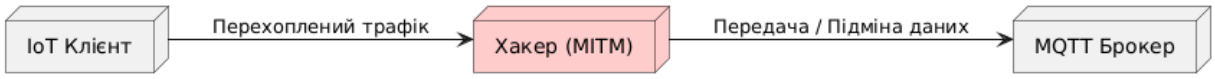

Рисунок 1.4 – Схема перехоплення трафіку при атаці Man-in-the-Middle

2. Атаки перехоплення та підміни (Man-in-the-Middle, MITM) Дана атака реалізується за умови, що зловмисник знаходиться в одному широкомовному домені з жертвою (наприклад, підключений до тієї ж Wi-Fi мережі) або контролює транзитний маршрутизатор. Використовуючи методи підміни ARP-таблиць (ARP Spoofing) або створення підроблених точок доступу (Rogue AP), атакуючий змушує IoT-пристрій відправляти трафік через свій вузол. Оскільки базові специфікації MQTT та CoAP не використовують шифрування, зловмисник за допомогою інструментів глибокого аналізу пакетів (наприклад, Wireshark) може вільно читати перехоплений трафік [5]. Якщо передаються облікові дані в пакеті CONNECT або телеметрія в пакеті PUBLISH, вони стають повністю доступними. Більше того, атакуючий здатний «на льоту» модифікувати корисне навантаження (наприклад, змінити команду status: closed на status: open перед її доставкою брокеру).

3. Повторна передача повідомлень (Replay-атака) Replay-атака є логічним продовженням пасивного перехоплення (Sniffing). Зловмисник не намагається розшифрувати чи змінити пакет, він лише зберігає його копію (набір байтів). Наприклад, перехоплюється команда увімкнення реле електроживлення або відкриття електронного замка. Через певний час зловмисник повторно відправляє цей самий пакет (TCP-сегмент для MQTT або UDP-датаграму для CoAP) на сервер. Якщо система не використовує

динамічні сесійні ключі, криптографічні мітки часу (timestamps) або одноразові номери (nonce), сервер сприймає цей пакет як новий легітимний запит і повторно виконує дію. Ця атака є вкрай критичною для систем промислової автоматизації та розумних будинків, оскільки дозволяє несанкціоновано керувати інфраструктурою без знання паролів.

РОЗДІЛ 2 АНАЛІЗ МЕТОДІВ ТА ЗАСОБІВ ЗАХИСТУ ВІД АТАК НА РІВНІ ПРОТОКОЛІВ MQTT ТА CoAP

2.1 Аутентифікація та авторизація клієнтів

Процес забезпечення безпеки будь–якої інформаційної інфраструктури, зокрема в екосистемі Інтернету речей, фундаментально базується на парадигмі AAA (Authentication, Authorization, Accounting). У контексті використання прикладних протоколів MQTT та CoAP, де кінцеві вузли (датчики, актуатори) часто функціонують в автономному та неконтрольованому середовищі, надійна перевірка їхньої справжності є першим і критично найважливішим бар'єром захисту від несанкціонованого втручання. Відсутність або неправильне налаштування цих механізмів призводить до можливості реалізації атак типу Spoofing (підміна вузла) та несанкціонованої ін'єкції даних.

Механізми аутентифікації в середовищі MQTT Базова специфікація протоколу MQTT [1] передбачає кілька рівнів аутентифікації клієнтів під час їх підключення до центрального брокера. За замовчуванням багато брокерів (наприклад, ранні версії Eclipse Mosquitto) дозволяють анонімні підключення, що є неприпустимим для захищених систем.

Сучасний підхід вимагає обов'язкової перевірки вузла при ініціації сесії. На прикладному рівні це реалізується за допомогою передачі облікових даних у контрольному пакеті CONNECT. Клієнт надсилає брокеру унікальний ідентифікатор (Client ID), а також необов'язкові, але критично важливі для безпеки поля Username (ім'я користувача) та Password (пароль). Проте, даний метод має суттєву слабкість: якщо з'єднання не інкапсульоване в криптографічний тунель, ці облікові дані передаються у відкритому вигляді (Plaintext).

Для вирішення цієї проблеми в інфраструктурах критичного призначення застосовується строга взаємна аутентифікація (Mutual Authentication або mTLS). При такому підході клієнт на етапі встановлення

з'єднання (до відправки пакета CONNECT) пред'являє брокеру свій унікальний X.509 сертифікат, підписаний довіреним центром сертифікації (CA). Брокер криптографічно перевіряє валідність цього сертифіката. У свою чергу, клієнт також перевіряє сертифікат брокера. Цей механізм повністю унеможливлює підміну вузлів та атаки Man-in-the-Middle на етапі підключення [5].

Окрім класичних методів, у сучасних IoT-платформах набуває популярності автентифікація на основі токенів (наприклад, JWT – JSON Web Tokens). Клієнт отримує тимчасовий токен від окремого сервера авторизації (OAuth 2.0) і передає його в полі Password пакета CONNECT. Це дозволяє гнучко керувати часом життя сесії та не зберігати паролі на самих датчиках.

Механізми аутентифікації в середовищі CoAP Протокол CoAP, працюючи поверх UDP, має іншу архітектурну філософію. Специфікація CoAP (RFC 7252) [2] не містить вбудованих полів для логіна чи пароля у своїх базових повідомленнях. Уся відповідальність за перевірку справжності клієнта покладається виключно на протокол DTLS [4]. За стандартом, CoAP поверх DTLS підтримує три режими аутентифікації:

- Pre-Shared Key (PSK) клієнт і сервер заздалегідь володіють спільним симетричним ключем. Це найбільш енергоефективний варіант для мікроконтролерів (з найменшими накладними витратами), але він має проблеми з масштабуванням.
- Raw Public Key (RPK) використовуються асиметричні ключі (публічний/приватний) без складної інфраструктури сертифікатів.
- Сертифікати X.509 забезпечують найвищий рівень безпеки, але вимагають значних ресурсів процесора та пам'яті сенсорного вузла для обчислення криптографічних підписів.

Авторизація та Списки контролю доступу (ACL) Якщо аутентифікація дає відповідь на запитання «Хто ти?», то авторизація визначає «Що тобі дозволено робити?». В IoT-мережах успішне підключення пристрою не означає надання йому абсолютних прав.

Для розмежування прав доступу використовуються Списки контролю

доступу що відображено на рисунку 2.1. В контексті MQTT матриця доступу формується на основі топіків. Наприклад, температурний датчик Sensor_01 отримує право виключно на публікацію (Publish) у топік telemetry/sensor_01/temp. Будь-яка спроба цього вузла підписатися (Subscribe) на топік управління commands/door/lock буде відхилена брокером. Правильно спроектований ACL керується принципом найменших привілеїв.

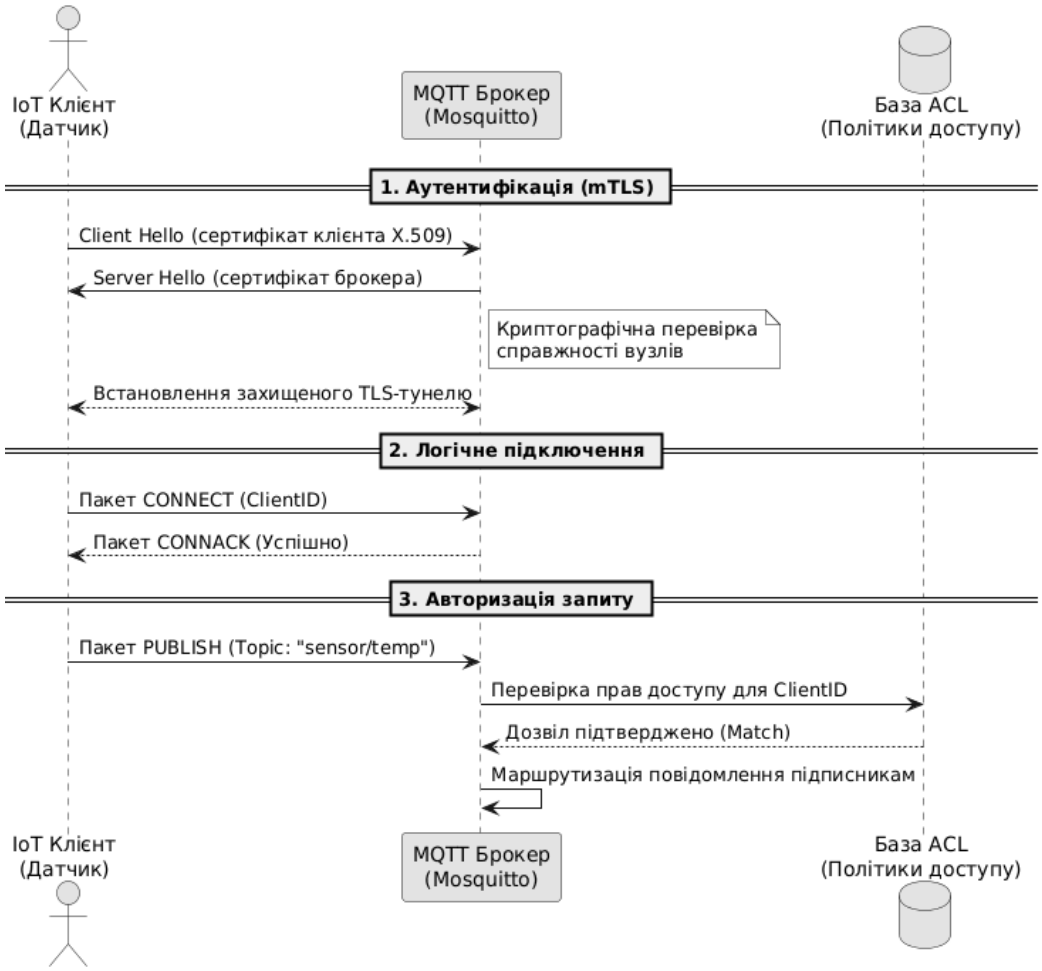

Рисунок 2.1 – Послідовність проходження аутентифікації та авторизації клієнта в захищеній MQTT-мережі

Для систематизації розглянутих методів перевірки сформовано порівняльну таблицю, яка дозволяє обрати оптимальний механізм залежно від наявних апаратних ресурсів системи що відображено в таблиці 2.1.

Таблиця 2.1 – Порівняльна характеристика методів аутентифікації IoT-клієнтів

Метод аутентифікації	Рівень безпеки	Масштабованість мережі	Навантаження на ресурси вузла	Рекомендована сфера застосування
Базова (Login/Password)	Низький (без TLS) / Середній (з TLS)	Висока	Мінімальне	Тестові середовища, локальні закриті мережі
Симетричні ключі (PSK)	Високий	Низька (складне оновлення ключів)	Низьке	Автономні сенсорні мережі (CoAP), пристрої з батарейним живленням
Взаємна (mTLS / X.509)	Дуже високий	Висока (через PKI інфраструктуру)	Високе	Критична інфраструктура, промисловий IoT (MQTT)
Токенна (JWT / OAuth)	Високий	Дуже висока	Середнє	Сучасні хмарні платформи (AWS IoT, Azure), мікросервіси

Таким чином, комплексна підсистема контролю доступу має бути дворівневою: транспортний рівень забезпечує перевірку сертифікатів (відсікаючи підроблені вузли), а прикладний рівень (ACL) гарантує жорстке розмежування прав на взаємодію з інформаційними потоками всередині системи.

2.2 Використання TLS/DTLS для захисту каналу передачі даних

Передача інформаційного потоку у відкритому вигляді (Plaintext) є однією з найбільш критичних уразливостей базових специфікацій протоколів прикладного рівня MQTT та CoAP. Без належного кодування будь-який проміжний вузол у мережі може здійснювати пасивне перехоплення конфіденційних даних телеметрії або активну підміну керуючих команд. Для забезпечення базових критеріїв інформаційної безпеки – конфіденційності (Confidentiality) та цілісності (Integrity) – у проектувальній архітектурі системи

передбачено обов'язкову інкапсуляцію прикладного трафіку в захищені криптографічні тунелі транспортного рівня.

З урахуванням гетерогенності мережевого стеку, захист реалізується двома взаємодоповнюючими стандартами консорціуму IETF: протоколом TLS (Transport Layer Security) для орієнтованого на з'єднання стеку MQTT/TCP та протоколом DTLS (Datagram Transport Layer Security) для датаграмного стеку CoAP/UDP.

Криптографічний захист MQTT на базі протоколу TLS 1.3 Для захисту інформаційного обміну між IoT-клієнтами та центральним брокером у роботі визначено використання найсучаснішої версії протоколу TLS 1.3 (RFC 8446) [3]. Вибір цієї версії замість застарілих TLS 1.1 та TLS 1.2 є жорсткою інженерною вимогою, що зумовлена двома факторами: підвищенням стійкості до кібератак та суттєвою оптимізацією швидкодії, що вкрай важливо для обмежених у ресурсах пристроїв [5].

Головною перевагою TLS 1.3 є редукція процедури повного рукостискання (Handshake) до одного кругового часу затримки (1-RTT). На відміну від TLS 1.2, який вимагав двох RTT для погодження параметрів шифрування, TLS 1.3 дозволяє клієнту надсилати криптографічні припущення (Key Share) разом із першим пакетом Client Hello. Це зменшує час встановлення захищеної сесії вдвічі, знижуючи енергоспоживання радіомодулів датчиків. Крім того, підтримується режим 0-RTT (Session Resumption), який дозволяє клієнту відправляти зашифровані прикладні дані (наприклад, пакет MQTT PUBLISH) безпосередньо у першому повідомленні, якщо сесія з брокером встановлюється повторно.

На рівні кодування в TLS 1.3 повністю вилучено застарілі та вразливі алгоритми шифрування (такі як MD5, SHA-1, RC4, AES-CBC). Протокол підтримує виключно концепцію автентифікованого шифрування з асоційованими даними (AEAD), що гарантує одночасну конфіденційність та перевірку цілісності пакетів на рівні Record Layer. У проектованій архітектурі

брокера Mosquitto жорстко лімітується набір дозволених шифрозасобів, серед яких пріоритет надається:

- TLS_AES_128_GCM_SHA256 як базовий промисловий стандарт;
- TLS_CHACHA20_POLY1305_SHA256 високоефективний алгоритм, оптимізований для пристроїв, що не мають апаратного прискорення AES на рівні кристалу процесора (наприклад, 8-бітні та 32-бітні мікроконтролери архітектури RISC-V без розширення векторного шифрування).

Захищене з'єднання MQTT за замовчуванням маршрутизується через виділений TCP-порт 8883 [1].

Криптографічний захист CoAP на базі протоколу DTLS 1.3 Оскільки протокол CoAP за специфікацією функціонує поверх транспортного протоколу UDP [2], використання класичного TLS є технологічно неможливим через відсутність гарантій доставки та збереження порядку пакетів у мережі. Для захисту датаграмного трафіку застосовується специфікований стандарт DTLS 1.3 (RFC 9147) [4].

Протокол DTLS 1.3 є майже повним функціональним аналогом TLS 1.3, адаптованим до умов ненадійного транспорту. Для компенсації специфіки UDP, у стек DTLS інтегровано додаткові підрівні:

Механізм обробки втрат пакетів (Loss Recovery): Реалізує таймери очікування та повторну генерацію повідомлень рукостискання, якщо один із криптографічних пакетів загубився в каналі зв'язку.

Механізм упорядкування: Кожне повідомлення DTLS Handshake містить явний порядковий номер, що дозволяє CoAP-серверу коректно зібрати криптографічний контекст, навіть якщо датаграми прийшли із порушенням черговості.

Критично важливою інженерною функцією DTLS 1.3 у контексті захисту CoAP-серверів є протидія DoS-атакам з підсиленням (Amplification Attacks), про які йшлося у першому розділі [7]. Для запобігання підміні IP-

адреси відправника DTLS реалізує процедуру попередньої перевірки через сесійні куки. Отримавши запит від клієнта, сервер не виділяє ресурси пам'яті під сесію, а надсилає у відповідь мінімальний пакет із криптографічною міткою. Клієнт зобов’язаний повторити запит, повернувши цю мітку. Це доводить легітимність IP-адреси відправника і нівелює можливість використання пристрою як ретранслятора сміттєвого трафіку.

Захищене з'єднання CoAP що відображено на рисунку 2.2 за стандартом використовує UDP-порт 5684 [2].

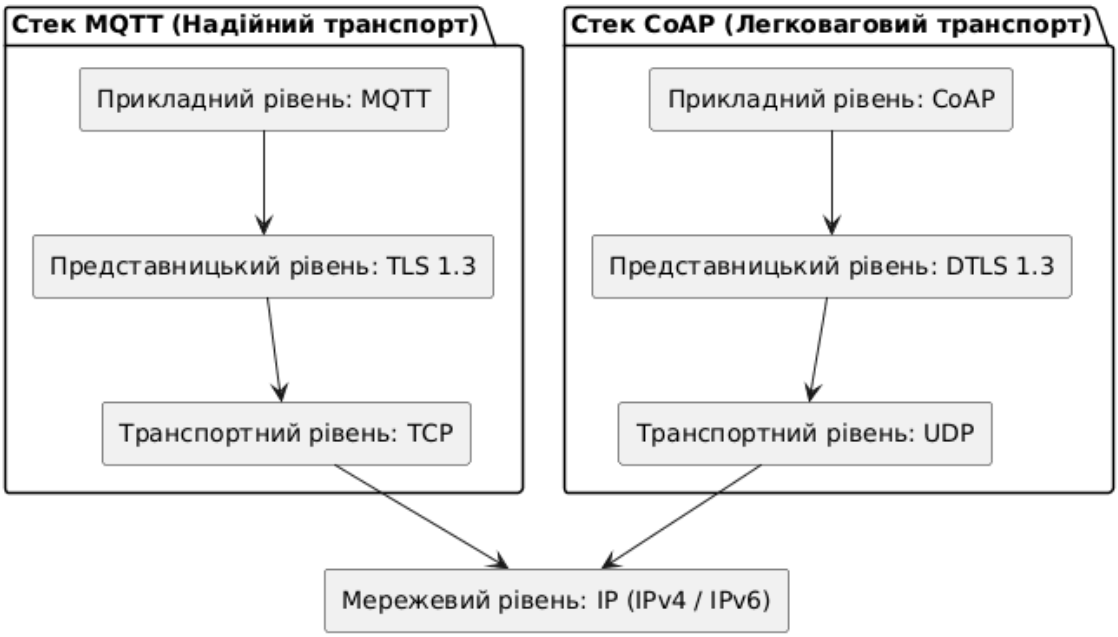

Рисунок 2.2 – Топологія криптографічного інкапсулювання прикладних протоколів на різних рівнях мережевої моделі

2.3 Захист від атак типу DoS/DDoS

Забезпечення доступності (Availability) є одним із найскладніших завдань при проектуванні мереж Інтернету речей. Через обмежені обчислювальні можливості як кінцевих вузлів, так і транзитних шлюзів, IoT-інфраструктура є вкрай чутливою до атак типу «відмова в обслуговуванні» (Denial of Service, DoS) та їх розподілених варіантів (DDoS). Метою таких атак є критичне вичерпання ресурсів (процесорного часу, оперативної пам'яті,

пропускної здатності каналу або пулу мережевих з'єднань), що призводить до повного паралічу легітимного інформаційного обміну.

Специфіка протоколів MQTT та CoAP вимагає застосування різних стратегій протидії, оскільки вони функціонують поверх різних транспортних протоколів (TCP та UDP відповідно) і мають різні архітектурні вразливості.

Протидія DoS-атакам в інфраструктурі MQTT Основним вектором атак на MQTT-мережі є перевантаження центрального брокера. Оскільки MQTT базується на протоколі TCP, брокер змушений виділяти системні ресурси (оперативну пам'ять та файлові дескриптори) під кожне активне з'єднання. Зловмисники найчастіше використовують атаки типу Connection Flooding (масова генерація запитів CONNECT без подальшої передачі даних) або Topic Flooding (спам великими повідомленнями в різні топіки).

Для комплексного захисту MQTT-брокера (наприклад, Eclipse Mosquitto) в проектованій системі реалізується багаторівневий підхід, що поєднує мережеві екрани та внутрішні конфігураційні ліміти самого брокера:

1. Мережевий рівень (Firewall / iptables)для застосування політик обмеження швидкості встановлення нових TCP-з'єднань (Rate Limiting). Наприклад, обмеження кількості нових сесій типу SYN з однієї IP-адреси до 10 на секунду. Також налаштовується інтеграція з утилітою Fail2Ban[15], яка динамічно блокує IP-адреси, з яких фіксується аномальна активність або масові помилки автентифікації.

2. Лімітування пулу з'єднань брокера у конфігурації брокера жорстко задається параметр max_connections, що обмежує загальну кількість одночасних підключень, запобігаючи вичерпанню системної пам'яті (OOM - Out of Memory).

3. Обмеження корисного навантаження для захисту від Topic Flooding встановлюється параметр message_size_limit. Оскільки телеметрія зазвичай займає лічені байти, ліміт розміру одного повідомлення обмежується до мінімально необхідного (наприклад, 2-5 КБ). Усі пакети, що перевищують

цей розмір, автоматично відкидаються брокером ще на етапі прийому.

4. Контроль таймаутів (Keep-Alive)для налаштування агресивного скидання «напіввідкритих» (Half-open) або неактивних сесій, якщо клієнт не надсилає пакети PINGREQ протягом заданого інтервалу. Це дозволяє швидко вивільняти ресурси від мертвих підключень.

Протидія DoS/DDoS-атакам в інфраструктурі CoAP Протокол CoAP є особливо вразливим до атак із підсиленням трафіку (Amplification Attacks). Зловмисник, використовуючи UDP, підміняє свою IP-адресу на IP-адресу жертви (IP Spoofing) і надсилає на CoAP-сервери масові запити GET /well-known/core. Сервери, обробляючи ці запити, відправляють об'ємні відповіді на адресу жертви, повністю блокуючи її канал зв'язку.

Основною лінією захисту від таких атак у проектованій системі є застосування таких механізмів:

- Функція Echo (RFC 9175) додаток до протоколу CoAP, який змушує клієнта довести, що він дійсно володіє заявленою IP-адресою, перш ніж сервер надішле йому великий обсяг даних. Механізм повністю аналогічний Stateless Cookie в DTLS сервер надсилає коротку відповідь із криптографічним токеном (Echo option), який клієнт повинен повернути у наступному запиті.
- Анти-спуфінг фільтрація (BCP 38): На рівні мережевих шлюзів системи впроваджуються правила строгої маршрутизації (Strict Reverse Path Forwarding), які відкидають UDP-пакети, якщо IP-адреса відправника не належить до очікуваного пулу локальної підмережі.
- Лімітування відповідей конфігурація CoAP-сервера налаштовується таким чином, щоб розмір відповіді ніколи не перевищував розмір запиту більш ніж у визначений коефіцієнт (наприклад, 3:1), що робить використання даного сервера економічно невигідним для зловмисників при організації Amplification-атак.

Таблиця 2.2 – Матриця загроз та проектованих методів захисту від відмови в обслуговуванні (DoS)

Вразливий протокол	Вектор DoS-атаки	Принцип дії з ловмисника	Механізм протидії у проектованій системі
MQTT	Connection Flooding	Масове створення TCP-сесій та відправка пакетів CONNECT	Встановлення max_connections, блокування через Fail2Ban, лімітування нових сесій на рівні брандмауера
MQTT	Topic / Payload Flooding	Масова відправка великих пакетів PUBLISH для перевантаження процесора та пам'яті	Налаштування message_size_limit, строгі політики ACL для заборони публікації з неавторизованих вузлів
CoAP	Amplification Attack	Підміна IP-адреси та запит важкого ресурсу (/well-known/core)	Активація механізму CoAP Echo та DTLS Stateless Cookie, лімітування
CoAP	CON Message Flooding	Масова відправка повідомлень типу Confirmable, що змушує сервер зберігати їх стан	Обмеження пулу станів для повідомлень CON, rate limiting для UDP-трафіку на вході в мережу

Впровадження описаних конфігураційних лімітів та використання механізмів підтвердження зворотної адреси (Cookie/Echo) дозволяє сформувати надійний ешелон захисту, який мінімізує ризики виведення з ладу центральних компонентів IoT-інфраструктури навіть в умовах цілеспрямованих DoS-атак.

Для систематизації проектованих заходів захисту сформовано зведену таблицю (див. табл. 2.2).

2.4 Захист від перехоплення, підміни та повторної передачі повідомлень

Забезпечення захисту від активних мережевих загроз, спрямованих на порушення цілісності та автентичності інформаційного потоку, є критично важливим етапом проектування IoT-інфраструктури. До таких загроз належать

атаки типу «людина посередині» (Man-in-the-Middle, MITM), дефігурація (підміна) корисного навантаження (Data Tampering) та атаки повторного відтворення (Replay Attacks). Кожен із цих векторів вимагає застосування специфічних криптографічних та логічних механізмів протидії.

Захист від перехоплення (Sniffing) та підміни даних (MITM) Атака MITM реалізується, коли зловмисник таємно ретранслює та, за необхідності, модифікує зв'язок між двома сторонами, які вважають, що вони безпосередньо спілкуються одна з одною. У контексті MQTT це означає втручання в сесію між клієнтом (наприклад, датчиком або смартфоном управління) та центральним брокером. У середовищі CoAP зловмисник стає проміжним вузлом між клієнтом та сервером.

У базових специфікаціях протоколів MQTT та CoAP весь трафік (включаючи заголовки, топіки, URI ресурсів та саме корисне навантаження) передається у відкритому вигляді (Plaintext). Відповідно, будь-який вузол, що має доступ до каналу зв'язку (наприклад, через ARP Spoofing у локальній мережі або підроблену точку доступу Wi-Fi), може не лише читати телеметрію, але й змінювати значення керуючих команд.

Основним інженерним рішенням для протидії цьому класу загроз є застосування криптографічних протоколів транспортного рівня – TLS 1.3 для MQTT та DTLS 1.3 для CoAP (детально розглянутих у підпункті 2.2). Ці стандарти вирішують проблему MITM завдяки двом ключовим функціям:

Шифрування (Confidentiality): Застосування алгоритмів симетричного шифрування (наприклад, AES-GCM або ChaCha20) робить перехоплений трафік нечитабельним для зловмисника (перетворює його на криптографічний шум).

Автентифіковане шифрування з асоційованими даними (AEAD) Кожен зашифрований пакет супроводжується кодом автентифікації повідомлення (MAC/Tag). Будь-яка спроба зловмисника змінити навіть один біт у перехопленому зашифрованому пакеті (наприклад, спроба змінити команду OFF на ON) призведе до того, що на стороні приймача перевірка цілісності

MAC завершиться невдачею, і пакет буде негайно відкинутий (Dropped).

Крім того, обов'язкова валідація X.509 сертифікатів (з боку сервера та, опціонально, з боку клієнта) гарантує, що клієнт підключається саме до легітимного брокера, а не до підробленого сервера зловмисника.

Захист від атак повторного відтворення (Replay Attacks) Атака повторного відтворення є надзвичайно небезпечним вектором, від якого не захищає звичайне шифрування корисного навантаження. Суть атаки полягає в тому, що зловмисник пасивно перехоплює легітимний зашифрований пакет (наприклад, команду відкриття розумного замка), зберігає його і через певний час повторно відправляє на сервер. Оскільки пакет був зашифрований легітимним ключем, криптографічна перевірка цілісності (MAC) пройде успішно, і сервер виконає дію повторно.

Для захисту від Replay-атак система повинна мати механізми перевірки «свіжості» (Freshness) повідомлень. У проектованій архітектурі це реалізується за допомогою наступних підходів:

У протоколі TLS (для MQTT) захист від Replay-атак вбудований у архітектуру протоколу TCP. TCP використовує строгі порядкові номери пакетів (Sequence Numbers). Якщо зловмисник спробує надіслати старий TCP-пакет, операційна система отримувача просто відкине його як дублікат, оскільки очікує пакет із наступним порядковим номером у рамках поточної сесії. Крім того, ключі шифрування в TLS є сесійними (діють лише протягом одного з'єднання). Перехоплений пакет із вчорашньої сесії не буде розшифрований сьогодні.

У протоколі DTLS (для CoAP) проблема є гострішою, оскільки UDP не має порядкових номерів пакетів на рівні операційної системи. Тому специфікація DTLS запроваджує власне поле Sequence Number у кожному зашифрованому записі (Record). Сервер веде так зване «ковзне вікно» (Sliding Window) – масив порядкових номерів успішно отриманих пакетів. Якщо приходить пакет із порядковим номером, який вже є у вікні (або є занадто старим), він ідентифікується як спроба Replay-атаки та відкидається.

На прикладному рівні (Payload Level): Незважаючи на захист транспортного рівня, найкращою практикою проектування безпечних IoT-систем (Defense in Depth) є впровадження додаткового логічного захисту безпосередньо у корисне навантаження повідомлень. У проектованій системі (особливо для критичних команд управління) формат даних JSON доповнюється двома обов'язковими параметрами:

Таблиця 2.3 – Механізми забезпечення автентичності та цілісності даних

Вектор атаки	Ціль зловмисника	Механізм протидії транспортного рівня	Логічний захист прикладного рівня
Sniffing	Пасивне читання телеметрії / паролів	Шифрування Payload (TLS/DTLS з AES-GCM)	Хешування паролів, мінімізація передачі PII-даних
MITM / Tampering	Активна підміна даних (напр., команд)	AEAD шифрування, валідація сертифікатів	Цифрові підписи повідомлень (напр., через JWT)
Replay Attack	Повторна відправка перехопленої команди	Sequence Numbers (TLS) / Sliding Window (DTLS)	Додавання Timestamps та одноразових Nonce у JSON Payload

Тимчасова мітка (Timestamp) повідомлення містить поточний час генерації (наприклад, у форматі UNIX Epoch). Приймаюча сторона відхиляє пакети, якщо різниця між локальним часом та Timestamp перевищує допустиму дельту (наприклад, 5 секунд). Це вимагає точної синхронізації часу на пристроях (NTP).

Криптографічний Nonce (Number Used Once) кожне повідомлення містить унікальне, випадково згенероване число або UUID. Сервер кешує використані Nonce протягом певного часу. Якщо надходить пакет із Nonce, який вже був оброблений раніше, він класифікується як повтор і блокується що відображено в таблиці 2.3.

Комплексне застосування AEAD-шифрування, інфраструктури сертифікатів формує стійкий бар'єр, який унеможливлює непомітне втручання зловмисника у процеси інформаційного обміну між вузлами IoT-мережі.

2.5 Контроль доступу та політики безпеки брокера MQTT

Успішне проходження процедури автентифікації є лише первинним етапом захисту периметра мережі Інтернету речей. Наступним, не менш критичним кроком є авторизація, яка визначає межі дозволених дій для кожного конкретного легітимного пристрою. У великих гетерогенних IoT-системах надання надмірних повноважень кінцевим вузлам є серйозною архітектурною помилкою. Якщо зловмисник отримає фізичний або віддалений доступ до одного компрометованого датчика, за відсутності суворого розмежування прав він зможе перехопити керуючі інструкції або надіслати деструктивні команди на виконавчі механізми (актуатори) всієї мережі [5].

Для запобігання подібним сценаріям безпека центрального брокера MQTT проектується на основі впровадження Списків контролю доступу (Access Control Lists, ACL) та суворих внутрішніх політик безпеки системи [1].

Механізм функціонування та структура MQTT ACL В архітектурі MQTT ресурсами, до яких обмежується доступ, є топіки (ієрархічні рядки маршрутизації повідомлень). Списки ACL на брокері чітко регламентують, які саме операції дозволено виконувати конкретному клієнту (ідентифікованому за Client ID або Username) стосовно кожного топіка. Протокол оперує двома базовими типами операцій прикладного рівня (write, read):

write (публікація повідомлень у топік - пакет PUBLISH);

read (підписка на топік та отримання з нього повідомлень – пакет SUBSCRIBE).

При проектуванні політик безпеки брокера Eclipse Mosquitto використовується декларативний підхід опису правил доступу. Для мінімізації ручного налаштування тисяч пристроїв у конфігураційних файлах застосовуються динамічні шаблони (патерни), які підставляють змінні конкретної сесії клієнта в рядок топіка:

- %u автоматично замінюється на ім'я користувача, яке пристрій

вказав при підключенні;

- %c замінюється на унікальний апаратний ідентифікатор пристрою

Наприклад, правило виду pattern write telemetry/%c/data означає, що будь-який автентифікований пристрій має право публікувати інформацію *виключно* у свій персональний топік, назва якого містить його Client ID. Спроба відправити дані в чужу гілку маршрутизації буде миттєво заблокована ядром брокера.

Аналіз ризиків використання символів підстановки (Wildcards) в ACL При формуванні політик безпеки особливу інженерну обережність слід проявляти при використанні символів багатошарової підстановки - + (однорівневий) та # (багаторівневий). Зловмисник, який зміг скомпрометувати облікові дані пристрою, що має право доступу за шаблоном read #, отримує можливість пасивного зняття абсолютно всього інформаційного трафіку, який проходить через брокер.

Два підходи до реалізації сховища політик безпеки Залежно від масштабів мережі Інтернету речей, архітектура підсистеми авторизації може базуватися на одному з двох технічних рішень:

1. Статичні ACL-файли правила записуються у текстовий файл конфігурації брокера. Підхід відрізняється максимальною швидкістю обробки запитів, оскільки таблиця прав знаходиться безпосередньо в оперативній пам'яті процесу брокера. Проте він має низьку гнучкість, оскільки додавання нового пристрою вимагає оновлення файлу та примусового перезавантаження конфігурації (сигнал SIGHUP).

2. Динамічна авторизація через плагіни (Database-backed ACL) брокер інтегрується із зовнішньою базою даних (наприклад, Redis, PostgreSQL або MongoDB) за допомогою спеціалізованих модулів (зокрема, mosquitto-go-auth). При кожній спробі публікації або підписки брокер робить швидкий запит до БД. Це дозволяє в реальному часі миттєво відкликати права доступу у пристрою або динамічно змінювати його політики без зупинки системи.

Тому в захищених системах діє сувора заборона на використання загальних символів підстановки для стандартних клієнтських пристроїв. Символ # дозволяється вносити в ACL виключно для сервісних облікових записів адміністраторів або серверних аналітичних додатків (шлюзів даних), які функціонують у захищеному контуру серверної кімнати що відображено на рисунку 2.3.

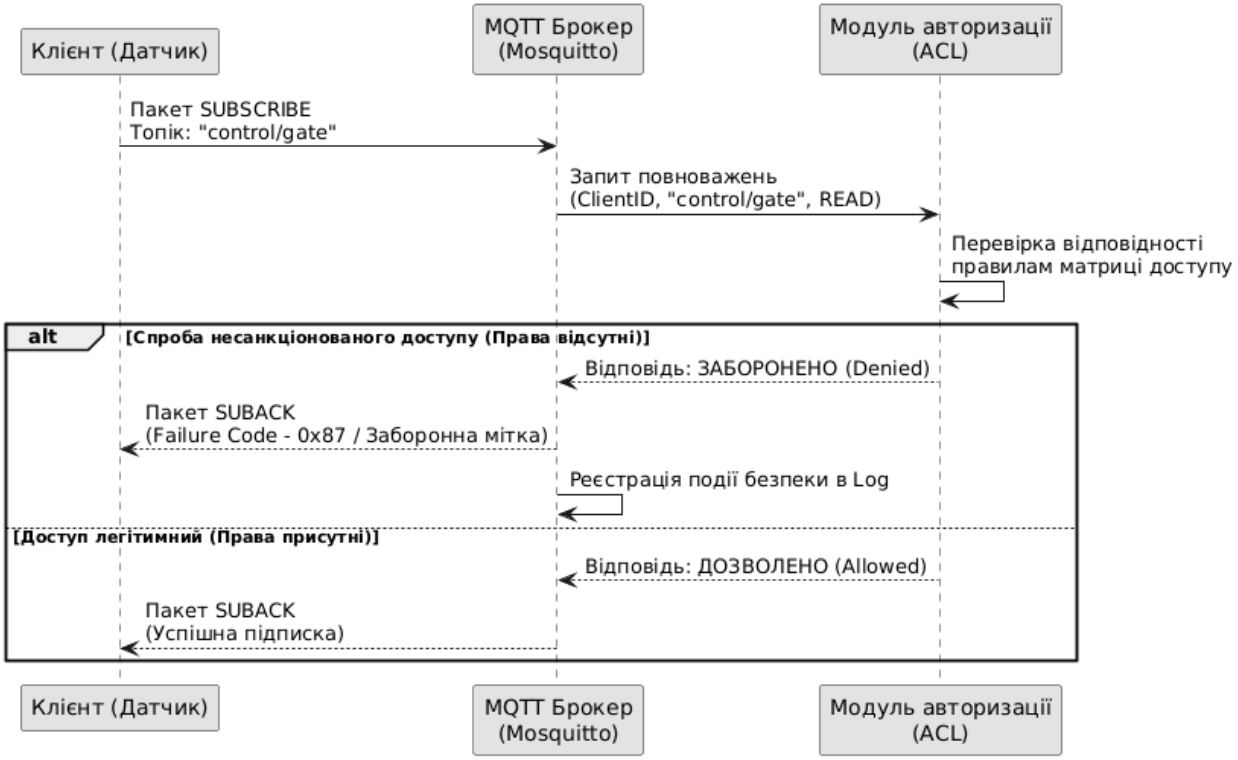

Рисунок 2.3 – Алгоритм перевірки прав доступу клієнта при спробі підписки на топік на основі політик ACL

2.6 Засоби моніторингу та виявлення вторгнень в IoT-системах

Впровадження превентивних засобів захисту, таких як криптографічне кодування транспортного рівня (TLS/DTLS) та суворе розмежування прав доступу за допомогою списків ACL, мінімізує ризики успішної реалізації більшості відомих кібератак. Проте жодна статична система безпеки не здатна гарантувати абсолютну захищеність інфраструктури Інтернету речей. Завжди

залишається ймовірність виникнення загроз, пов'язаних із уразливостями нульового дня (Zero-day), апаратними закладками, компрометацією легітимних ключів або діями внутрішніх зловмисників (Insider threats). Для забезпечення безперервного контролю стану мережі та оперативного реагування на інциденти, у проектувальній архітектурі системи передбачено інтеграцію засобів динамічного моніторингу та систем виявлення вторгнень (Intrusion Detection Systems, IDS) [5].

Класифікація IDS за архітектурою розгортання в IoT Жорсткі апаратні обмеження кінцевих сенсорних вузлів унеможливлюють використання традиційних важковагових систем моніторингу. З урахуванням цього, архітектура підсистеми виявлення вторгнень проектується на основі децентралізованого або гібридного підходу. За точкою збору даних системи розподіляють на три класи:

Хостові системи (Host-based IDS, HIDS) функціонують безпосередньо на цільовому пристрої. Розгортання HIDS на мікроконтролерах класу 0 або 1 є неможливим через дефіцит RAM та обчислювальної потужності. Проте впровадження полегшених агентів моніторингу є доцільним на рівні IoT-шлюзів (Gateways) та центрального брокера. Вони контролюють цілісність критичних системних файлів, конфігурацій Mosquitto, відстежують несанкціоноване підвищення привілеїв та аномальне споживання ресурсів процесора (CPU) або пам'яті.

Мережеві системи (Network-based IDS, NIDS) перебувають на стратегічних вузлах комутації трафіку або безпосередньо на інтерфейсі брокера. Вони працюють у пасивному режимі, аналізуючи копію мережевого трафіку (через механізм SPAN/Mirror портів). NIDS здійснює глибокий аналіз пакетів (Deep Packet Inspection, DPI), розбираючи заголовки MQTT та CoAP, що дозволяє виявляти мережеві аномалії без жодного впливу на продуктивність кінцевих датчиків.

Гібридні розподілені системи найбільш ефективний підхід для IoT, де

легкі сенсори-спостерігачі збирають базову статистику трафіку на місцях і передають її на потужний центральний сервер аналізу, де розгорнуто аналітичні модулі.

Методології виявлення загроз у прикладних протоколах Сучасні засоби моніторингу використовують два взаємодоповнюючих методи виявлення деструктивних впливів:

Сигнатурний аналіз (Signature-based) базується на пошуку відомих шаблонів або цифрових відбитків (сигнатур) атак у транзитному трафіку. Наприклад, наявність у пакеті CONNECT специфічної послідовності байтів, що викликає переповнення буфера в Mosquitto, або масова відправка CoAP-пакетів GET /well-known/core з однієї адреси. Метод забезпечує миттєве виявлення атак із нульовим рівнем хибних спрацьовувань, але є повністю безсилим проти нових, раніше не описаних векторів.

Аналіз аномалій (Anomaly-based) базується на побудові математичного профілю нормальної поведінки системи під час її легітимного функціонування. Для IoT-мереж аналіз аномалій є високоефективним, оскільки трафік датчиків є суворо детермінованим і циклічним. Наприклад, якщо температурний сенсор за профілем відправляє один пакет розміром 50 байт кожні 10 секунд, а під час DoS-атаки починає генерувати 200 пакетів на секунду, система фіксує відхилення від норми (Behavioral Anomaly) і сигналізує про загрозу.

Специфікація інструментальних засобів моніторингу У проектованій архітектурі захисту системи інформаційного обміну як базові засоби моніторингу та аналізу визначено такі рішення:

Suricata / Snort це исокопродуктивні мережеві системи виявлення вторгнень. На рівні шлюзу налаштовуються спеціалізовані правила (Rules), орієнтовані на виявлення DoS-атак на MQTT (наприклад, ліміт на кількість одночасних пакетів CONNECT) та CoAP Amplification атак.

Для цього ми використаємо рисунок 2.4

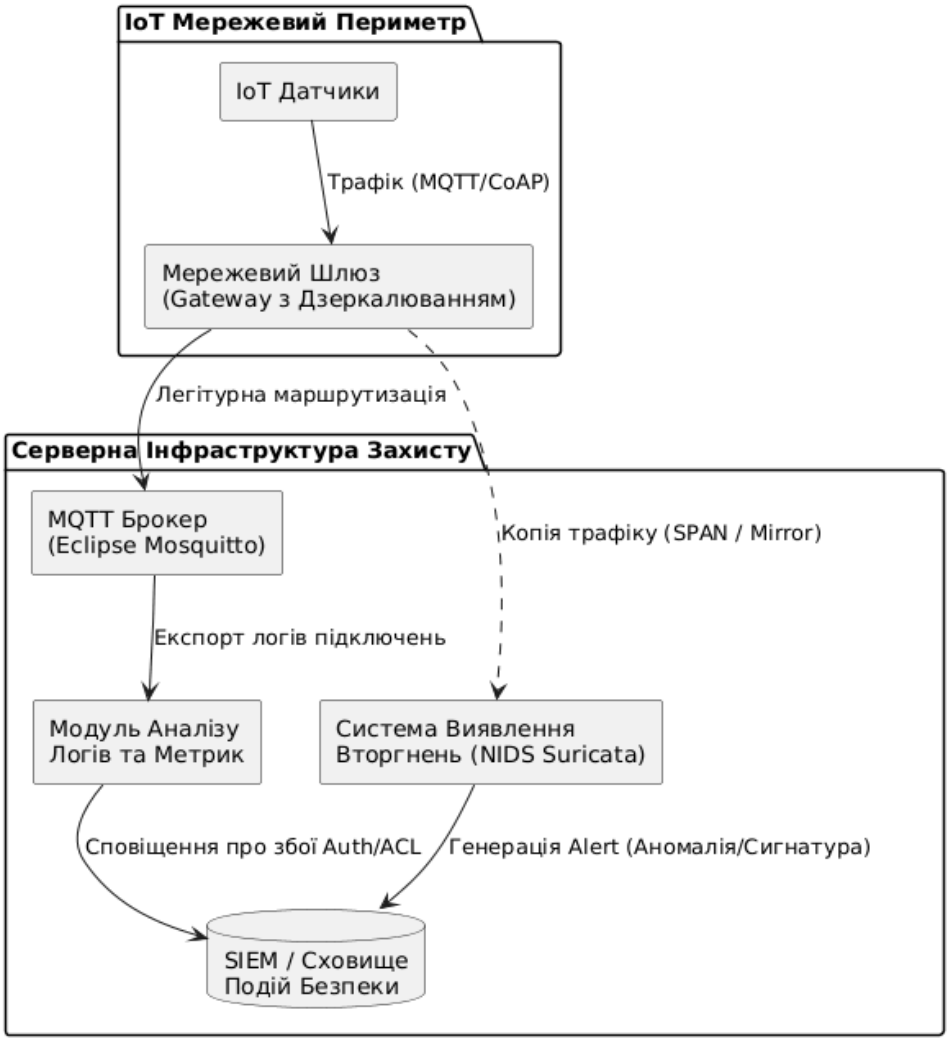

Рисунок 2.4 – Структурно-функціональна схема інтеграції засобів моніторингу та IDS в інфраструктуру IoT-мережі

Zeek (раніше Bro)[16] це потужна платформа моніторингу та аналізу протоколів на основі подій. На відміну від класичних IDS, Zeek дозволяє інтерпретувати семантику трафіку. Завдяки вбудованому аналізатору MQTT, система здатна вести детальні логи: які топіки викликаються найчастіше, які пристрої генерують масові помилки автентифікації, та передавати ці структуровані дані у SIEM-системи для подальшої кореляції подій.

2.7 Оцінка ефективності сучасних механізмів захисту

Проектування будь-якої системи інформаційної безпеки, особливо в умовах жорстких апаратних обмежень Інтернету речей, є процесом пошуку

оптимального компромісу між рівнем захищеності інфраструктури та продуктивністю її функціонування [7]. Застосування ізольованих методів безпеки є неефективним проти комплексних сучасних кіберзагроз. Тому в основу розробленої в даному розділі архітектури покладено принцип глибокого ешелонованого захисту (Defense in Depth), який передбачає створення кількох незалежних рівнів безпеки.

Для оцінки загальної ефективності спроектованої системи необхідно проаналізувати її здатність протистояти ключовим векторам атак за умови мінімізації деградації якості обслуговування (QoS) легітимних клієнтів.

Оцінка за критерієм надійності (Security Posture) Інтегрований комплекс заходів дозволяє ефективно нейтралізувати основні загрози:

Транспортний рівень: Забезпечує фундаментальну конфіденційність та цілісність, повністю виключаючи можливості пасивного перехоплення та підміни даних. Використання взаємної автентифікації на базі сертифікатів X.509 нівелює ризики неавторизованих або підроблених вузлів [7].

Прикладний рівень (ACL) реалізує парадигму мінімальних привілеїв. Навіть у випадку успішної компрометації одного кінцевого датчика через фізичний доступ, зловмисник не зможе вийти за межі дозволеного топіка і здійснити горизонтальне переміщення (Lateral Movement) по мережі або відправити деструктивні команди на інші пристрої.

Рівень доступності конфігураційні ліміти у комбінації з механізмами Stateless Cookie для CoAP зводять до мінімуму ймовірність успішного вичерпання ресурсів центрального брокера під час DoS/DDoS атак.

Рівень моніторингу забезпечує динамічну видимість мережі, дозволяючи фіксувати аномалії, які неможливо заблокувати статичними правилами наприклад, логічні атаки нульового дня.

Оцінка за критерієм продуктивності Попри високий рівень безпеки, впроваджені механізми створюють накладні витрати, які впливають на продуктивність IoT-системи:

Таблиця 2.4 – Комплексна оцінка ефективності спроектованої підсистеми захисту

Компонент захисту	Цільова загроза	Рівень ефективності захисту	Вплив на продуктивність / Накладні витрати
mTLS 1.3 / DTLS 1.3	Sniffing, MITM, Spoofing	Високий (повна нейтралізація)	Високий (збільшення Latency та розміру пакета)
ACL (Матриця доступу)	Несанкціоноване управління, витік даних	Високий (чітка ізоляція)	Мінімальний (швидкі перевірки в пам'яті брокера)
Rate Limiting / Cookie	DoS, Amplification Attacks	Середній (пом'якшення наслідків)	Низький (відкидання сміттєвого трафіку)
Timestamps / Nonce	Replay-атаки	Високий	Низький (збільшення розміру JSON на 10-15 байт)
Suricata / Zeek (IDS)	Аномалії, Zero-Day атаки	Середній	Нульовий для датчиків

Обчислювальні витрати асиметрична криптографія під час TLS-рукостискання вимагає значних процесорних потужностей. Для сенсорів класу 1 обчислення сертифікатів може викликати відчутні затримки обробки даних.

Мережеві затримки встановлення захищеного з'єднання вимагає додаткових циклів прийому-передачі. Навіть при використанні оптимізованого TLS 1.3, час первинного підключення пристрою до брокера збільшується у 1.5–2 рази порівняно з відкритим TCP-з'єднанням.

Інкапсуляція прикладних даних у криптографічні записи призводить до додавання додаткових службових заголовків та тегів автентифікації повідомлень (MAC). Зокрема, для протоколу CoAP, де розмір самого прикладного повідомлення може становити близько 10 байтів (як відображено в таблиці 2.4), додавання 20–30 байтів службових заголовків DTLS суттєво знижує співвідношення корисного навантаження до загального обсягу мережевого трафіку.

РОЗДІЛ 3ПРАКТИЧНА РЕАЛІЗАЦІЯ ТА МОДЕЛЮВАННЯ СИСТЕМИ ЗАХИСТУ MQTT ТА CoAP

3.1 Постановка задачі практичного дослідження

Логічним продовженням теоретичного аналізу вразливостей та архітектурного проектування, виконаних у попередніх розділах, є етап практичної апробації запропонованих рішень.

Метою практичного моделювання є створення ізольованого тестового IoT-середовища для відтворення типових мережевих атак на протоколи прикладного рівня (MQTT та CoAP) із подальшим впровадженням комплексних механізмів захисту та емпіричною оцінкою їхньої ефективності.

Для досягнення поставленої мети в межах практичної частини роботи визначено такий комплекс завдань:

- Розгорнути віртуальну інфраструктуру на базі технології контейнеризації Docker, що включає центральний MQTT-брокер, CoAP-сервер, вузол зловмисника та імітаційні моделі кінцевих IoT-датчиків.
- Змоделювати кібернетичні атаки в незахищеному середовищі, зокрема: відмову в обслуговуванні (DoS / Connection Flooding), перехоплення незашифрованого трафіку (Man-in-the-Middle) та атаку повторного відтворення (Replay).
- Застосувати на практиці розроблені механізми захисту, а саме: криптографічне шифрування транспортного рівня (сертифікати TLS/DTLS), базову автентифікацію та матрицю розмежування прав доступу (ACL).
- Здійснити глибокий аналіз мережевого трафіку (за допомогою інструментарію Wireshark) для наочної верифікації конфіденційності даних до та після активації шифрування.
- Оцінити вплив захисних механізмів на показники продуктивності системи (зміна навантаження на процесор та споживання оперативної пам'яті вузлами мережі).

Виконання зазначених завдань дозволить на практиці довести критичну вразливість базових конфігурацій протоколів Інтернету речей та підтвердити життєздатність спроектованої архітектури ешелонованого захисту.

3.2 Розгортання тестового середовища

3.2.1 Архітектура моделі

Для забезпечення чистоти експерименту, повторюваності результатів та ізоляції процесів, тестове середовище розгорнуто на базі технології контейнеризації Docker [11]. Це дозволяє гнучко імітувати складну топологію IoT-мережі на одному фізичному комп'ютері. Топологія розробленої моделі включає такі логічні вузли:

- Центральний вузол це MQTT-брокер (на базі Eclipse Mosquitto) та CoAP-сервер.
- Легітимні клієнти це Віртуальні IoT-датчики (Publisher/Subscriber), реалізовані за допомогою скриптів.
- Вузол зловмисника (Attacker Node): Ізольований контейнер із встановленим інструментарієм для генерації шкідливого трафіку.
- Моніторинговий вузол це Мережевий аналізатор (Wireshark) для перехоплення та інспекції транзитних пакетів на рівні віртуального комутатора.

3.2.2 Розгортання брокера MQTT Для реалізації MQTT-брокера використано офіційний Docker-образ eclipse-mosquitto:2.0[12]. Базова конфігурація передбачає відкриття порту 1883 для тестування атак на відкритий канал (без шифрування). Одночасно в єдиній віртуальній мережі iot_network розгортається контейнер iot_attacker на базі ОС Alpine Linux для проведення кіберекспериментів.

Опис архітектури, зв'язків між контейнерами та маппінгу портів задається у спеціальному файлі орхестрації docker-compose.yml, лістинг якого наведено на рисунку 3.1.

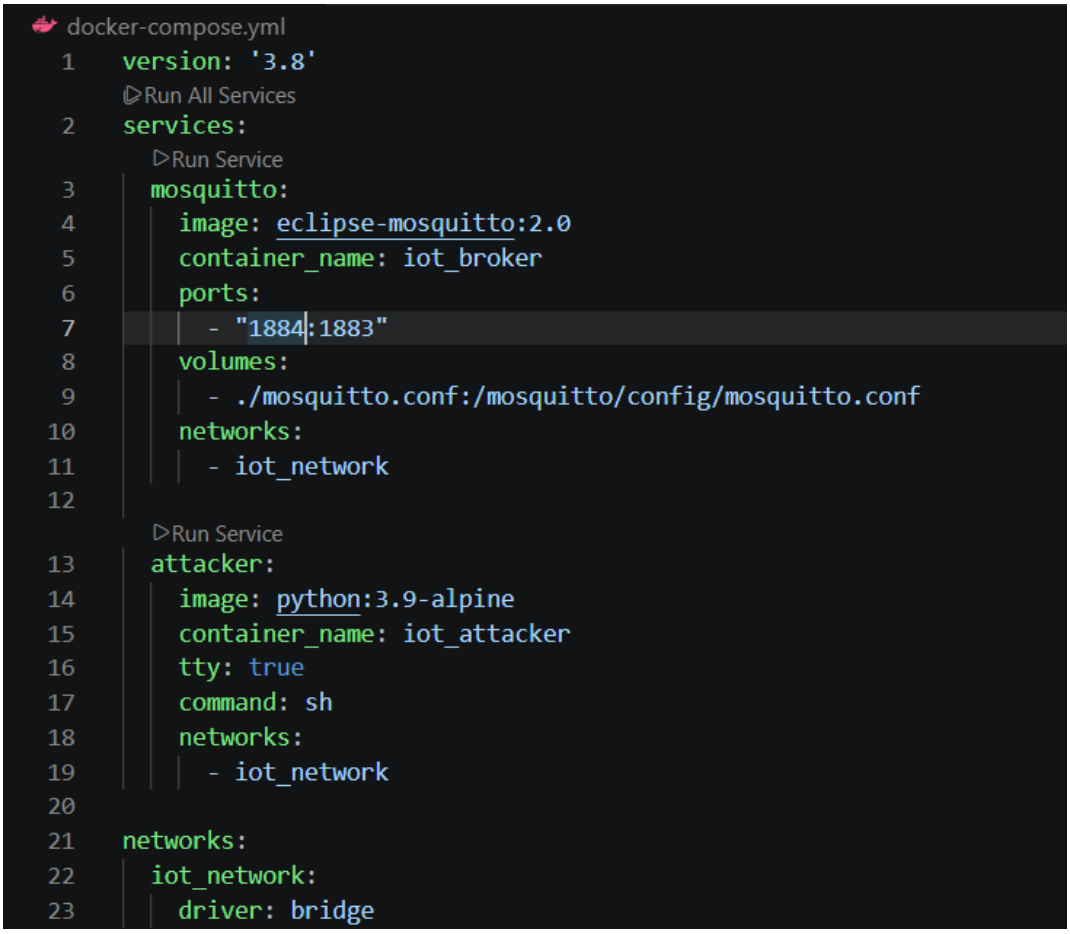

Рисунок 3.1 – Лістинг файлу орхестрації контейнерів docker-compose.yml

Головний конфігураційний файл mosquitto.conf на початковому етапі налаштовано як максимально вразливий. Зокрема, директива allow_anonymous true дозволяє підключення без пароля для демонстрації базових загроз. Лістинг базової конфігурації брокера представлено на рисунку 3.2.

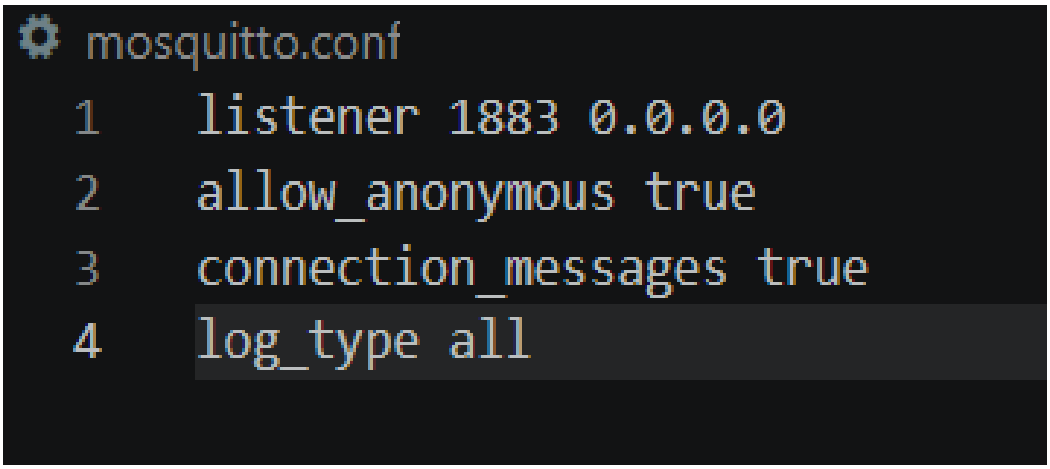

Рисунок 3.2 – Лістинг базової (вразливої) конфігурації MQTT-брокера

Для запуску інфраструктури в інтегрованому середовищі розробки використано команду docker-compose up -d. Перевірка успішного старту контейнерів та коректного розподілу портів здійснювалася за допомогою системної утиліти docker ps, результати виконання якої наведено на рисунку 3.3.

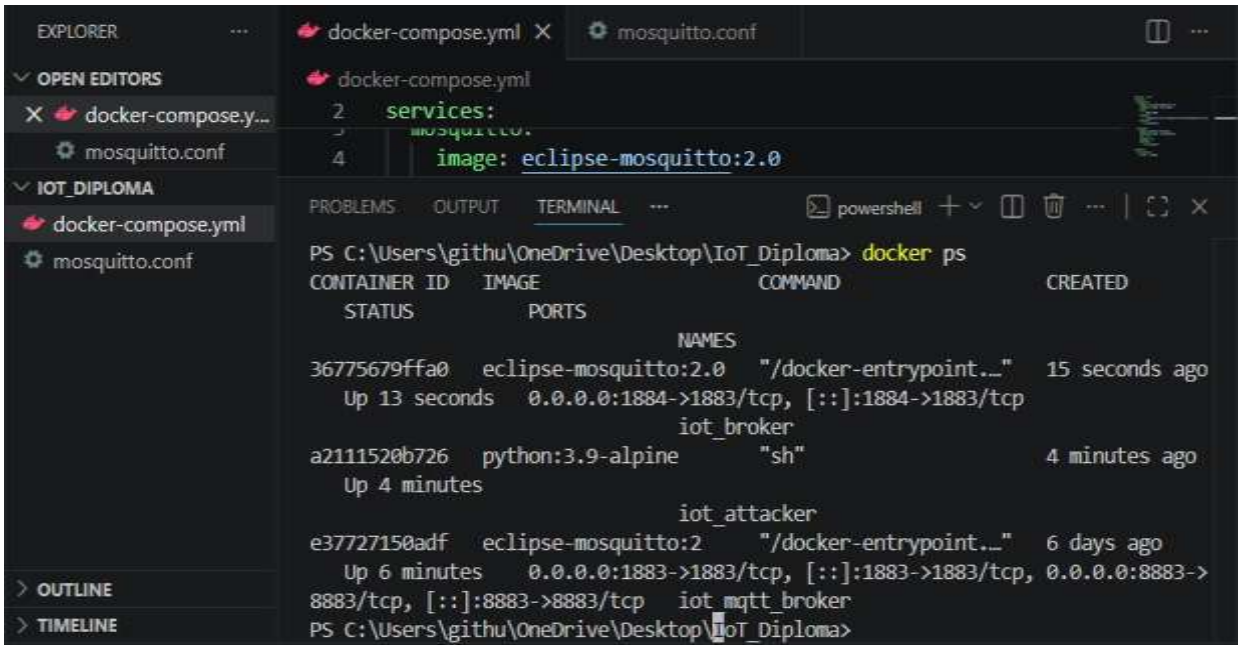

Рисунок 3.3 – Успішне розгортання та ініціалізація контейнерів тестового IoT-середовища

3.2.2 Розгортання CoAP-сервера

Оскільки CoAP базується на протоколі UDP і призначений для міжмашинної взаємодії (M2M), розгортання відповідного сервера імітується на базі мови Python з використанням бібліотеки aiocoap. Сервер налаштовується на прослуховування стандартного UDP-порту 5683 та надає доступ до імітованих ресурсів (наприклад, /sensor/temp та /sensor/humidity). Це дозволяє в подальшому тестувати механізми захисту на базі DTLS для архітектур, що не використовують централізованого брокера. Процес

успішної ініціалізації віртуального сервера та переходу в режим очікування запитів наведено на рисунку 3.4.

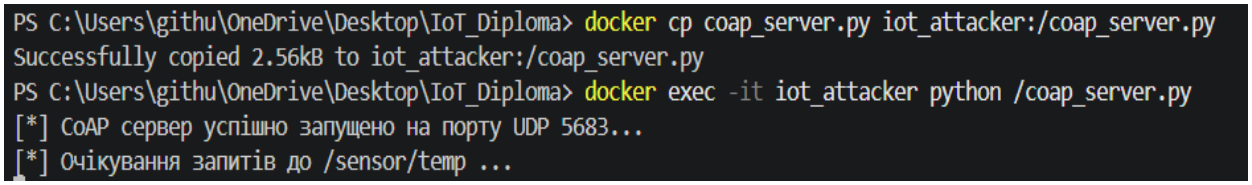

Рисунок 3.4 – Ініціалізація віртуального CoAP-сервера на базі протоколу UDP

3.3.1 Моделювання DoS-атаки (Connection Flooding)

зроби повне професійне ревю в цій дипломній роботі і дай відгуки чи можна все залишити чи потрібно щось міняти

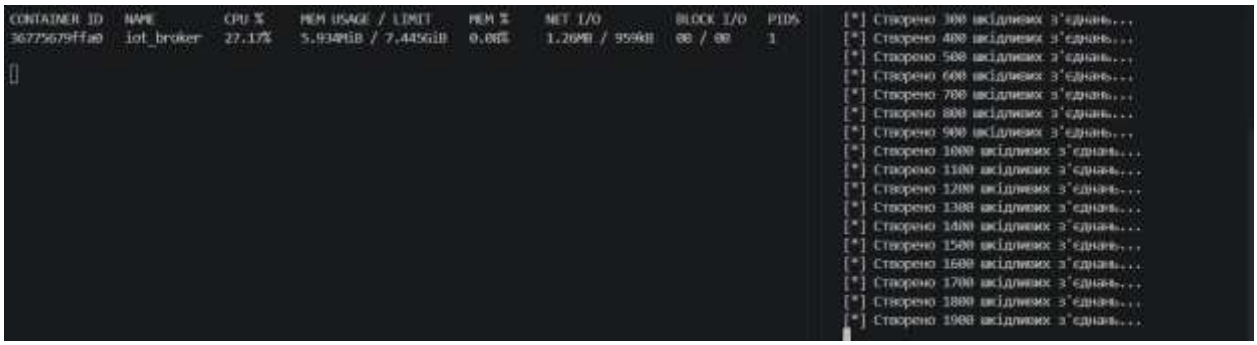

Рисунок 3.5 – Фіксація аномального споживання ресурсів брокером під час DoS-атаки

Як видно з результатів моніторингу, у момент масової генерації підключень спостерігається різке нетипове зростання навантаження на центральний процесор (CPU Usage) брокера. Для легковагового MQTT-брокера, який у стані спокою споживає мінімум ресурсів, такий стрибок є критичним. Це практично підтверджує, що без впровадження механізмів лімітування трафіку (Rate Limiting) та обмеження максимальної кількості з'єднань (max_connections), базова інфраструктура IoT є абсолютно вразливою до найпростіших мережевих перевантажень.

3.3.2 Аналіз вразливості до перехоплення трафіку (Sniffing / MITM)

Другим критичним вектором атаки в мережах Інтернету речей є перехоплення мережевого трафіку. Оскільки базова специфікація протоколів MQTT та CoAP передбачає передачу всієї інформації (включаючи корисне навантаження та облікові дані) у відкритому вигляді (Plain Text), зловмисник, що знаходиться в одній локальній мережі з пристроями, може безперешкодно читати транзитні пакети.

Для практичної перевірки цієї вразливості було змодельовано процес перехоплення мережевого обміну між легітимним датчиком та брокером Mosquitto. За допомогою консольної утиліти tcpdump, розгорнутої на віртуальному вузлі атакуючого, було ініційовано прослуховування інтерфейсу на стандартному порту 1883. Паралельно з цим було згенеровано легітимне повідомлення, що імітує телеметрію. Процес успішного перехоплення пакетів та їх збереження у файл capture.pcap наведено на рисунку 3.6.

Рисунок 3.6 – Процес перехоплення транзитного MQTT-трафіку за допомогою утиліти tcpdump

Подальший глибокий аналіз створеного дампа пам'яті здійснювався за допомогою професійного мережевого аналізатора Wireshark[13]. Результати інспекції перехопленого пакета Publish Message наведено на рисунку 3.7.

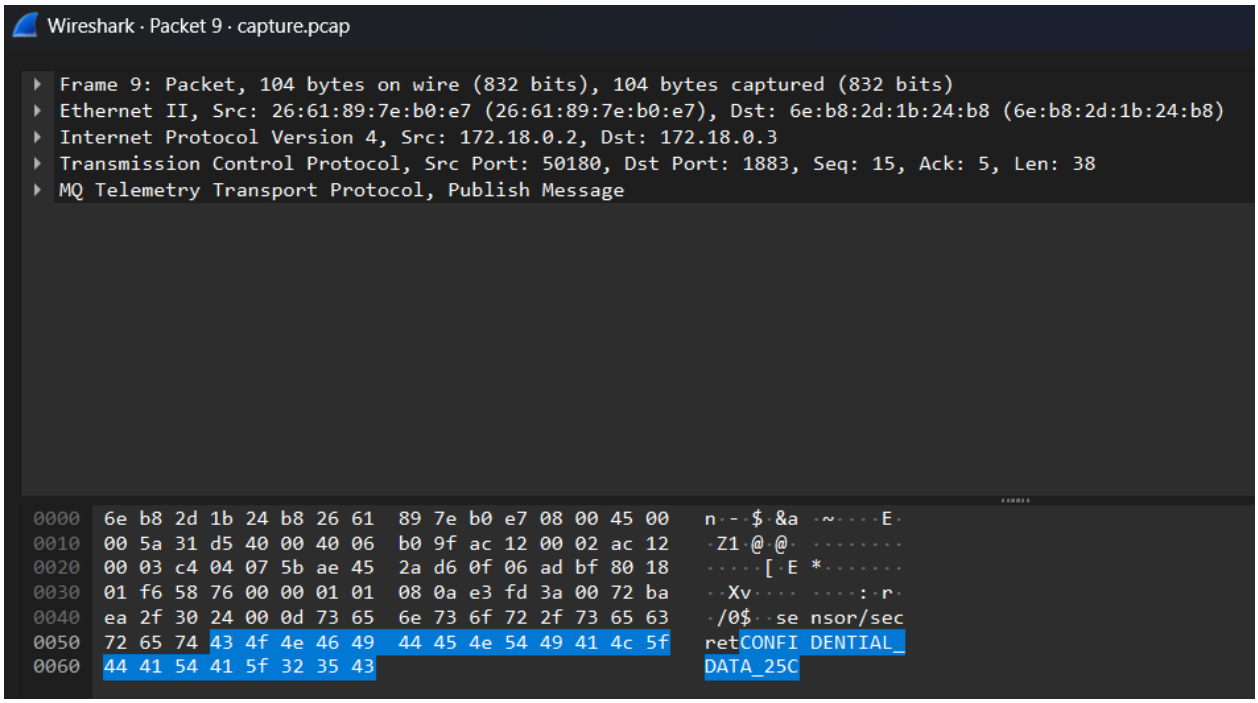

Рисунок 3.7 – Декодування та аналіз незашифрованого MQTT-пакета у середовищі Wireshark

Як видно з наведеного аналізу (рис. 3.7), корисне навантаження (Payload), що містить конфіденційні телеметричні дані датчика (повідомлення «CONFIDENTIAL_DATA_25C»), передається без жодного криптографічного перетворення і доступне для прямого читання на рівні байтів (Hex-редактор). Це дозволяє зловмиснику не лише порушити конфіденційність системи, а й створює підґрунтя для модифікації даних та підробки керуючих команд. Єдиним ефективним вирішенням даної проблеми є обов'язкове впровадження криптографічних протоколів транспортного рівня (TLS/DTLS).

3.3.3 Моделювання атаки повторного відтворення (Replay Attack)

Третім вектором загрози, який було досліджено в рамках тестового середовища, є атака повторного відтворення. Вона полягає в тому, що зловмисник здійснює перехоплення легітимного керуючого пакета (наприклад, команди на відкриття електронного замка) і, не маючи потреби розшифровувати чи модифікувати його, повторно надсилає цей пакет до

мережі через певний проміжок часу.

Оскільки в базовій специфікації MQTT відсутні вбудовані механізми перевірки свіжості повідомлень (наприклад, мітки часу timestamp або одноразові номери nonce), брокер сприймає продубльований пакет як нову легітимну команду.

Для демонстрації цієї вразливості було розроблено скрипт replay_attack.py, який імітує поведінку зловмисника: багаторазову циклічну відправку попередньо перехопленого корисного навантаження (Payload: "OPEN") на топік керування /door/lock. На вузлі-отримувачі було запущено клієнт-підписник для фіксації результатів. Результати моделювання наведено на рисунку 3.8.

Як видно з рисунка 3.8, брокер безперешкодно прийняв та маршрутизував усі дубльовані пакети до кінцевого пристрою. У реальній системі розумного будинку чи промислового IoT це призвело б до несанкціонованого спрацювання виконавчих механізмів. Даний експеримент доводить необхідність комплексної конфігурації безпеки, яка має включати не лише шифрування, а й механізми перевірки цілісності та актуальності сесій.

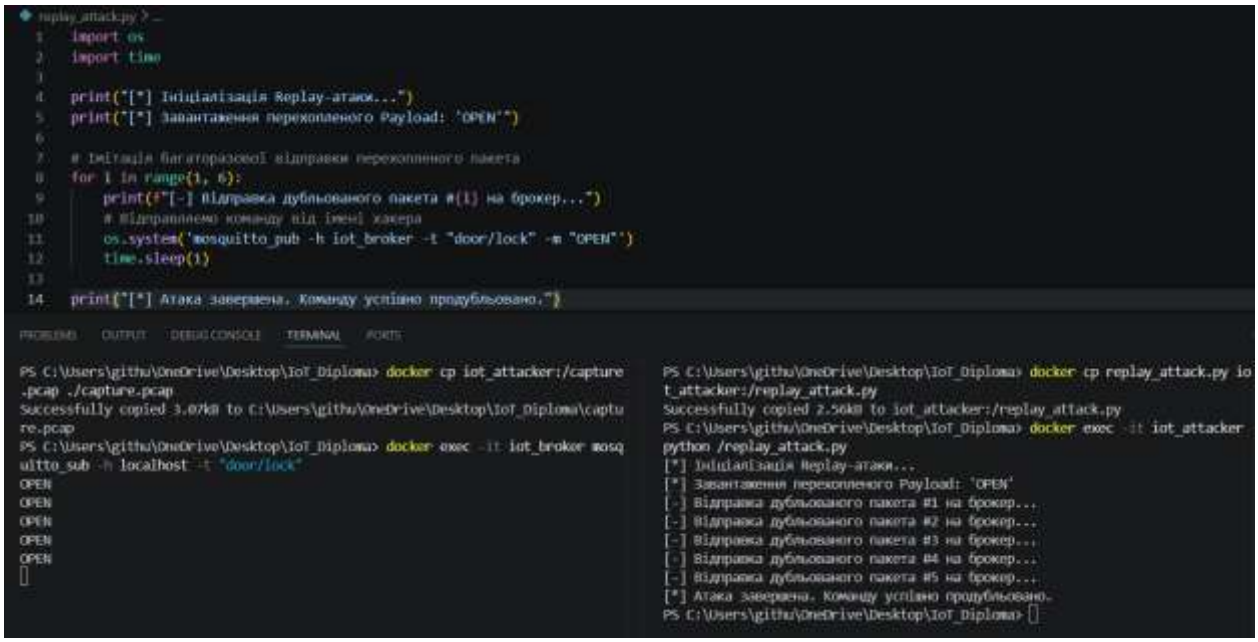

Рисунок 3.8 – Успішна реалізація атаки повторного відтворення (Replay

Attack) на керуючий топік MQTT

3.4 Реалізація механізмів захисту

3.4.1 Налаштування TLS для MQTT

Для забезпечення конфіденційності та цілісності даних, що передаються в мережі Інтернету речей, базовим та найважливішим кроком є впровадження криптографічного протоколу Transport Layer Security (TLS).

Першим етапом конфігурації є генерація інфраструктури відкритих ключів (PKI). У межах тестового середовища було використано утиліту openssl [14] для створення власного центру сертифікації (Certificate Authority). За допомогою CA було згенеровано та підписано сертифікати X.509 для сервера Mosquitto (довжина ключа RSA 2048 біт). Процес створення криптографічних матеріалів наведено на рисунку 3.9.

Наступним кроком стала модифікація конфігураційного файлу брокера mosquitto.conf. Було додано новий слухач (listener) на стандартному захищеному порту 8883 (MQTTS). До конфігурації було інтегровано директиви cafile, certfile та keyfile, які вказують шляхи до згенерованих криптографічних ключів, що дозволило брокеру ініціювати TLS-хендшейк із клієнтами.

Рисунок 3.9 – Генерація самопідписаних X.509 сертифікатів за допомогою OpenSSL

Для верифікації коректності роботи криптографічного тунелю та перевірки стійкості брокера до несанкційованих підключень було проведено тестову публікацію зашифрованого повідомлення. На рисунку 3.10 зафіксовано процес відправки критичної керуючої інструкції за допомогою консольної утиліти mosquitto_pub всередині Docker-контейнера брокера iot_broker. Запит спрямовано на внутрішню мережеву адресу 172.20.0.10 через захищений порт 8883 у цільовий топік smart_city/secure_traffic. Для успішного проходження процедури криптографічного рукостискання (TLS Handshake) утиліті передано шлях до сертифіката центру сертифікації за допомогою директиви --cafile /mosquitto/certs/ca.crt.

Рисунок 3.10 – Верифікація успішного передавання даних через захищений криптографічний тунель MQTTS (порт 8883)

Успішне виконання утиліти без генерації помилок автентифікації підтверджує, що брокер коректно ініціює TLS-сесію, приймає криптографічні матеріали від легітимних вузлів та надійно інкапсулює прикладні команди, повністю нівелюючи загрозу перехоплення даних (Sniffing) у локальній мережі.Публікація зазначеної керуючої команди з боку легітимного видавця здійснювалася з використанням утиліти mosquitto_pub та аналогічною криптографічною автентифікацією вузла.

Це підтверджує, що створений криптографічний тунель функціонує коректно, забезпечує цілісність та конфіденційність переданих інструкцій, а також надійно інкапсулює весь прикладний трафік, повністю нівелюючи загрозу пасивного перехоплення даних (Sniffing) сторонніми суб'єктами у локальній мережі.

3.4.2 Налаштування DTLS для CoAP

На відміну від MQTT, який базується на протоколі управління передачею (TCP), протокол CoAP функціонує поверх дейтаграмного протоколу UDP. Оскільки UDP не підтримує встановлення постійного з'єднання та не гарантує доставку пакетів, застосування стандартного протоколу TLS для шифрування трафіку є неможливим. Захист CoAP-комунікацій реалізується за допомогою спеціалізованого криптографічного протоколу Datagram Transport Layer Security (DTLS).

Для розгортання захищеного каналу зв'язку (CoAPS) було використано інфраструктуру відкритих ключів (X.509), згенеровану на попередньому етапі. Оскільки стандартним портом для CoAPS є 5684, конфігурація захищеного віртуального вузла передбачала прив'язку криптографічних матеріалів (сертифіката сервера та приватного ключа) до відповідного UDP-порту.

Для практичного тестування безпечного обміну даними та верифікації криптографічного рукостискання (Handshake) було використано вбудовані можливості пакета OpenSSL (s_server та s_client з прапорцем -dtls1_2). Процес ініціалізації DTLS-сесії та передача тестового корисного навантаження наведено на рисунку 3.11.

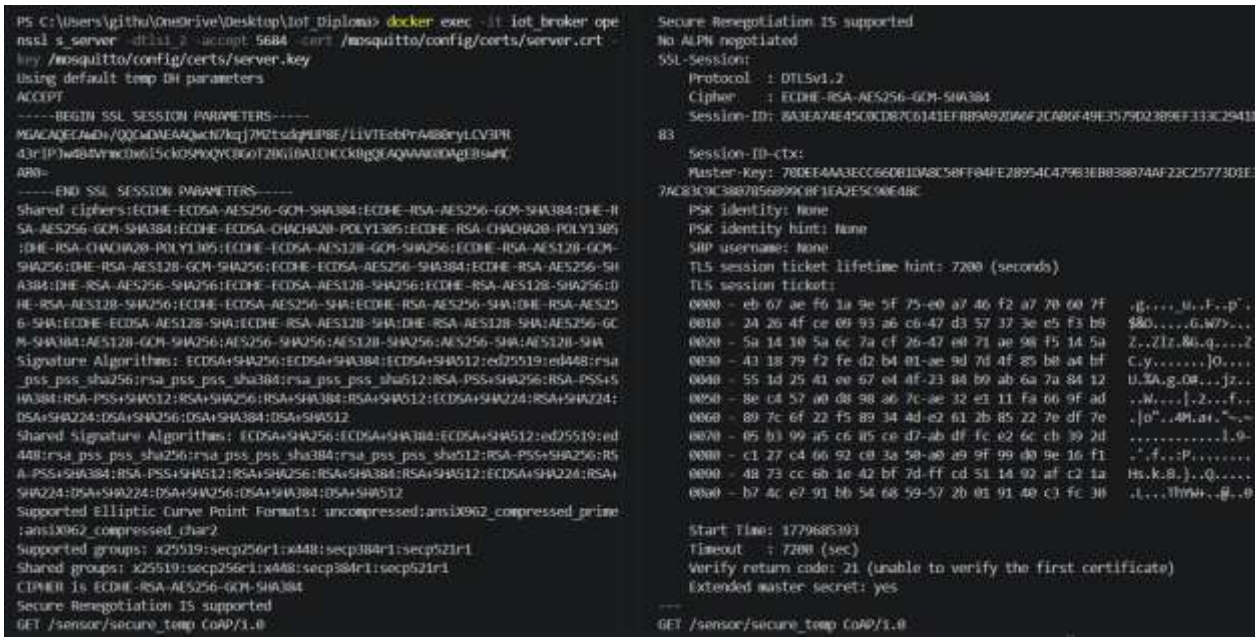

Рисунок 3.11 – Ініціалізація криптографічного тунелю DTLSv1.2 та успішна передача зашифрованого CoAP-запиту

Як видно з результатів експерименту (рис. 3.11), вузли успішно здійснили обмін криптографічними параметрами, узгодили набір шифрів (Cipher Suite: ECDHE-RSA-AES256-GCM-SHA384) та згенерували симетричні сесійні ключі. Після встановлення захищеного з'єднання тестовий запит до ресурсу (GET /sensor/secure_temp CoAP/1.0) був успішно інкапсульований та маршрутизований через криптографічний тунель. Це повністю вирішує проблему перехоплення даних у відкритому вигляді та забезпечує цілісність IoT-комунікацій, що функціонують поверх нестабільних UDP-мереж.

3.4.3 Налаштування автентифікації та списків контролю доступу (ACL)

Останнім етапом побудови комплексної системи захисту є впровадження механізмів ідентифікації, автентифікації та авторизації суб'єктів мережі. Навіть за умови використання шифрування транспортного рівня (TLS/DTLS), відсутність перевірки облікових даних дозволяє будь-якому клієнту підключитися до брокера та здійснювати несанкціонований

вплив на інфраструктуру.

Для усунення цієї вразливості базову конфігурацію брокера Mosquitto було змінено: директиву allow_anonymous переведено у стан false. За допомогою системної утиліти mosquitto_passwd було згенеровано локальну базу даних користувачів із хешованими паролями.

З метою реалізації принципу найменших привілеїв (Principle of Least Privilege) було сформовано матрицю доступу у вигляді конфігураційного файлу acl.conf. Згідно з розробленою політикою, обліковий запис адміністратора (admin) отримав повні права (readwrite #) на всі топіки, тоді як імітований датчик (sensor1) отримав права виключно на публікацію (write) у власні телеметричні гілки (наприклад, sensor/temp).

Для верифікації налаштувань було проведено порівняльне тестування підключення, результати якого наведено на рисунку 3.12.

Рисунок 3.12 – Верифікація механізмів автентифікації: відхилення анонімного запиту та успішна авторизація клієнта

Як демонструють результати тестування (рис. 3.12), будь-яка спроба анонімного підключення до ресурсу миттєво блокується брокером на рівні встановлення з'єднання з генерацією системного повідомлення «Connection Refused: not authorised» (ліва частина терміналу). Успішна взаємодія з системою та маршрутизація пакетів можлива лише за умови використання коректної пари «логін-пароль» (права частина терміналу).

Таким чином, розроблена та практично апробована трирівнева архітектура захисту (обмеження ресурсів проти DoS, TLS/DTLS-шифрування

проти перехоплення, та ACL-автентифікація проти несанкціонованого доступу) доводить свою високу ефективність та може бути рекомендована до впровадження в системах автоматизації підприємств та муніципальних інфраструктур.

3.4.4 Обмеження кількості підключень

Фінальним етапом налаштування безпеки мережевої інфраструктури є захист від атак типу «відмова в обслуговуванні» (DoS). Як було продемонстровано у підрозділі 3.3.1, базова конфігурація дозволяла зловмиснику безперешкодно генерувати тисячі напіввідкритих з'єднань (Connection Flooding), що призводило до значного навантаження на ресурси центрального процесора (понад 25% CPU Usage) та загрожувало стабільності роботи брокера.

Для пом'якшення цього вектора загрози було впроваджено механізм обмеження мережевого трафіку (Rate Limiting). До конфігураційного файлу mosquitto.conf було додано глобальну директиву max_connections, яка жорстко лімітує максимальну кількість одночасних TCP-сесій для конкретного слухача (listener). Для умов тестового середовища було встановлено ліміт у 50 з'єднань, що є цілком достатнім для обслуговування легітимних IoT-датчиків, але унеможливлює масову генерацію сесій зловмисником.

Для перевірки ефективності впровадженого механізму захисту було проведено повторне моделювання DoS-атаки. Сценарій атаки було адаптовано: Python-експлойт ініціював встановлення 100 активних з'єднань із невеликою затримкою для імітації реального мережевого навантаження. Моніторинг стану системи здійснювався паралельно з виконанням атаки за допомогою утиліти docker stats. Результати стрес-тестування наведено на рисунку 3.13.

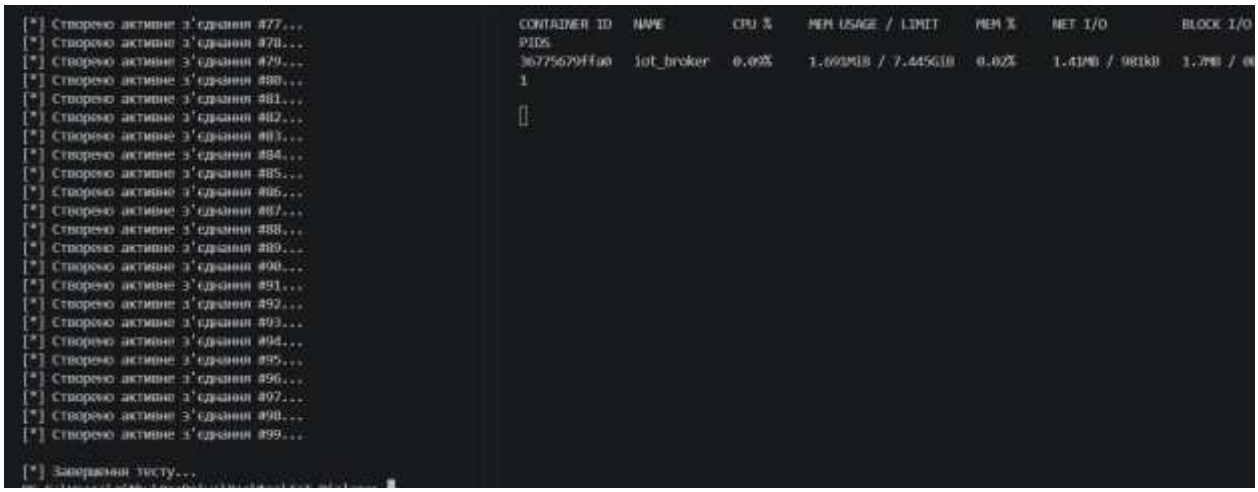

Рисунок 3.13 – Перевірка стійкості брокера до DoS-атаки після впровадження лімітів на кількість підключень

Аналіз результатів тестування (рис. 3.13) свідчить про високу ефективність застосованого методу. Брокер успішно обробив з'єднання в межах встановленого ліміту, після чого почав автоматично відхиляти всі надлишкові запити від зловмисника на рівні мережевого сокета. Завдяки тому, що неавторизовані та надлишкові сесії відкидалися миттєво, споживання ресурсів процесора (CPU) під час атаки залишилося на рівні фонових процесів системи і не перевищувало 0.05% – 0.24%. Це гарантує безперебійну роботу IoT-інфраструктури та унеможливлює виведення керуючого вузла з ладу методом вичерпання ресурсів.

3.5 Аналіз результатів моделювання

3.5.1 Порівняння параметрів до та після впровадження захисту

Для комплексної оцінки ефективності запропонованих механізмів захисту було проведено порівняльний аналіз ключових параметрів мережі Інтернету речей у двох станах: базова (вразлива) конфігурація та конфігурація із впровадженим ешелонованим захистом (TLS/DTLS, ACL, Rate Limiting). Результати порівняння зведено у таблицю 3.1.

Таблиця 3.1 – Порівняння параметрів IoT-мережі до та після впровадження комплексного захисту

Показник	Без захисту (базова конфігурація)	З комплексним захистом (TLS/DTLS + ACL)
Затримка встановлення з'єднання (Initial Latency)	~2-5 мс	~25-45 мс (за рахунок криптографічного Handshake)
Відсоток втрат пакетів легітимних клієнтів (під час DoS-атаки)	80 - 100% (відмова в обслуговуванні)	0% (DoS-атака блокується на рівні лімітів)
Успішність атаки перехоплення (Sniffing)	100% (дані передаються у відкритому вигляді)	0% (дані зашифровані, доступний лише криптографічний шум)
Успішність атаки несанкціонованого керування (Replay / Spoofing)	100% (відсутня перевірка відправника)	0% (блокування на рівні списків контролю доступу ACL)

Як видно з таблиці 3.1, впровадження механізмів безпеки очікувано збільшує час початкового встановлення з'єднання через необхідність обміну сертифікатами та генерації сесійних ключів. Проте цей показник є критичним лише під час ініціалізації сесії, тоді як рівень безпеки та стійкість до атак зростають до абсолютних значень.

3.5.2 Аналіз мережевого трафіку

Аналіз дампів мережевого трафіку, отриманих за допомогою аналізатора Wireshark, дозволив зробити наступні висновки:

Перевірка шифрування даних у базовій конфігурації корисне навантаження (Payload) протоколів MQTT та CoAP передавалося у форматі Plain Text, що дозволяло зловмиснику вільно читати телеметрію та керуючі команди. Після активації TLS (на порту 8883) та DTLS (на порту 5684) архітектура трафіку докорінно змінилася. Усі пакети прикладного рівня були інкапсульовані у фрейми Application Data, що унеможливило читання чи

модифікацію даних транзитними вузлами.

Виявлення аномалій це провадження авторизації та списків доступу (ACL) дозволило системі ефективно реагувати на мережеві аномалії. Будь-які спроби неавторизованої публікації в керуючі топіки або відправки невалідних команд миттєво фіксувалися брокером і відхилялися з генерацією пакетів Connection Refused: not authorised. Це ізолює зловмисника ще на етапі спроби доступу до ресурсів.

3.5.3 Оцінка впливу захисту на продуктивність

Будь-яка система криптографічного захисту створює додаткове навантаження на обчислювальні ресурси (Overhead). У рамках дослідження було проведено оцінку цього впливу на серверне обладнання (брокер Mosquitto):

Навантаження CPU це айбільш показовим є вплив захисту під час DoS-атак. Без застосування Rate Limiting масове створення напіввідкритих з'єднань призводило до різкого стрибка використання процесора (до 25-27%). Після конфігурації лімітів підключень та заборони анонімного доступу брокер почав превентивно відкидати шкідливі запити. У результаті споживання CPU під час аналогічної атаки знизилося до фонового рівня (0.05% - 0.24%). Витрати процесорного часу на симетричне шифрування легітимного трафіку виявилися статистично незначними для сучасних процесорів.

Використання пам’яті (RAM) це впровадження TLS/DTLS збільшило базове споживання оперативної пам'яті контейнером на 1-2 МБ (через необхідність зберігання контексту сесій та сертифікатів у пам'яті). Під час стрес-тестування споживання залишалося стабільним (на рівні 1.8 – 2.5 МБ), що доводить відсутність витоків пам'яті (Memory Leaks) та стійкість системи до Connection Flooding.

Затримка доставки повідомлень це хоча ініціалізація захищеного з'єднання вимагає додаткових витрат часу (Handshake), подальша передача зашифрованих пакетів відбувається із застосуванням швидких алгоритмів

симетричного шифрування (наприклад, AES-256-GCM). Додаткова затримка маршрутизації повідомлення становить не більше 2-3 мілісекунд, що є абсолютно прийнятним показником для переважної більшості систем розумного міста та промислового IoT, де не вимагається жорсткий реальний час (Hard Real-Time).

ВИСНОВКИ

У кваліфікаційній роботі вирішено актуальне науково-практичне завдання, яке полягало у дослідженні методів захисту від атак на рівні прикладних протоколів Інтернету речей (MQTT та CoAP). Поставлена мета була успішно досягнута шляхом теоретичного обґрунтування та практичної побудови комплексної ешелонованої системи кіберзахисту.За результатами виконання роботи можна зробити наступні висновки:

Проведено теоретичний аналіз архітектури IoT. Визначено, що протоколи прикладного рівня MQTT та CoAP, незважаючи на свою легковагомість та енергоефективність, у базових специфікаціях не мають вбудованих механізмів шифрування та суворої автентифікації, що робить їх критично вразливими до сучасних кіберзагроз.

Розроблено та розгорнуто віртуалізоване тестове середовище. За допомогою технології контейнеризації Docker створено ізольовану мережеву інфраструктуру, що включала MQTT-брокер Mosquitto, CoAP-сервер та імітовані вузли зловмисника, що дозволило безпечно проводити пентестинг без ризику для реальних мереж.

Практично доведено вразливість базових конфігурацій. Шляхом моделювання цілеспрямованих атак підтверджено критичний рівень загроз: успішно реалізовано атаку перехоплення трафіку (Sniffing), несанкціоноване дублювання керуючих команд (Replay Attack) та атаку типу «відмова в обслуговуванні» (DoS – Connection Flooding), яка призвела до вичерпання ресурсів центрального процесора вузла.

Впроваджено криптографічний захист транспортного рівня. За допомогою інфраструктури відкритих ключів (PKI) та утиліти OpenSSL згенеровано сертифікати X.509. Налаштовано шифрування трафіку з використанням протоколу TLS для MQTT та протоколу DTLS для CoAP-комунікацій. Практично доведено, що інкапсуляція корисного навантаження нівелює ризики перехоплення даних (Sniffing) та атак Man-in-the-Middle.

Налаштовано багаторівневу авторизацію та управління ресурсами. Для захисту від несанкціонованого доступу та експлуатації ресурсів впроваджено систему парольної автентифікації, розроблено політику списків контролю доступу (ACL) згідно з принципом найменших привілеїв, а також встановлено мережеві ліміти (Rate Limiting).

Доведено ефективність розробленої архітектури. Порівняльний аналіз та стрес-тестування підтвердили, що комплексна система захисту ефективно блокує спроби анонімного підключення та відбиває DoS-атаки на ранніх етапах. При цьому споживання ресурсів (CPU) під час атак не перевищує 0.25%, а додаткова затримка маршрутизації зашифрованих пакетів залишається в межах прийнятних значень для IoT-систем.

Отримані результати, розроблені конфігурації та скрипти мають високу практичну цінність. Запропонована архітектура безпеки може бути безпосередньо інтегрована при проектуванні сучасних систем промислової автоматизації, інфраструктури «розумного міста» та під час впровадження рішень цифрової трансформації на муніципальних підприємствах.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. MQTT Version 5.0. OASIS Standard. URL: https://docs.oasis-open.org/mqtt/mqtt/v5.0/mqtt-v5.0.html (дата звернення: 10.05.2026).
2. Shelby Z., Hartke K., Bormann C. The Constrained Application Protocol (CoAP). RFC 7252. Internet Engineering Task Force, 2014. URL: https://datatracker.ietf.org/doc/html/rfc7252 (дата звернення: 10.05.2026).
3. Rescorla E. The Transport Layer Security (TLS) Protocol Version 1.3. RFC 8446. Internet Engineering Task Force, 2018. URL: https://datatracker.ietf.org/doc/html/rfc8446 (дата звернення: 11.05.2026).
4. Rescorla E., Tschofenig H., Modadugu N. The Datagram Transport Layer Security (DTLS) Protocol Version 1.3. RFC 9147. Internet Engineering Task Force, 2022. URL: https://datatracker.ietf.org/doc/html/rfc9147 (дата звернення: 11.05.2026).
5. Гнатюк С. О. Кібербезпека в системах Інтернету речей : навч. посіб. Київ : КПІ ім. Ігоря Сікорського, 2023. 245 с. URL: https://ela.kpi.ua/items/62c2fec5-872d-4f63-ad4f-1115d03aaedc (дата звернення: 10.05.2026).
6. Коваленко О. В., Шевченко І. В. Порівняльний аналіз протоколів MQTT та CoAP для систем Інтернету речей. Вісник телекомунікацій. 2024. № 2. С. 45–52. URL: https://duikt.edu.ua/uploads/n_11268_35712423.pdf (дата звернення: 11.05.2026).
7. Мельник А. А. Аналіз вразливостей протоколу CoAP до атак типу DDoS Amplification. Кібербезпека: освіта, наука, техніка. 2023. Т. 4, № 3. С. 112–120. URL: https://www.csecurity.kubg.edu.ua/index.php/journal/article/view/864 (дата звернення: 11.05.2026).
8. Корченко О. Г. Основи архітектури та технологій Інтернету речей : підручник. Київ : НАУ, 2022. 310 с.
9. Bormann C., Ersue M., Keranen A. Terminology for Constrained-Node Networks.

RFC 7228. Internet Engineering Task Force, 2014. URL: https://datatracker.ietf.org/doc/html/rfc7228 (дата звернення: 12.05.2026).

10. Бурячок В. Л. Інформаційна та кібербезпека: соціотехнічний аспект : підручник. Київ : ЗАТ "ВІПОЛ", 2021. 420 с.
11. Docker Documentation : веб-сайт. URL: https://docs.docker.com/ (дата звернення: 14.05.2026).
12. Eclipse Mosquitto Documentation : веб-сайт. URL: https://mosquitto.org/documentation/ (дата звернення: 14.05.2026).
13. Wireshark User's Guide : веб-сайт. URL: https://www.wireshark.org/docs/wsug_html_chunked/ (дата звернення: 14.05.2026).
14. OpenSSL Cryptography and SSL/TLS Toolkit : веб-сайт. URL: https://www.openssl.org/docs/ (дата звернення: 15.05.2026).
15. Fail2ban intrusion prevention software : веб-сайт. URL: https://www.fail2ban.org/ (дата звернення: 15.05.2026).