Міністерство освіти і науки України
Київський національний університет імені Тараса Шевченка

Факультет інформаційних технологій
Кафедра кібербезпеки та захисту інформації

ДОПУСТИТИ ДО ЗАХИСТУ:
В.о. завідувача кафедри
кібербезпеки
та захисту інформації
_____ Іван ПАРХОМЕНКО
«_» _____ 2025 р.

ПОЯСНЮВАЛЬНА ЗАПИСКА

кваліфікаційної роботи

галузь знань *12 Інформаційні технології*
(шифр і назва галузі знань)

спеціальність *125 Кібербезпека та захист інформації*
(код і назва спеціальності)

освітній ступень *магістр*

освітньо-наукова програма *Кібербезпека*
(назва освітньої програми)

на тему: «Метод управління захистом інформації в IoT»

Виконавець: студент ІІ курсу, групи КБм-22

_____ **Владислав КОШЛА**
(підпис) (Ім'я, ПРІЗВИЩЕ)

	Ім'я, ПРІЗВИЩЕ	**Підпис**
Науковий керівник	Сергій ТОЛЮПА	
Нормоконтроль	Юрій БАБЕНКО	

Київ 2025

Міністерство освіти і науки України
Київський національний університет імені Тараса Шевченка

Факультет інформаційних технологій
Кафедра кібербезпеки та захисту інформації

ЗАТВЕРДЖЕНО:
В.о. завідувача кафедри
кібербезпеки
та захисту інформації
_____ Іван ПАРХОМЕНКО
«25» жовтня 2024 р.

ЗАВДАННЯ

на виконання кваліфікаційної роботи

спеціальності *125 Кібербезпека та захист інформації*
(код і назва спеціальності)

освітній ступень *магістр*

Здобувача(ки) КБм-22 (група) Кошли Владислава Юрійовича (прізвище ім'я по-батькові)

Тема дипломної роботи Методи управління захистом інформації в IoT

1. ПІДСТАВИ ДЛЯ ПРОВЕДЕННЯ РОБОТИ

Рішення засідання кафедри кібербезпеки та захисту інформації факультету інформаційних технологій протокол № 4 від 24.10.2024

2. МЕТА ТА ВИХІДНІ ДАНІ ДЛЯ ПРОВЕДЕННЯ РОБІТ

Об'єкт досліджень	Процес управління захистом інформації в системах Інтернету речей (IoT)
Предмет досліджень	Методи, моделі та технології управління захистом інформації в IoT-системах, зокрема методи аутентифікації, криптографії, моніторингу безпеки, поведінкового аналізу та Zero Trust Architecture.
Мета	Підвищення ефективності методів управління захистом інформації в середовищі IoT, з урахуванням сучасних загроз, обмежень пристроїв та специфіки їх функціонування, а також реалізувати практичні рішення з підвищення безпеки.

Вихідні дані для проведення роботи	Аналітичний огляд загроз та вразливостей IoT-систем, існуючі нормативно-правові акти (GDPR, NIS2, Cybersecurity Act), публікації та дослідження в галузі кібербезпеки IoT, відомі випадки атак на IoT, технічна база (IoT-пристрій ESP8266)

3. ОЧІКУВАНІ НАУКОВІ РЕЗУЛЬТАТИ

Наукова новизна	У роботі запропоновано комплексний підхід до управління захистом IoT-систем, який поєднує архітектуру нульової довіри, поведінковий контроль доступу та криптографічне забезпечення журналів подій. Проведено експериментальну оцінку ефективності запропонованих рішень на реальному пристрої.
Практична цінність	Результати роботи можуть бути використані для створення безпечних IoT-рішень у критичних галузях і адаптовані до ресурсних обмежень пристроїв для підвищення загального рівня безпеки.

4. ЕТАПИ ВИКОНАННЯ РОБОТИ

Найменування етапів робіт	Строки виконання робіт (початок-кінець)
Уточнення постановки задачі	25.10.2024 – 01.11.2024
Аналіз літературних джерел	02.11.2024 – 15.11.2024
Формулювання мети, об’єкта та предмета дослідження	16.11.2024 – 22.11.2024
Аналіз літературних джерел та нормативно-правової бази у сфері IoT	23.11.2024 – 10.12.2024
Ідентифікація основних загроз, вразливостей та класифікація атак на IoT-пристрої	11.12.2024 – 15.01.2025
Розробка методів управління захистом (Zero Trust, криптографія, аудит тощо), з урахуванням обмежень IoT-середовища	16.01.2025 – 10.02.2025
Реалізація елементів Zero Trust Architecture на ESP8266	11.02.2025 – 01.03.2025
Реалізація поведінкової логіки авторизації та фільтрації запитів IoT-користувачів	02.03.2025 – 15.03.2025
Реалізація журналу подій із криптографічним ланцюгуванням записів (audit trail)	16.03.2025 – 10.04.2025
Проведення тестування IoT-пристрою	11.04.2025 – 15.04.2025

Аналіз результатів та оцінка ефективності запропонованих рішень	16.04.2025 – 01.05.2025
Оформлення пояснювальної записки згідно методичних рекомендацій	02.05.2025 – 15.05.2025
Подача пакету документів на розгляд ЕК	15.05.2025 – 19.05.2025

Завдання видав _____ Сергій ТОЛЮПА
(підпис) (Ім'я, ПРІЗВИЩЕ)

Завдання прийняв
до виконання _____ Владислав КОШЛА
(підпис) (Ім'я, ПРІЗВИЩЕ)

Дата видачі завдання: 25.10.2024 р.
Термін подання дипломної роботи до ЕК 19.05.2025 р.

РЕФЕРАТ

Пояснювальна записка до кваліфікаційної магістерської роботи «Методи управління захистом інформації в IoT» складається зі вступу, трьох розділів основної частини, висновків, списку використаних джерел та додатків. Загальний обсяг роботи – 99 сторінок. Робота містить 34 рисунка, 4 таблиць. Список використаних джерел включає 48 найменувань.

Об’єкт дослідження – процес управління захистом інформації в системах Інтернету речей (IoT).

Предмет дослідження – методи, моделі та технології управління захистом інформації в IoT-системах, зокрема методи аутентифікації, криптографії, моніторингу безпеки, поведінкового аналізу та Zero Trust Architecture.

Мета роботи – підвищення методів ефективності управління захистом інформації в середовищі IoT, з урахуванням сучасних загроз, обмежень пристроїв та специфіки їх функціонування, а також реалізувати практичні рішення з підвищення безпеки.

Актуальність роботи: полягає в зростанні кількості IoT-пристроїв, які активно впроваджуються в критично важливі сфери, але залишаються вразливими до кібератак через обмежені ресурси та слабкий вбудований захист, що потребує ефективного управління інформаційною безпекою.

Практична цінність: Результати роботи можуть бути використані для створення безпечних IoT-рішень у критичних галузях і адаптовані до ресурсних обмежень пристроїв для підвищення загального рівня безпеки.

Наукова новизна: У роботі запропоновано комплексний підхід до захисту IoT із Zero Trust, поведінковим контролем доступу та криптографічним журналюванням подій, ефективність якого експериментально підтверджено на реальному пристрої.

Ключові слова: Інтернет речей, кібербезпека IoT, управління захистом, Zero Trust Architecture, безпечний аудит, поведінковий контроль, ESP8266, криптографія.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ ТА СКОРОЧЕНЬ

IoT	–	Internet of Things
ІБ	–	Інформаційна безпека
ZTA	–	Zero Trust Architecture
IDS/IPS	–	Intrusion Detection/Prevention Systems
ESP8266	–	Microcontroller for IoT with Wi-Fi support
TLS	–	Transport Layer Security
MQTT	–	Message Queuing Telemetry Transport
OWASP	–	Open Web Application Security Project
API	–	Application Programming Interface
GDPR	–	General Data Protection Regulation
NIS2	–	Network and Information Security Directive 2
ENISA	–	European Union Agency for Cybersecurity
VPN	–	Virtual Private Network
DDoS	–	Distributed Denial of Service
AI	–	Artificial Intelligence
ML	–	Machine Learning
Audit Trail	–	Cryptographically Protected Event Log
AES	–	Advanced Encryption Standard
CVE	–	Common Vulnerabilities and Exposures

ЗМІСТ

ВСТУП

У сучасних умовах стрімкого розвитку інформаційних технологій Інтернет речей (Internet of Things, IoT) стає невід'ємною складовою цифрової трансформації суспільства. IoT охоплює велику кількість пристроїв, що забезпечують взаємодію фізичного світу з інформаційними системами шляхом обміну даними через стандартні мережеві протоколи. За прогнозами компанії Cisco, до 2030 року кількість пристроїв, підключених до Інтернету, перевищить 50 мільярдів одиниць [1]. Вони охоплюють всі сфери людської діяльності — від побутових систем автоматизації до високотехнологічних об'єктів енергетичної інфраструктури.

Використання пристроїв Інтернету речей дозволяє значно підвищити ефективність бізнес-процесів, оптимізувати витрати ресурсів, розширити можливості дистанційного моніторингу і управління системами [2]. Проте, водночас, широке розповсюдження IoT створює нові виклики у сфері інформаційної безпеки. Пристрої, які постійно підключені до глобальних мереж, стають потенційними мішенями для зловмисників. Компрометація IoT-пристроїв може призвести не лише до витоку конфіденційних даних, а й до порушення роботи критичних інфраструктур, створення загроз життю та здоров'ю людей [3].Особливої важливості питання кібербезпеки набуває у зв'язку з використанням IoT у сферах із підвищеним ризиком — охорона здоров'я, енергетика, транспорт, промисловість. Неконтрольований доступ до медичних пристроїв або систем керування енергетичними установками може мати катастрофічні наслідки. Саме тому питання забезпечення цілісності, конфіденційності та доступності даних у середовищі IoT вимагають негайного вирішення.

Однією з особливостей IoT-пристроїв є їх обмеженість в обчислювальних ресурсах, що ускладнює імплементацію класичних механізмів захисту. Це вимагає розробки спеціалізованих підходів до забезпечення безпеки, з урахуванням специфіки архітектури IoT-систем.

У межах даної магістерської роботи досліджуються актуальні загрози інформаційній безпеці об'єктів Інтернету речей, аналізуються слабкі місця типових пристроїв та визначаються найбільш ефективні методи захисту. Крім того, здійснюється практична реалізація механізмів управління захистом на прикладі реального IoT-пристрою із впровадженням сучасних концепцій, таких як поведінковий контроль доступу, криптографічний захист журналу подій та реалізація принципів архітектури нульової довіри (Zero Trust Architecture).

Актуальність обраної теми обумовлена зростаючою кількістю IoT-пристроїв у повсякденному житті та бізнесі, а також збільшенням кількості кіберінцидентів, що прямо пов'язані із недостатнім рівнем захищеності об'єктів Інтернету речей. Своєчасне впровадження ефективних заходів захисту має критичне значення для запобігання потенційним збиткам, захисту приватної інформації, збереження працездатності критичних систем і загальної безпеки цифрового середовища.

Таким чином, розробка та дослідження методів управління захистом інформації в IoT є надзвичайно важливим завданням сучасної кібербезпеки та має високу практичну цінність у контексті побудови стійких до атак інформаційних систем майбутнього.

РОЗДІЛ 1
ЗАГРОЗИ ТА МЕТОДИ УПРАВЛІННЯ ЗАХИСТОМ ІНФОРМАЦІЇ В IOT

1.1 Концепція інтернету речей (iot)

Інтернет речей (Internet of Things, IoT) — це концепція, яка передбачає об'єднання фізичних пристроїв у єдину мережу для автоматизованого збору, передачі та обробки даних без безпосереднього втручання людини. IoT поєднує реальний та цифровий світ, дозволяючи пристроям працювати в автономному режимі, що відкриває широкі можливості для автоматизації та ефективного управління інформацією. Основними принципами IoT є підключеність, автономність, масштабованість та безпека. Підключеність означає, що всі пристрої можуть взаємодіяти між собою та з мережею, автономність передбачає самостійне виконання пристроями певних дій, масштабованість дозволяє безперешкодне розширення системи, а безпека є критичним фактором у зв’язку з великою кількістю підключених елементів [1].

На рис. 1.1 зображено комплексне представлення IoT та IIoT . Архітектуру IoT можна класифікувати на сім рівнів: сприйняття, транспортування, межа, обробка, застосування, бізнес і рівень безпеки. Система працює як замкнутий цикл, полегшуючи виробництво індивідуальних товарів, адаптованих відповідно до конкретних вимог кожного кінцевого споживача. Швидке зростання IoT вимагає впровадження надійних протоколів безпеки та конфіденційності для пом’якшення потенційної вразливості системи та загроз. Крім того, у сфері IIoT такі фактори, як надійність, масштабованість і енергоспоживання, стають вирішальними міркуваннями. У нинішніх умовах звичайні заходи безпеки не завжди підходять.

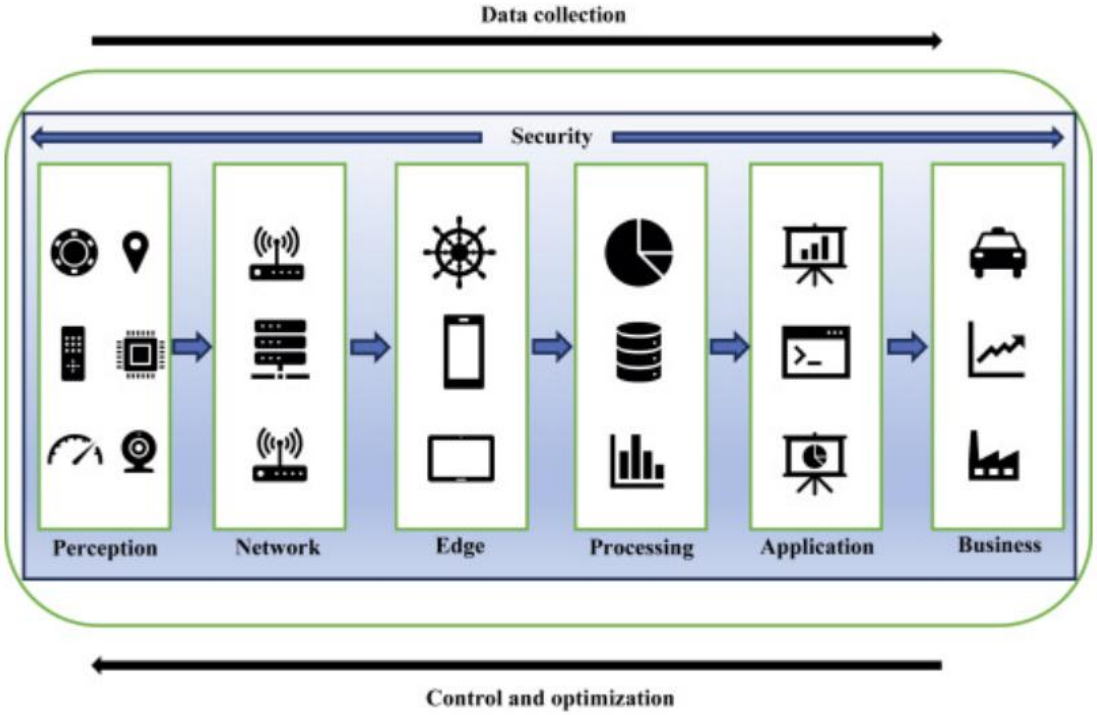

Рис. 1.1 Рівні архітектури IoT

Архітектура IoT складається з чотирьох основних рівнів. Перший рівень – пристрійний (Edge Layer**)** – включає фізичні сенсори, виконавчі механізми та мікроконтролери, які отримують і передають дані. Другий рівень – комунікаційний (Network Layer) – відповідає за з'єднання пристроїв між собою через різні мережеві технології, такі як Wi-Fi, Bluetooth, ZigBee, LoRaWAN або 5G, а також використовує протоколи передачі даних MQTT, CoAP, HTTP та AMQP. Третій рівень – обчислювальний (Processing Layer**)** – включає хмарні платформи (AWS IoT, Google Cloud IoT, Azure IoT) та інструменти для обробки даних, такі як Big Data, AI та машинне навчання. Четвертий рівень – прикладний (Application Layer) – містить інтерфейси для користувачів (мобільні додатки, веб-платформи) та інтелектуальні системи для автоматизованого керування й аналітики.

Сьогодні IoT активно використовується у різних сферах. У побутовій автоматизації та розумних будинках технології IoT забезпечують управління освітленням, клімат-контролем, безпекою та побутовими пристроями, такими як розумні термостати Nest або системи відеоспостереження Ring. **У** промисловому секторі (Industrial IoT, IIoT) IoT дозволяє здійснювати моніторинг обладнання,

контролювати логістичні процеси та оптимізувати виробництво. В охороні здоров’я IoT застосовується для віддаленого моніторингу пацієнтів, носимих пристроїв (розумні годинники, кардіомонітори) та роботизованих хірургічних систем. У сфері розумних міст (Smart Cities) технології IoT допомагають в управлінні дорожнім рухом, моніторингу навколишнього середовища, контролі за споживанням електроенергії та води. У транспорті та логістиці IoT забезпечує GPS-відстеження вантажів, автоматизацію процесів доставки та управління автономними транспортними засобами.

Попри численні переваги, IoT має низку викликів. Безпека є одним із головних ризиків, адже велика кількість підключених пристроїв збільшує вірогідність атак, таких як DDoS, компрометація даних чи викрадення конфіденційної інформації. Конфіденційність також є серйозною проблемою, оскільки пристрої IoT можуть збирати та передавати чутливі дані користувачів. Сумісність різних платформ залишається викликом через відсутність загальноприйнятих стандартів. Енергоефективність є ще одним аспектом, оскільки автономні пристрої потребують ефективного управління живленням, особливо в умовах роботи від батарей.

З точки зору подальшого розвитку IoT, можна виокремити кілька ключових тенденцій. По-перше, впровадження 5G та LPWAN дозволить підключати ще більше пристроїв до мережі, забезпечуючи швидку та стабільну передачу даних. По-друге, штучний інтелект і машинне навчання сприятимуть покращенню аналітики та автономного керування IoT-системами. По-третє, блокчейн може бути використаний для створення більш безпечних та децентралізованих IoT-екосистем, дозволяючи підвищити захищеність даних та ідентифікацію пристроїв. Нарешті, Zero Trust Security стає новою моделлю забезпечення кібербезпеки IoT, що передбачає постійну перевірку кожного пристрою та мінімізацію довіри в межах мережі [2].

1.2 Основні загрози та вразливості iot

Інтернет речей (IoT) активно впроваджується в різні сфери життя, проте з цим зростає кількість загроз, які можуть порушити безпеку користувачів та організацій.

Багато IoT-пристроїв мають обмежені обчислювальні ресурси, що не дозволяє інтегрувати надійні механізми захисту. Це робить їх вразливими до атак, які можуть призвести до витоку конфіденційної інформації, несанкціонованого втручання у функціонування систем, компрометації даних або навіть використання IoT-інфраструктури для здійснення глобальних кібератак.

Загрози для IoT можна розділити на кілька категорій: апаратні вразливості, загрози на рівні комунікацій, загрози для програмного забезпечення та соціальна інженерія [3].

Загрози, пов’язані з апаратним забезпеченням

Апаратні загрози є одними з найбільш небезпечних, оскільки фізичний доступ до пристрою дозволяє отримати контроль над його функціями або змінити конфігурацію. Наприклад, багато IoT-пристроїв, таких як камери відеоспостереження чи датчики в розумних містах, розміщені у відкритому середовищі, що полегшує зловмисникам можливість отримання фізичного доступу. Окрім фізичного маніпулювання, зловмисники можуть використовувати різні методи атак. Одним із критичних ризиків є підміна мікропрограмного забезпечення (firmware attacks), коли атакуючі змінюють прошивку пристрою, встановлюючи бекдори або інше шкідливе ПЗ, яке може дозволити їм контролювати пристрій дистанційно.

Реальним прикладом таких атак є злам автомобільних систем Tesla, коли дослідники безпеки продемонстрували можливість переписування прошивки, що дозволяло отримати повний контроль над функціонуванням транспортного засобу. Ще одним прикладом є атака на медичні імпланти, такі як кардіостимулятори, де змінене мікропрограмне забезпечення могло створювати ризики для здоров'я пацієнтів.

Додатково, велика кількість IoT-пристроїв не мають вбудованих механізмів шифрування збережених даних, що дає змогу зловмисникам отримати доступ до конфіденційної інформації, якщо пристрій буде скомпрометований. Наприклад, дослідники виявили, що багато "розумних" дверних замків не використовують шифрування для збереження кодів доступу, що дозволяло зловмисникам зчитувати їх за допомогою простих технічних засобів. Такі загрози підкреслюють важливість

посилення захисту IoT-пристроїв на рівні апаратного забезпечення та програмного коду.

Загрози на рівні комунікацій

Оскільки IoT-пристрої постійно взаємодіють між собою та із зовнішніми серверами, загрози на рівні зв’язку є одними з найбільш розповсюджених. Найбільш небезпечними є атаки типу "Man-in-the-Middle (MitM)", коли зловмисник перехоплює трафік між пристроєм та сервером, змінюючи команди або отримуючи доступ до конфіденційної інформації. Такі атаки можуть використовуватися для перехоплення логінів, паролів, токенів аутентифікації або навіть підміни команд управління пристроєм. Наприклад, вразливість у системах розумних камер відеоспостереження дозволила хакерам отримувати відеопотік у реальному часі та змінювати налаштування пристроїв [4].

Значною загрозою є DDoS-атаки, які можуть використовувати зламані IoT-пристрої для генерації масового трафіку з метою перевантаження серверів. Прикладом є ботнет Mirai, який у 2016 році інфікував тисячі IoT-пристроїв, зокрема IP-камери та маршрутизатори, використовуючи стандартні паролі. Атака ботнету спричинила серйозні перебої в роботі великих онлайн-платформ, таких як Twitter, Netflix і Reddit. Подібний випадок стався у 2018 році з ботнетом VPNFilter, який уразив понад 500 тисяч маршрутизаторів і мережевих пристроїв, дозволяючи атакувальникам перехоплювати трафік, відключати пристрої або навіть змінювати дані, що передаються мережею.

Окрім цього, загрозою є використання небезпечних або застарілих протоколів зв’язку, таких як Telnet або HTTP, які передають дані у відкритому вигляді. Це дозволяє зловмисникам легко отримувати логіни, паролі та інші важливі дані користувачів. Реальним прикладом такої атаки є випадок із виробником IoT-пристроїв TRENDnet, коли через уразливість у веб-інтерфейсі тисячі камер спостереження стали доступними в мережі без аутентифікації. Це демонструє важливість використання сучасних криптографічних протоколів, таких як TLS 1.3, та регулярного оновлення прошивки пристроїв.

Загрози для програмного забезпечення IoT

Окрім апаратних та мережевих загроз, програмне забезпечення IoT також має низку вразливостей. Наприклад, багато пристроїв використовують слабку автентифікацію, що дозволяє зловмисникам отримати доступ до них за допомогою стандартних або легко вгадуваних паролів. Наприклад, ботнет Mirai, який здійснив масштабну атаку у 2016 році, використовував список типових паролів для проникнення в тисячі IoT-пристроїв, таких як камери спостереження та маршрутизатори.

Доволі поширеною проблемою є відсутність механізму автоматичного оновлення. Це призводить до того, що пристрої залишаються вразливими до експлойтів та відомих атак, оскільки користувачі часто не оновлюють їх вручну. Реальним прикладом є атака на інтелектуальні дверні замки, коли через застаріле програмне забезпечення зловмисники змогли отримати віддалений контроль над пристроями. Додатково, деякі IoT-пристрої використовують небезпечні API, що дозволяють отримати несанкціонований доступ до критично важливих даних. Наприклад, у 2021 році дослідники кібербезпеки виявили уразливість у API розумних термостатів, що дозволяла змінювати налаштування температури в оселях без відома власників [5].

Поширеним методом атак є буферне переповнення (Buffer Overflow), коли атакуючі вводять спеціальні команди, що викликають помилки в роботі пристрою та дозволяють виконувати довільний код. Така уразливість була виявлена у мережевих камерах безпеки, де зловмисники могли дистанційно змінювати параметри відеозапису, або навіть вимикати камери для проведення несанкціонованих дій. Таким чином, слабкий захист програмного забезпечення IoT-пристроїв створює значні ризики для безпеки як окремих користувачів, так і цілих організацій.

Соціальна інженерія та людський фактор

Навіть якщо система має високий рівень технічного захисту, людський фактор залишається слабкою ланкою безпеки IoT. Одним із найпоширеніших методів атак є фішинг, коли зловмисники надсилають підроблені електронні листи або повідомлення, змушуючи користувачів розголошувати конфіденційну інформацію. Така атака може мати різні форми, зокрема через електронну пошту, SMS або

соціальні мережі, де зловмисники маскуються під офіційні організації, щоб виманити особисті дані.

Ще однією критичною проблемою є використання слабких або стандартних паролів. Багато IoT-пристроїв постачаються з дефолтними паролями, які користувачі не змінюють, що дозволяє зловмисникам легко отримати доступ до мережі. Наприклад, ботнет Mirai, який атакував тисячі IoT-пристроїв, використовував прості паролі для проникнення в камери відеоспостереження та маршрутизатори. Також відомий випадок атаки на інтелектуальні дверні замки, коли через слабке шифрування зловмисники змогли віддалено відкривати двері без відома власників.

Окрім паролів, зловмисники можуть використовувати підроблене програмне забезпечення (malware), яке видає себе за офіційні оновлення або корисні додатки. Наприклад, у 2020 році хакери створили підроблений мобільний додаток для керування розумними лампами, який фактично викрадав дані користувачів. Таким чином, соціальна інженерія залишається однією з головних загроз для безпеки IoT, оскільки спрямована на людську довірливість та відсутність кібергігієни.

Ще одним небезпечним вектором атак є використання підробленого ПЗ. Деякі користувачі можуть завантажувати шкідливі додатки, які маскуються під офіційні IoT-додатки, але насправді виконують зловмисні дії, такі як передача паролів або віддалене управління пристроями.

Приклади реальних атак на IoT

Окрім ботнету Mirai, який спричинив масштабні перебої в роботі Інтернету, існує низка інших відомих атак на IoT. Наприклад, у 2018 році був виявлений шкідливий код VPNFilter, який заражав маршрутизатори та дозволяв зловмисникам отримувати контроль над мережами. Атака призвела до зараження понад 500 тисяч пристроїв у 54 країнах світу, дозволяючи хакерам віддалено керувати маршрутизаторами, перехоплювати трафік і навіть повністю виводити пристрої з ладу.

Ще одним небезпечним інцидентом стала атака на систему водопостачання у Флориді у 2021 році, коли зловмисники змогли змінити параметри очищення води через уразливість IoT-пристроїв. Використовуючи віддалений доступ, хакери

підвищили рівень гідроксиду натрію у водопостачанні міста Олдсмар, що могло призвести до отруєння населення. Вчасне виявлення атаки оператором дозволило запобігти катастрофі.

Іншим значним випадком стала атака на пристрої Philips Hue у 2020 році. Дослідники кібербезпеки виявили уразливість у смарт-лампах, яка дозволяла зловмисникам отримати контроль над пристроями через мережевий протокол Zigbee. Використовуючи заражену лампу, хакери могли проникнути в домашню мережу користувача, розповсюджуючи шкідливе програмне забезпечення на інші пристрої.

Також у 2019 році кіберзлочинці використали вразливість у розумних холодильниках Samsung, що дозволяло їм перехоплювати облікові дані користувачів Google через небезпечне з'єднання пристрою з мережею. Це підкреслює, що навіть побутові IoT-пристрої можуть становити значну загрозу, якщо їх безпека не приділяється належної уваги.

Вразливості IoT

Умовно можемо розділити вразливості IoT на 9 категорій. Крім того, у таблиці показано відображення між атаками IoT та вразливими місцями IoT (див. додаток Б).

1. Недостатній рівень фізичної безпеки. Більшість IoT-пристроїв функціонують у неконтрольованих середовищах, що робить їх вразливими до фізичного доступу зловмисників. Вони можуть взяти пристрій під контроль, завдавши йому механічних пошкоджень, викривши криптографічні алгоритми, скопіювавши прошивку або зламавши контрольні та кібердані.

2. Обмежене енергозабезпечення. Пристрої IoT часто мають обмежені джерела живлення та не завжди підтримують функцію самозарядки. Це відкриває можливість для атак, коли зловмисник шляхом надсилання великої кількості запитів витрачає енергію пристрою, роблячи його недоступним для законного використання.

3. Недосконала автентифікація. Обмежені ресурси IoT-пристроїв ускладнюють реалізацію надійних механізмів автентифікації. Це дає зловмисникам можливість створювати фальшиві вузли в мережі, змінювати або підробляти дані, а також використовувати слабкі механізми захисту для отримання несанкціонованого

доступу. Якщо автентифікаційні ключі не зберігаються або не передаються належним чином, це може призвести до їх втрати, компрометації або знищення.

4. Ненадійне шифрування. У критичних кіберфізичних системах, таких як електромережі, промислові підприємства та розумні будинки, захист даних є надзвичайно важливим. Шифрування забезпечує збереження та конфіденційність інформації, проте обмежені ресурси IoT-пристроїв можуть негативно впливати на ефективність криптографічних алгоритмів. Це створює можливості для обходу шифрування та отримання доступу до конфіденційних даних або маніпулювання пристроями.

5. Наявність зайвих відкритих портів. Багато IoT-пристроїв містять відкриті порти, які не є критично необхідними для їх роботи. Це створює потенційні вразливості, які можуть використовуватися зловмисниками для встановлення несанкціонованого зв’язку з пристроєм та проведення атак.

6. Слабкий контроль доступу. Для безпеки IoT-пристроїв необхідне належне управління обліковими даними. Однак багато пристроїв постачаються з типовими паролями, які користувачі не змінюють після встановлення. Крім того, значна частина IoT-платформ має високі рівні доступу за замовчуванням, що дозволяє зловмисникам отримувати контроль над пристроями та загрожувати безпеці даних.

7. Недостатнє управління оновленнями та виправленнями. Регулярне оновлення прошивки та операційної системи IoT-пристроїв критично важливе для усунення вразливостей. Проте багато виробників не забезпечують автоматичні оновлення або ж вони не мають механізмів перевірки цілісності, що створює ризики впровадження шкідливих змін у систему.

8. Небезпечні практики програмування. Дослідження показують, що багато оновлень мікропрограм містять відомі вразливості, такі як бекдори, облікові записи root за замовчуванням або відсутність SSL-захисту. Це дає зловмисникам можливість здійснювати атаки, змінювати дані або отримувати несанкціонований доступ до пристроїв.

9. Відсутність ефективних механізмів аудиту. Багато IoT-систем не мають розвинених протоколів реєстрації подій, що ускладнює виявлення зловмисної

активності. Це дозволяє атакам залишатися непоміченими, що значно ускладнює аналіз інцидентів і реагування на них.

Вразливості безпеки в IoT широко досліджуються, зокрема OWASP виділяє 10 основних загроз (див. додаток В). Крім того у таблиці (див. додаток В), наведена класифікація вразливостей IoT, яка охоплює кілька ключових аспектів.

Тапер перейдемо трохи до статистики. На рис. 1.2 графік демонструє щорічну кількість атак зловмисного програмного забезпечення (Malware) на пристрої Інтернету речей (IoT) у період з 2018 по 2022 рік. Дані зібрані компанією SonicWall Capture Labs та опубліковані на платформі Statista у березні 2023 року.

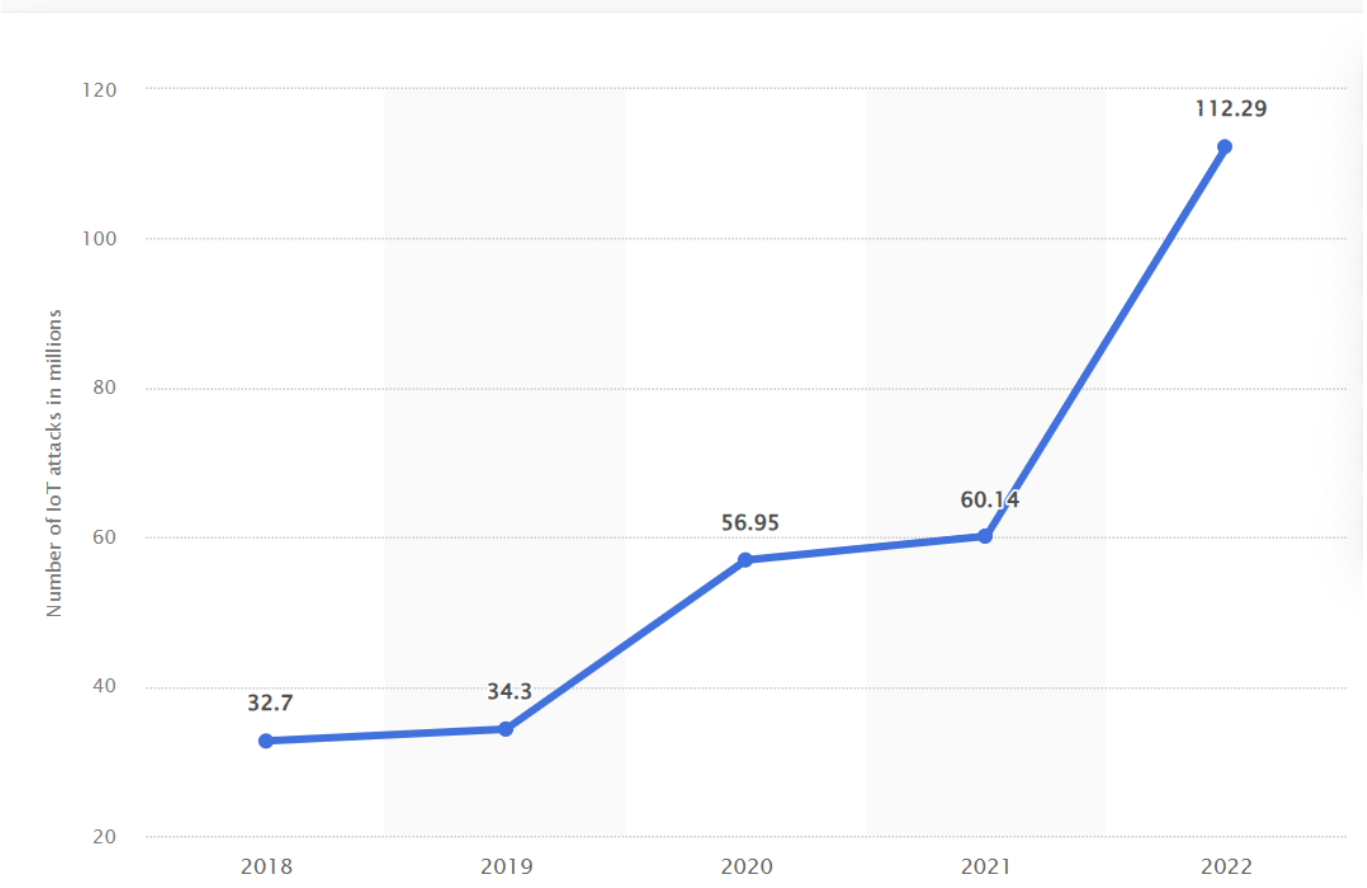

Рис. 1.2 Кількість атак на IoT у період 2018 по 2022 рік (дані у мільйонах)

Згідно зі статистикою, у 2018 році було зафіксовано 32,7 мільйона атак, що стало відносно невисоким показником, оскільки IoT-пристрої ще не набули масового поширення, а питання їх безпеки лише починало привертати увагу дослідників. У 2019 році кількість атак зросла до 34,3 мільйона, що становило приріст у 4,9%.

Незначне зростання, ймовірно, було пов’язане зі збільшенням кількості IoT-пристроїв та появою перших масштабних ботнет-атак, таких як Mirai.

У 2020 році відбувся різкий стрибок кількості атак – до 56,95 мільйона (+66% порівняно з попереднім роком). Це могло бути спричинене масовою цифровізацією через пандемію COVID-19, значним зростанням використання IoT у домашніх і корпоративних мережах, а також активним використанням ботнетів Mozi та Dark Nexus, які експлуатували вразливості пристроїв. У 2021 році кількість атак досягла 60,14 мільйона (+5,6%), що вказувало на деяке уповільнення темпів зростання. Це, можливо, пояснюється тим, що багато компаній почали впроваджувати заходи кібербезпеки, що тимчасово зменшило ефективність атак.

Найбільш вражаючим став 2022 рік, коли кількість атак вибухово зросла до 112,29 мільйона (+86,7%). Основними причинами такого зростання стали розширення атак ботнетів (Mirai, Mozi, Hajime), використання вразливостей у VPNFilter та RATs (Remote Access Trojans), а також недостатня безпека IoT-пристроїв у промисловості та критично важливій інфраструктурі.

Головними факторами, що спричинили загальне зростання атак, є масове поширення IoT-пристроїв, які часто мають слабкі налаштування безпеки (відкриті порти, дефолтні паролі), активне використання ботнетів для атак на мережеві пристрої, низький рівень безпеки багатьох IoT-пристроїв через відсутність оновлень прошивки та надійних методів автентифікації, а також глобальні кіберконфлікти, у яких зловмисники використовують IoT-пристрої для атак на критичну інфраструктуру та шпигунства.

Очікується, що тенденція зростання атак на IoT-пристрої продовжуватиметься у 2023–2025 роках, оскільки IoT-екосистема стрімко розширюється, а рівень безпеки багатьох пристроїв залишається низьким. Для боротьби з такими загрозами необхідні нові методи захисту, зокрема впровадження Zero Trust Architecture (ZTA), використання штучного інтелекту для моніторингу безпеки IoT та посилення законодавчого регулювання у сфері безпеки Інтернету речей. Ця інформація може бути корисною для вашого дослідження у магістерській роботі, особливо у розділах, присвячених аналізу атак та рекомендаціям щодо покращення безпеки IoT.

Згідно рис.1.3 щомісячна кількість атак зловмисного програмного забезпечення на пристрої Інтернету речей (IoT) у період з 2020 по 2022 рік демонструє чітку тенденцію до зростання, що свідчить про збільшення загроз у цій сфері. У 2020 році загальний рівень атак залишався відносно стабільним, починаючи з 3,74 мільйона у січні та незначними коливаннями протягом року. Найменша кількість атак була зафіксована у квітні (2,07 млн), а найбільша у жовтні (10,83 млн).

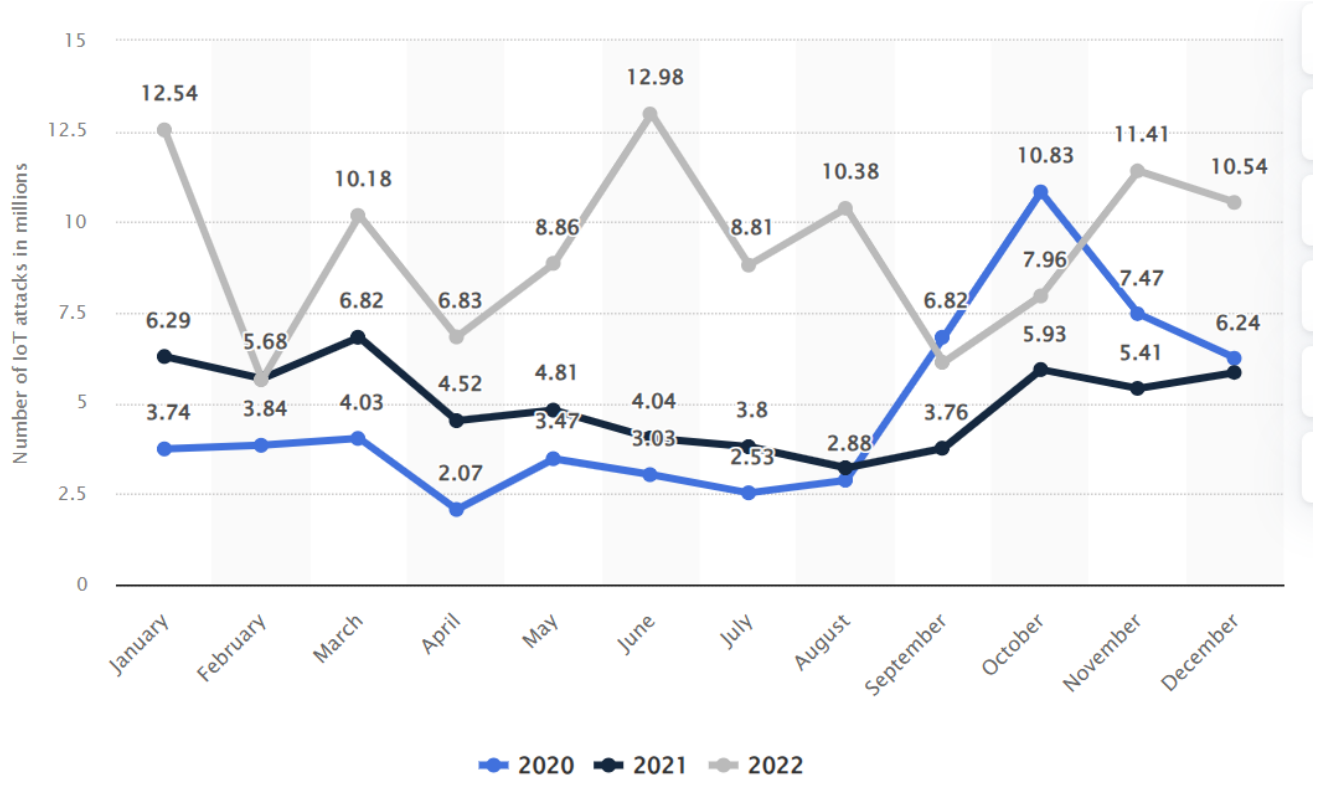

Рис. 1.3 Щомісячна кількість атак на IoT (в мільйонах)

У 2021 році ситуація почала змінюватися, і середня кількість атак поступово зростала. Січень розпочався з рівня 6,29 млн атак, що було вже значно більше, ніж у попередньому році. Показники залишалися відносно стабільними до серпня, після чого почалося поступове збільшення активності зловмисників. У жовтні 2021 року спостерігався пік атак, що досяг 5,93 млн, після чого рівень загроз залишався високим до кінця року.

2022 рік став періодом значного зростання атак на IoT-пристрої. Уже в січні було зафіксовано рекордну кількість атак – 12,54 млн, що майже вдвічі перевищувало показники січня 2021 року. Протягом року фіксувалися різкі стрибки активності

зловмисного програмного забезпечення, зокрема у червні (12,98 млн), жовтні (10,83 млн) та листопаді (11,41 млн). Це свідчить про те, що кіберзлочинці все більше націлюються на пристрої IoT, використовуючи їхні вразливості для масштабних атак.

Загалом, аналіз цих даних вказує на посилення загроз у сфері IoT та необхідність впровадження більш ефективних механізмів захисту. Це може бути зумовлено як збільшенням кількості IoT-пристроїв у повсякденному житті, так і вдосконаленням методів атак з боку зловмисників. Важливим аспектом стає використання комплексних стратегій кібербезпеки, що включають оновлення програмного забезпечення, застосування сучасних методів аутентифікації та впровадження штучного інтелекту для аналізу загроз і виявлення атак у реальному часі.

Методи мінімізації загроз в IoT

Для забезпечення безпеки IoT необхідно впроваджувати комплексні заходи захисту. Перш за все, важливо використовувати надійну автентифікацію, включаючи двофакторну перевірку та унікальні паролі для кожного пристрою. Регулярне оновлення програмного забезпечення дозволяє закривати відомі вразливості та запобігати експлойтам.

Шифрування даних є ключовим елементом захисту IoT. Використання сучасних криптографічних алгоритмів, таких як AES-256 та TLS 1.3, дозволяє захистити інформацію від несанкціонованого доступу. Окрім цього, важливим заходом є мережевий захист, який включає використання міжмережевих екранів (Firewall) та систем виявлення вторгнень (IDS/IPS), що дозволяють виявляти та блокувати підозрілу активність у мережі.

Додатково, слід впроваджувати сегментування мережі, розділяючи IoT-пристрої на окремі сегменти для запобігання поширенню атак. Використання штучного інтелекту та машинного навчання для аналізу аномалій у поведінці пристроїв також може значно підвищити рівень безпеки IoT.

Загалом, хоча Інтернет речей відкриває нові можливості для автоматизації та ефективного управління, безпека залишається серйозним викликом. Без належного

захисту IoT-системи можуть стати легкою мішенню для зловмисників, що робить питання кібербезпеки критично важливим для подальшого розвитку цієї технології.

1.3 Огляд нормативно-правових аспектів захисту iot

З огляду на швидке розширення Інтернету речей (IoT) у всіх сферах життєдіяльності, зокрема в критично важливих секторах, таких як медицина, промисловість, транспорт і фінансові установи, постає питання безпеки та захисту інформації. Використання IoT передбачає збір, зберігання та обробку великого обсягу даних, що створює потенційні загрози несанкціонованого доступу, витоку інформації та кібератак. Тому нормативно-правове регулювання у цій сфері є вкрай важливим для забезпечення надійного функціонування IoT-систем [3].

У зв'язку з глобальним розвитком технологій Інтернету речей (IoT) та збільшенням кількості підключених пристроїв виникає потреба у створенні відповідного правового регулювання, яке б забезпечувало належний рівень безпеки та конфіденційності даних користувачів. Різні міжнародні організації та держави розробили ряд нормативно-правових актів, які визначають вимоги до захисту інформації та кібербезпеки IoT-систем.

Одним із ключових документів у цій сфері є Загальний регламент захисту даних (GDPR), прийнятий Європейським Союзом. Він встановлює чіткі правила щодо обробки персональних даних, що стосуються також IoT-пристроїв. GDPR передбачає використання таких механізмів, як шифрування, псевдонімізація, обмеження доступу та вимога отримання згоди користувача на збір його даних. Важливим аспектом є також право користувачів на видалення своїх даних («право на забуття»). Випадки порушень GDPR призводять до значних штрафів, як це сталося у 2021 році, коли компанія Amazon була оштрафована на 746 мільйонів євро за недотримання вимог щодо збору персональних даних.

Ще одним важливим документом є Директива NIS (Network and Information Security Directive), яка зобов’язує операторів критичної інфраструктури

впроваджувати заходи безпеки для мінімізації кібератак. У 2023 році була ухвалена оновлена версія цієї директиви – NIS2, що розширює вимоги до кібербезпеки для постачальників цифрових послуг, державних установ та великих компаній, які працюють у сфері IoT. Вона передбачає обов'язкове впровадження ризик-орієнтованих заходів, моніторинг загроз та швидке реагування на інциденти безпеки [6].

Також слід виокремити Cybersecurity Act (2019). Даний документ вводить систему сертифікації безпеки IoT-пристроїв, спрямовану на забезпечення їхньої надійності та відповідності стандартам кібербезпеки. Відповідно до цього акту, Європейське агентство з кібербезпеки (ENISA) отримує повноваження на розробку схем сертифікації для IoT-пристроїв, програмного забезпечення та послуг. Ця сертифікація дозволяє оцінити рівень безпеки IoT-продуктів перед їхнім виходом на ринок та гарантує, що вони відповідають європейським стандартам. Крім того, акт сприяє створенню єдиного підходу до кібербезпеки IoT у країнах ЄС, що допомагає уникнути фрагментації законодавства та забезпечити належний рівень захисту даних користувачів.

Ще варто сказати про цифрову стратегію Великої Британії (2017) – це комплексний документ, який визначає ключові напрями розвитку цифрової економіки країни, включаючи заходи безпеки для впровадження IoT. Стратегія охоплює питання кібербезпеки, регулювання обміну даними, управління ризиками та стандартизації IoT-технологій. Основними її положеннями є забезпечення стійкості мережевої інфраструктури, розробка механізмів захисту конфіденційної інформації користувачів IoT-пристроїв та сприяння впровадженню безпечних IoT-рішень у сфері промисловості, транспорту та охорони здоров'я. Окрему увагу в документі приділено необхідності співпраці державного та приватного секторів для створення єдиної екосистеми кібербезпеки, яка зможе ефективно реагувати на сучасні виклики в сфері IoT.

У США особливу роль у регулюванні IoT відіграє Закон Каліфорнії про IoT (SB-327), який зобов'язує виробників впроваджувати унікальні паролі та механізми

безпечної автентифікації. Даний закон став основою для подальших федеральних ініціатив щодо безпеки IoT-пристроїв [7].

Окремо слід виділити Кіберстратегію Європейського Союзу (EU Cybersecurity Strategy), яка передбачає сертифікацію IoT-пристроїв, створення механізмів швидкого реагування на загрози та розширення можливостей кіберзахисту на рівні держав-членів ЄС. Однією з її цілей є формування надійного середовища для використання IoT у державних установах, критичних галузях економіки та серед звичайних споживачів**.**

Ще слід коротко описати стандарти в інших країнах. Регулювання в інших країнах

- Китай ухвалив Державну програму розвитку IoT, яка передбачає створення 500 "розумних міст".
- Південна Корея у 2014 році прийняла Генеральний план створення IoT.
- Японія у 2016 році опублікувала документ "Загальні положення безпеки систем Інтернету речей", що є частиною Національної стратегії кібербезпеки.

В Україні нормативно-правова база щодо Інтернету речей поки що перебуває на стадії формування, і значна частина питань, пов’язаних із безпекою IoT, регулюється загальними законодавчими актами. Одним із основних документів у цій сфері є Закон України "Про захист інформації в інформаційно-телекомунікаційних системах", який встановлює вимоги до безпеки даних та регламентує порядок їхнього зберігання, передачі та захисту. Однак цей закон не містить конкретних норм щодо особливостей функціонування IoT-систем.

Ще одним важливим законодавчим актом є Закон України "Про захист персональних даних", який регулює збір, обробку та зберігання персональних даних, включаючи дані, що генеруються IoT-пристроями. Відсутність чітких вимог щодо обробки інформації, яку збирають IoT-пристрої, створює певні ризики щодо захисту конфіденційності користувачів і можливості витоку даних [8].

Деякі аспекти цифрового розвитку, включаючи Інтернет речей, згадуються у Концепції розвитку цифрової економіки України (2018-2020). Ця концепція визначає основні напрями цифрової трансформації, серед яких використання IoT у

промисловості, транспорті та розумних містах. Однак документ не має сили закону і не містить детальних вимог щодо впровадження безпечних IoT-рішень.

На сьогодні в Україні відсутня спеціалізована нормативно-правова база, яка б регулювала питання безпеки IoT на рівні окремих законів або підзаконних актів. Це створює потенційні ризики для користувачів та операторів IoT-систем, оскільки немає чітких правил щодо автентифікації пристроїв, захисту комунікацій, сертифікації IoT-обладнання та відповідальності за можливі загрози. Для вирішення цих проблем необхідно розробити комплексний законодавчий підхід, що включатиме вимоги до захисту IoT-інфраструктури, механізми запобігання кібератакам та стандарти взаємодії пристроїв у межах єдиної екосистеми кібербезпеки.

Стандартизація IoT. Окрім законодавчих ініціатив, значну роль у забезпеченні безпеки IoT відіграють міжнародні стандарти, які встановлюють вимоги до управління інформаційною безпекою, захисту мережевої інфраструктури, обробки даних та кібербезпеки пристроїв [9].

- ISO/IEC 27001 – стандарт управління інформаційною безпекою, який визначає вимоги до створення, впровадження, підтримки та постійного вдосконалення системи управління інформаційною безпекою (ISMS). Він забезпечує організаціям комплексний підхід до управління ризиками безпеки інформації, включаючи аспекти IoT.
- ISO/IEC 27030 – специфічний стандарт, що визначає вимоги до безпеки IoT-систем, включаючи принципи управління ризиками, захисту конфіденційності, автентифікації пристроїв та стійкості до атак. Він допомагає організаціям застосовувати ефективні заходи безпеки для IoT-інфраструктур.
- NIST Special Publication 800-183 – документ, який надає рекомендації щодо архітектури IoT, зокрема визначає основні компоненти IoT-систем, ризики безпеки та методи їхньої мінімізації. Виданий Національним інститутом стандартів і технологій США (NIST), цей стандарт є важливим орієнтиром для організацій, що впроваджують IoT-технології.
- ITU-T Y.2060, Y.2063, Y.2069 – серія стандартів, розроблених Міжнародним союзом електрозв’язку (ITU), які регламентують загальні принципи роботи IoT,

архітектурну модель, термінологію та вимоги до функціонування мереж IoT. Ці стандарти мають міжнародне значення та застосовуються в різних галузях.

- IEEE 2413 – стандарт, розроблений Інститутом інженерів електротехніки та електроніки (IEEE), що визначає структуру IoT-систем, уніфікує підхід до розробки, розгортання та інтеграції IoT-рішень у різних сферах, таких як розумні міста, енергетика, медицина та промисловість.
- ENISA (Агентство Європейського Союзу з кібербезпеки) – надає рекомендації щодо базової безпеки IoT, включаючи заходи щодо захисту мереж, виявлення загроз та реагування на інциденти. ENISA розробляє керівні принципи для підвищення безпеки IoT-пристроїв та інфраструктур у країнах ЄС.

А тепер розглянемо більш детально ITU Y, ENISA та IEEE.

Міжнародний союз електрозв'язку (ITU-T) розробляє стандарти для телекомунікаційних технологій, включаючи IoT.

ITU-T Y.2060 (2012) (огляд Інтернету речей) розглянемо його більш детально:

- Визначає, що таке IoT, його можливості та застосування.
- Описує загальну функціональну архітектуру IoT (пристрої, мережі, хмарні сервіси).
- Включає поняття масової взаємодії пристроїв та збору даних у реальному часі.

ITU-T Y.2063 (основа WEB речей, Web of Things, WoT) - Доповнює концепцію IoT, зосереджуючись на інтеграції Інтернету речей із веб-технологіями:

- Описує механізми взаємодії IoT-пристроїв через веб-сервіси та API.
- Включає принципи використання протоколів HTTP, RESTful API, MQTT для взаємодії IoT-пристроїв.
- Дозволяє інтегрувати IoT у бізнес-додатки через веб-технології.

ITU-T Y.2069 (терміни і визначення Інтернету речей) - цей стандарт встановлює загальні терміни та визначення, що використовуються у сфері IoT:

- Уніфікує понятійний апарат для дослідників, інженерів та компаній.
- Описує терміни, такі як IoT-платформа, сенсорні мережі, шлюзи, взаємодія між пристроями**.**

- Визначає категорії пристроїв та мереж, що використовуються у IoT.

IEEE 2413 (структурний шаблон Інтернету речей) – міжнародний інститут інженерів з електротехніки та електроніки (IEEE) розробив IEEE 2413, який пропонує єдину архітектуру для IoT-систем:

- Визначає багаторівневу модель IoT, яка включає фізичні пристрої, комунікаційні протоколи та рівень управління даними.
- Дозволяє стандартизувати взаємодію пристроїв через спільні API та програмні рішення.
- Включає аспекти безпеки, управління доступом та довіри в IoT-системах.
- Враховує галузеві потреби (розумні міста, медицина, промисловість).

ENISA – Рекомендації щодо базової безпеки IoT. ENISA (European Union Agency for Cybersecurity) – це агентство ЄС, яке займається кібербезпекою.

Основні рекомендації ENISA щодо безпеки IoT:

1. Ідентифікація загроз IoT:
- Визначення основних атак: ботнети (Mirai), підміна пристроїв, DDoS-атаки.
- Аналіз слабких місць IoT, пов'язаних із недостатньою автентифікацією.

2. Безпека на рівні пристрою:
- Обов’язкове використання шифрування даних (AES, TLS, VPN).
- Надійна автентифікація пристроїв (двофакторна, сертифікати).
- Вбудовані механізми оновлення програмного забезпечення.

3. Безпечні мережеві протоколи для IoT:
- Використання MQTT, CoAP з захистом на рівні Transport Layer Security (TLS).
- Захист LPWAN та 5G-з’єднань для IoT-пристроїв.

4. Захист конфіденційності:
- Мінімізація збору персональних даних.
- Дотримання регламенту GDPR для IoT-пристроїв у ЄС.

5. Рекомендації для виробників IoT:
- Вбудовування механізмів автоматичного оновлення.
- Використання Zero Trust Architecture для IoT.

- Впровадження AI-based Intrusion Detection Systems (IDS/IPS) для моніторингу атак.

Міжнародна стандартизація IoT є важливою складовою забезпечення кібербезпеки, оскільки дозволяє виробникам і постачальникам послуг дотримуватися єдиних вимог та рекомендацій. Використання стандартів сприяє підвищенню довіри до IoT-технологій, забезпечує їхню сумісність та допомагає запобігти потенційним загрозам [11].

IWF зосереджує увагу на ширшому аспекті розробки додатків, проміжного програмного забезпечення та підтримуючих функцій для корпоративного Інтернету речей. У цьому контексті на IoT World Forum працюють над створенням узагальненої горизонтальної архітектурної моделі, яка забезпечить взаємну сумісність усіх IoT-компонентів: пристроїв і контролерів, мереж, сховищ даних, додатків і аналітики (рис. 1.4).

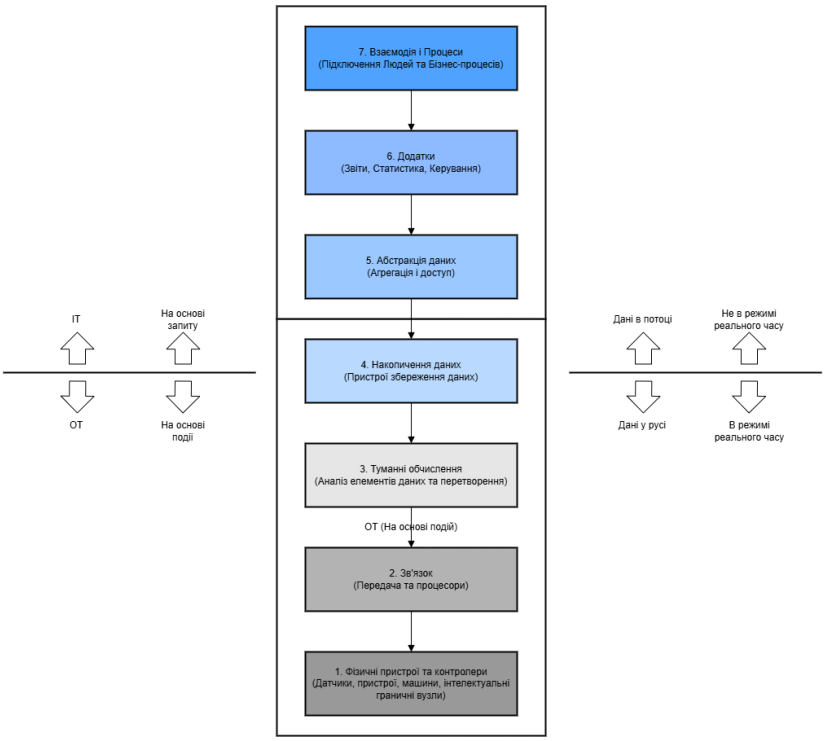

Рис. 1.4 Еталонна модель IoT від IWF

Запропонована форумом еталонна графічна модель розподіляє ці компоненти за рівнями, що сприяє створенню «відкритих IoT-систем» із гарантованою сумісністю. Семирівнева модель, яку запропонував IWF, представлена на рис. 1.4.

1.4 Сфери використання iot

Пристрої IoT охоплюють різноманітні сфери діяльності, де вони підвищують ефективність, зменшують витрати та забезпечують нові рівні автоматизації.

Побутова сфера використання.

Розумний будинок (Smart Home) – це концепція житлового приміщення, оснащеного системам0438 автоматизації, які керують освітленням, кліматом, безпекою, побутовою технікою та іншими пристроями за допомогою Інтернету речей (IoT). Головною метою розумного будинку є підвищення комфорту, безпеки, енергоефективності та інтеграція всіх домашніх пристроїв у єдину екосистему, яка може управлятися дистанційно через смартфон, голосові команди або запрограмовані сценарії.

Ідея розумного будинку бере свій початок у 1970-х роках, коли були розроблені перші системи домашньої автоматизації X10. Ця технологія дозволяла передавати сигнали через електромережу для дистанційного керування побутовими приладами. У 1990-х з’явилися перші комерційні системи розумного будинку, що використовували дротові протоколи зв’язку. З початку 2000-х років розвиток бездротових технологій (Wi-Fi, ZigBee, Z-Wave) та Інтернету речей сприяв масовому впровадженню розумних пристроїв. Сьогодні такі компанії, як Google, Amazon, Apple, Xiaomi та Samsung, активно розвивають екосистеми розумного будинку.

На рис.1.5 показано процес передачі даних через хмарну архітектуру в розумному будинку. Процес створення розумного будинку починається з проєктування, коли визначаються потреби користувача, підбираються пристрої та складається план інтеграції. Далі вибирається платформа управління, наприклад, Google Home, Apple HomeKit або Amazon Alexa, яка об'єднує всі пристрої в єдину систему. Після цього здійснюється інтеграція пристроїв, таких як розумне освітлення, датчики руху, системи безпеки та термостати [12]. На наступному етапі налаштовуються сценарії автоматизації, що дозволяють, наприклад, вмикати світло при вході до приміщення або регулювати температуру залежно від часу доби.

Завершальним етапом є тестування та оптимізація системи для забезпечення її ефективної роботи.

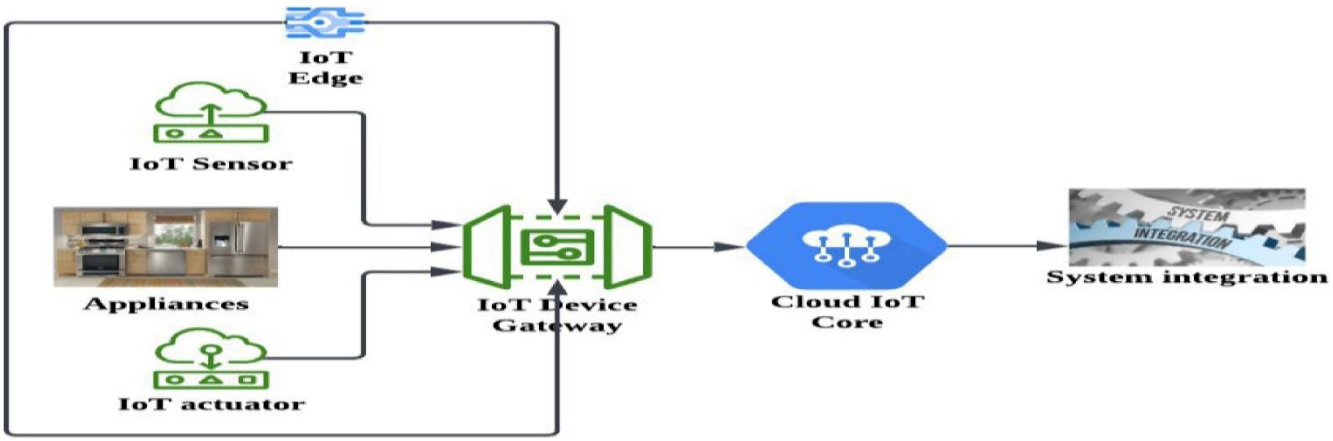

Рис. 1.5 Процес передачі даних через хмарну архітектуру

Функціональність розумного будинку включає автоматизоване освітлення, клімат-контроль, системи безпеки, контроль енергоспоживання, автоматизацію побутових процесів та інтеграцію розважальних систем. Автоматизоване освітлення дозволяє керувати світлом за допомогою датчиків руху, голосових команд або мобільного додатка. Клімат-контроль, який реалізується через розумні термостати, підтримує оптимальну температуру та вологість у приміщенні. Системи безпеки включають камери спостереження, датчики руху, розумні дверні замки та охоронні сигналізації, що забезпечують захист житла. Контроль енергоспоживання здійснюється через розумні розетки та лічильники, які допомагають знижувати витрати на електроенергію. Автоматизація побутових процесів передбачає інтеграцію техніки, наприклад, кавоварок, пилососів-роботів та розумних холодильників, що спрощує повсякденні завдання. Крім того, домашні розважальні системи, такі як мультимедійні центри та голосові помічники, створюють додатковий комфорт.

Однією з найбільш розвинених сфер використання IoT є автоматизація житла. Основні напрямки включають:

- Розумні термостати (*Google Nest, Ecobee*) – аналізують поведінку мешканців, навчаються їхнім звичкам та автоматично регулюють температуру в залежності від погодних умов і присутності людей.

- Інтелектуальні системи безпеки (*Ring, Arlo, Xiaomi*) – містять камери високої роздільної здатності, датчики руху, дверні замки з біометричною аутентифікацією та системи розпізнавання облич.

- Автоматизоване освітлення (*Philips Hue, Yeelight*) – регулюється на основі часу доби, присутності користувачів у кімнатах, рівня природного освітлення та може змінювати кольорову температуру для створення потрібної атмосфери.

- Розумні побутові прилади – холодильники (*Samsung Family Hub*) можуть відстежувати запаси продуктів і пропонувати рецепти; пральні машини визначають рівень забруднення одягу та оптимізують споживання води; пилососи-роботи (*iRobot Roomba*) інтегруються з голосовими асистентами та автоматично будують карту приміщення для ефективного прибирання; кавоварки та кухонні гаджети можуть готувати напої за розкладом або реагувати на команди користувача через мобільний додаток.

- Автоматизація поливу та догляду за рослинами – інтелектуальні системи визначають рівень вологості ґрунту та автоматично подають воду в потрібному обсязі, зменшуючи витрати води та підвищуючи ефективність догляду за зеленими насадженнями.

Перспективи розвитку розумних будинків охоплюють впровадження штучного інтелекту, який дозволить системам самостійно адаптуватися до звичок мешканців, розширену інтеграцію з технологіями "розумного міста", що забезпечить взаємодію з транспортною, комунальною та екологічною інфраструктурою, а також поліпшення стандартів безпеки для захисту від кібератак. Важливим напрямком розвитку є також використання енергоефективних рішень, таких як сонячні панелі та розумні системи опалення, що дозволять значно зменшити витрати на комунальні послуги [13].

Транспортна та логістична сфера використання.

Інтернет речей (IoT) у транспорті та логістиці – це система підключених до мережі пристроїв, які забезпечують збір, аналіз та обмін даними з метою підвищення ефективності, безпеки та автоматизації транспортних та логістичних процесів. Основною метою є оптимізація перевезень, зниження витрат, зменшення впливу на довкілля та підвищення рівня комфорту для користувачів.

Перші системи моніторингу транспорту почали впроваджуватися у 1980-х роках у вигляді GPS-трекерів для відстеження руху вантажівок. З розвитком бездротового зв'язку та Інтернету речей у 2000-х роках почали з'являтися інтелектуальні транспортні системи (ITS), що включають сенсори дорожнього руху, телематику, інтелектуальне керування світлофорами та моніторинг стану транспортних засобів.

Автомобільна безпека має чотирирівневу архітектуру, включаючи захищені інтерфейси, захищені шлюзи та захищені мережі та захищену обробку, як показано на рис.1.6.

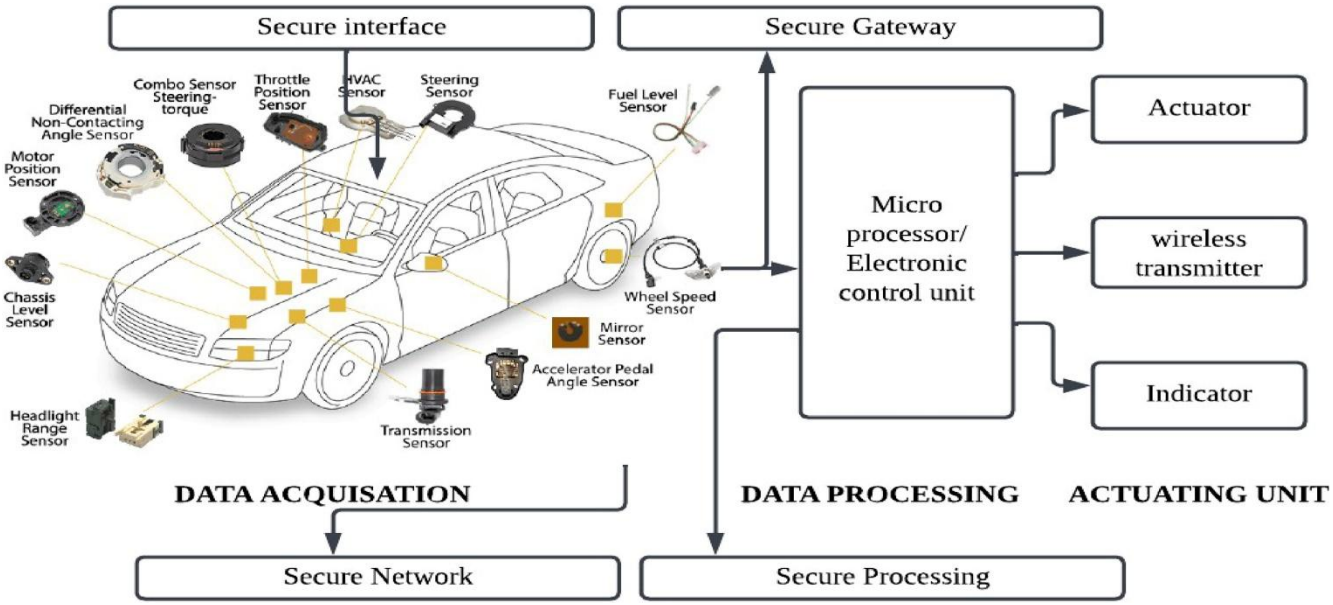

Рис. 1.6 Автомобільна безпека

Як створюється і використовується IoT у транспорті та логістиці

- Моніторинг транспорту – GPS-трекери, акселерометри та інші датчики відстежують місцеположення, швидкість, стан транспорту, аналізують поведінку водіїв та виявляють потенційно небезпечні ситуації.

- Автоматизація вантажоперевезень – логістичні компанії використовують IoT для прогнозування часу доставки, розрахунку оптимальних маршрутів, автоматизації складування та контролю рівня завантаження транспорту.

- Безпілотний транспорт – автономні автомобілі, дрони та роботизовані системи доставки зменшують потребу в людському факторі, покращують точність і безпеку перевезень, знижують затримки та експлуатаційні витрати.

- Інтелектуальні системи управління трафіком – датчики на дорогах, розумні світлофори та камери контролюють потоки транспорту, аналізують затори та автоматично змінюють режими світлофорів для покращення трафіку.

- Моніторинг стану транспортних засобів – IoT-сенсори виявляють зношення деталей, стежать за рівнем пального, діагностують стан двигуна та автоматично планують технічне обслуговування, що зменшує ризик аварій.

- Автоматизоване управління паркінгом – розумні паркувальні системи визначають наявність вільних місць, інформують водіїв через мобільні додатки та здійснюють безконтактну оплату.

- Моніторинг вантажу – сенсори температури, вологості та ударів гарантують збереження вантажів у належних умовах під час перевезень, що є критично важливим для фармацевтики, харчової та хімічної промисловості.

Функції IoT у транспорті та логістиці:

- GPS-моніторинг транспорту – дозволяє контролювати місцеположення транспортних засобів у реальному часі, допомагаючи оптимізувати маршрути, забезпечувати безпеку перевезень і запобігати крадіжкам або несанкціонованому використанню транспортних засобів.

- Оптимізація маршрутів – за допомогою алгоритмів штучного інтелекту та аналізу дорожньої ситуації система може прокладати найбільш ефективні маршрути, враховуючи погодні умови, завантаженість доріг і витрати на паливо.

- Безпека пасажирів та водіїв – інтелектуальні системи контролю руху включають камери відеоспостереження, датчики виявлення зіткнень, моніторинг стану водія (визначення втоми, підвищеного серцебиття) і аварійне гальмування, що зменшує ризик ДТП.

- Моніторинг вантажу – сенсори контролюють температуру, вологість, тиск і безпеку вантажів під час перевезень, що особливо важливо для транспортування продуктів харчування, медичних препаратів та небезпечних матеріалів.
- Інтеграція з розумними містами – взаємодія IoT-транспортної інфраструктури з міськими системами управління дорожнім рухом, розумними світлофорами та автоматичними станціями технічного обслуговування для мінімізації заторів і підвищення екологічності транспорту.
- Автоматизовані системи паркування – використання датчиків і мобільних додатків для визначення вільних місць на паркінгах у режимі реального часу, що знижує витрати часу водіїв і сприяє зменшенню заторів у міських зонах.

Перспективи розвитку IoT у транспорті та логістиці:

- Розвиток безпілотного транспорту – покращення технологій автономного водіння, що дозволяє мінімізувати вплив людського фактора та підвищити ефективність перевезень.
- Використання AI та Big Data – глибокий аналіз транспортних потоків, автоматична зміна дорожніх знаків у залежності від трафіку та створення адаптивних систем управління рухом.
- Перехід на екологічний транспорт – інтеграція IoT із зарядними станціями для електромобілів, управління споживанням енергії та розробка водневих транспортних систем.
- Гіперінтелектуальні транспортні мережі – створення єдиної екосистеми транспорту, в якій громадський, комерційний та приватний транспорт зможуть взаємодіяти через спільні хмарні платформи, що дозволить зменшити витрати та підвищити безпеку перевезень.
- Інтелектуальне обслуговування транспортних засобів – використання IoT для прогнозування необхідності технічного обслуговування, що дозволяє уникати несподіваних поломок і аварій на дорогах.

Медична сфера.

Медичний Інтернет речей (IoMT – Internet of Medical Things) – це екосистема взаємопов'язаних інтелектуальних медичних пристроїв, сенсорів, мобільних додатків, програмного забезпечення та хмарних сервісів, які використовуються для збору, моніторингу та аналізу медичних даних у реальному часі. IoMT інтегрує технології телемедицини, штучного інтелекту, великих даних (Big Data), блокчейну та кібербезпеки для вдосконалення діагностики, оптимізації лікування, віддаленого моніторингу пацієнтів та автоматизації лікарських процесів. На рис.1.7 на блок-схемі показано процес охорони здоров'я на основі Інтернету речей через хмару.

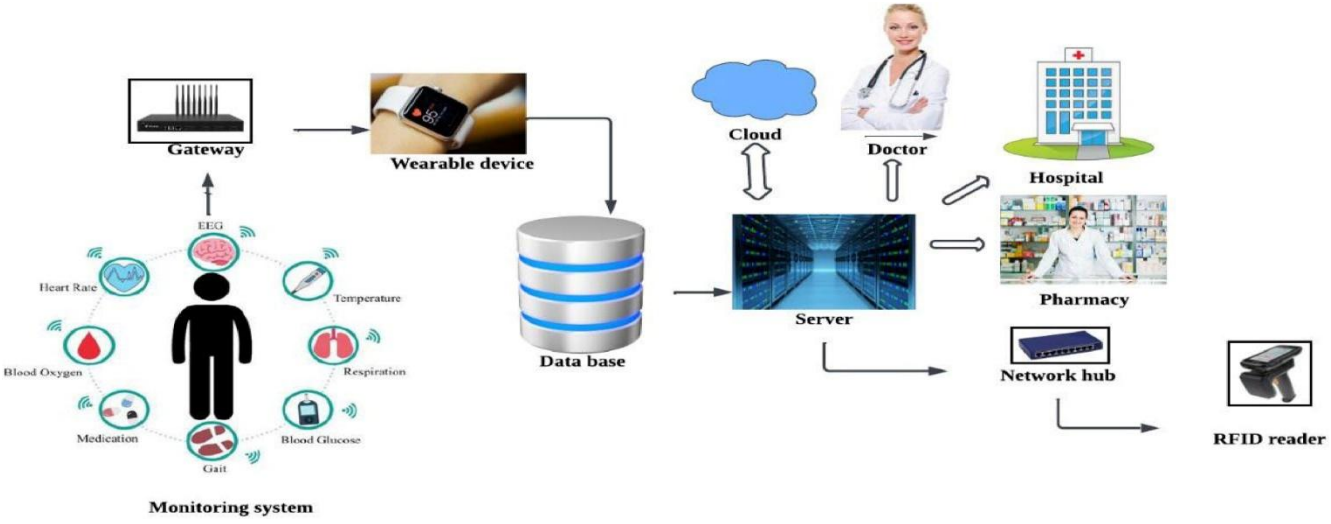

Рис. 1.7 Процес охорони здоров'я на основі Інтернету речей

IoMT дозволяє зменшити людський фактор у діагностиці та лікуванні, покращуючи точність медичних рішень, мінімізуючи ризики лікарських помилок і значно покращуючи доступність медичних послуг. Ця технологія також сприяє розвитку превентивної медицини, дозволяючи прогнозувати загрози здоров'ю на ранніх етапах.

Основною метою IoMT є підвищення ефективності медичних послуг, зниження навантаження на лікарів та медичні установи, покращення доступу до медичної допомоги, особливо у віддалених регіонах, а також запобігання критичним станам за допомогою ранньої діагностики та прогнозного аналізу.

Концепція підключених медичних пристроїв бере початок у 1960-х роках, коли почали з'являтися перші кардіостимулятори. У 1990-х роках медичні технології

значно вдосконалилися завдяки розвитку цифрової обробки сигналів, а у 2000-х – зростання швидкості бездротового зв'язку дозволило лікарям дистанційно отримувати дані про стан пацієнтів. Сучасний IoMT став можливим завдяки широкому поширенню сенсорних технологій, хмарних обчислень та мобільних додатків.

Як створюється та використовується IoMT

- Підключені медичні пристрої – датчики, імпланти, носимі пристрої та діагностичне обладнання збирають дані про здоров'я пацієнта та передають їх у систему для аналізу.
- Передача даних у хмару – використання Wi-Fi, Bluetooth, 5G, LPWAN та інших бездротових технологій для безперебійної передачі медичних даних між пристроями, лікарями та медичними закладами.
- Обробка та аналіз інформації – алгоритми штучного інтелекту та великі дані аналізують отримані показники, прогнозують можливі ризики та допомагають приймати рішення щодо лікування.
- Інтеграція з електронними медичними картками (EHR) – автоматичне оновлення записів пацієнта для забезпечення точного та актуального діагностичного профілю, що дозволяє лікарям швидко отримувати необхідну інформацію.
- Дистанційний моніторинг пацієнтів – лікарі можуть переглядати дані в режимі реального часу та коригувати лікування, що особливо корисно для пацієнтів із хронічними захворюваннями або післяопераційного догляду.
- Розумні лікарні – автоматизація управління лікарняними ресурсами, моніторинг стану медичних приладів, контроль доступу до лікарняних приміщень та підтримка адміністративних процесів.

Функції IoMT

- Телемедицина – забезпечує консультації лікарів через відеозв'язок, обмін цифровими даними та дистанційний моніторинг стану здоров'я.
- Розумні медичні пристрої – кардіостимулятори, глюкометри, інсулінові помпи, розумні годинники та браслети (Apple Watch, Fitbit, Garmin), що відстежують серцевий ритм, рівень кисню в крові, артеріальний тиск та інші показники.

- Автоматизація лікарень – використання IoT для контролю стану пацієнтів, управління медикаментами, моніторингу операційних залів, моніторингу роботи обладнання та підтримки медичного персоналу.
- Прогнозування загроз здоров'ю – аналіз медичних даних дозволяє виявляти ризики розвитку хронічних захворювань, прогнозувати інсульти, серцеві напади та інші критичні стани.
- Дистанційне спостереження за хворими – пацієнти можуть отримувати безперервний моніторинг та персоналізовані рекомендації без необхідності частих візитів до лікаря.
- Медична робототехніка – автономні системи допомоги у хірургії (Da Vinci Surgical System), мобільні роботи для доставки ліків та аналізів, а також розумні протези та екзоскелети.
- Керування запасами ліків – сенсорні системи відстежують запаси медикаментів у лікарнях та аптеках, автоматично замовляючи необхідні препарати.
- Реабілітаційні технології – IoMT допомагає у фізичній реабілітації пацієнтів після травм або операцій, використовуючи розумні ортези, системи моніторингу рухової активності та аналізу відновлення організму.

Перспективи розвитку IoMT

- Інтеграція штучного інтелекту – використання AI для діагностики, прогнозування розвитку хвороб та персоналізованого підбору лікування.
- Розвиток біометричних технологій – вдосконалення пристроїв для точнішого аналізу біологічних показників, таких як аналіз крові в режимі реального часу.
- Впровадження 5G – підвищення швидкості та надійності передавання медичних даних у режимі реального часу для зменшення затримок у комунікаціях.
- Розширення автономних систем охорони здоров'я – використання роботизованих асистентів для догляду за пацієнтами, автономних лабораторій та мобільних клінік.
- Посилена кібербезпека – розробка надійних методів шифрування та захисту медичних даних від кіберзагроз, включаючи блокчейн-технології.

- Нанотехнології та імплантати – розвиток біосенсорів, мікрочіпів та внутрішньотканинних пристроїв для безперервного збору медичних даних.
- Генетична медицина та IoMT – використання IoT для збору даних про генетичні схильності та розробки індивідуальних методів лікування, що сприятиме персоналізованій медицині.

Завдяки IoT-технологіям лікарі можуть швидше реагувати на критичні стани, покращувати діагностику та створювати персоналізовані схеми лікування, що значно покращує якість життя пацієнтів.

1.5 Методи управління безпекою iot

Основними методами управління безпекою IoT є стандартизація, політики безпеки, контроль доступу, аутентифікація та ідентифікація, криптографічний захист, моніторинг та виявлення загроз, управління оновленнями, а також застосування програмно-конфігурованих мереж (SDN) для гнучкого керування безпекою інформаційних потоків [14].

1. Стандартизація та політики безпеки IoT

Одним із ключових методів управління безпекою IoT є впровадження міжнародних стандартів та регуляторних вимог. Такі стандарти, як ISO/IEC 27001, NIST SP 800-183 та IEC 62443, містять вимоги щодо захисту даних, мережевої безпеки та управління ризиками. Встановлення внутрішніх політик безпеки, що регламентують використання IoT-пристроїв, є важливим кроком у запобіганні несанкціонованому доступу та витоку інформації.

2. Контроль доступу

Контроль доступу є основним засобом управління безпекою IoT, що дозволяє обмежувати доступ до пристроїв та даних лише авторизованим користувачам та системам. Використання ефективних методів контролю доступу значно знижує ризики витоку даних та атак на IoT-інфраструктуру. Основні підходи включають:

- Рольовий контроль доступу (RBAC) – метод, при якому права доступу надаються відповідно до ролей користувачів у системі. Наприклад, у розумному

будинку адміністратор може мати повний контроль над усіма пристроями, тоді як користувачі з роллю "гості" матимуть лише обмежений доступ, наприклад, до системи освітлення.

- Атрибутний контроль доступу (ABAC) – більш гнучкий підхід, який враховує кілька атрибутів (ідентичність користувача, місцезнаходження, час доби, тип пристрою тощо). Наприклад, система безпеки на виробничому підприємстві може дозволяти доступ до конфіденційних даних лише з пристроїв, підключених через корпоративну мережу, та в робочий час.
- Модель нульової довіри (Zero Trust Architecture, ZTA) – концепція, згідно з якою жоден пристрій або користувач не вважається надійним за замовчуванням. Для доступу до IoT-пристроїв кожен запит повинен проходити багаторівневу перевірку автентичності та відповідати політикам безпеки. Наприклад, у корпоративному середовищі доступ до мережевого принтера може бути дозволений лише після перевірки пристрою на відповідність безпековим стандартам компанії.

Поєднання цих методів забезпечує багаторівневий контроль доступу та значно підвищує загальний рівень безпеки IoT-екосистеми.

3. Моніторинг, виявлення та реагування на загрози

Постійний моніторинг IoT-мереж є важливою складовою управління безпекою. Основні методи включають:

- Системи виявлення та запобігання атак (IDS/IPS) – аналізують мережевий трафік та блокують потенційні атаки, наприклад, DDoS-атаки чи спроби несанкціонованого доступу.
- Аналіз поведінки пристроїв – виявлення аномалій у роботі IoT-пристроїв на основі машинного навчання. Наприклад, якщо розумний термостат починає надсилати незвично великий обсяг даних у нічний час, система безпеки може визначити це як потенційну загрозу.
- SIEM-системи (Security Information and Event Management) – централізований збір та аналіз журналів подій безпеки. Наприклад, система SIEM може виявити спроби входу з підозрілих IP-адрес і вжити відповідних заходів.

- Автоматизовані засоби реагування (SOAR) – автоматизація процесів виявлення та ліквідації загроз. Наприклад, якщо виявлено ботнет-атаку, система SOAR може автоматично заблокувати скомпрометовані пристрої та повідомити адміністратора.

4. Управління оновленнями та патч-менеджмент

Одним із найпоширеніших способів атак на IoT-пристрої є експлуатація вразливостей у програмному забезпеченні. Тому важливим методом управління безпекою є ефективне управління оновленнями, що включає:

- Автоматизоване оновлення прошивки та програмного забезпечення – регулярне застосування патчів безпеки для усунення вразливостей. Наприклад, виробники розумних камер спостереження випускають оновлення, що виправляють помилки в захисті перед атаками типу Man-in-the-Middle.
- Цифровий підпис оновлень – забезпечення їх автентичності та запобігання компрометації. Наприклад, оновлення для промислових IoT-систем підписуються виробником, що унеможливлює встановлення шкідливих модифікацій.
- Перевірка сумісності та тестування оновлень – запобігання збоям у роботі IoT-пристроїв. Наприклад, у розумному будинку оновлення прошивки розумного термостата проходить тестування на сумісність з іншими пристроями, щоб уникнути збоїв у системі управління кліматом [15].

5. Використання програмно-конфігурованих мереж (SDN) для безпеки IoT

Технологія SDN (Software-Defined Networking) дозволяє централізовано керувати безпекою інформаційних потоків у мережах IoT, гнучко адаптуючи політики безпеки до поточних загроз. Завдяки SDN адміністратори можуть створювати програмно-керовані правила маршрутизації трафіку, виявляти потенційні загрози в реальному часі та динамічно змінювати конфігурацію мережі для захисту від атак.

Основні переваги SDN у сфері IoT:

- Гнучке управління політиками безпеки – адміністратор може оперативно змінювати політики доступу та фільтрації трафіку. Наприклад, у випадку виявлення

підозрілої активності SDN-контролер може автоматично змінити маршрутизацію трафіку або блокувати підключення певних IoT-пристроїв.

- Динамічне виявлення та реагування на загрози – SDN-контролер може аналізувати мережевий трафік у реальному часі, ідентифікувати аномальні шаблони поведінки та автоматично перенаправляти підозрілий трафік на аналіз. Наприклад, у випадку DDoS-атаки SDN може обмежити пропускну здатність для зловмисного трафіку або направити його через спеціальні вузли для фільтрації.
- Резервування контролерів та маршрутизації – SDN-архітектура дозволяє створювати резервні контролери, які забезпечують безперервність роботи мережі в разі виходу основного контролера з ладу. Наприклад, у розумних містах SDN може автоматично перенаправляти трафік сенсорних мереж для підтримання стабільності комунікацій між IoT-пристроями навіть при локальних збоях у мережі.
- Інтеграція з SIEM-системами – поєднання SDN із системами управління подіями інформаційної безпеки (SIEM) дозволяє централізовано аналізувати безпекові інциденти та приймати рішення щодо реакції на загрози.

6. Захист хмарних сервісів IoT

Одним із важливих аспектів управління безпекою IoT є захист хмарних платформ, які використовуються для обробки та зберігання даних. Найбільш популярними хмарними рішеннями є Amazon Web Services (AWS), Microsoft Azure IoT, Google Cloud IoT, які пропонують багаторівневі механізми безпеки.

Основні методи захисту хмарних IoT-сервісів включають:

- Модулі розпізнавання та запобігання DDoS-атак, що використовують машинне навчання для виявлення аномального трафіку.
- Системи розподілу трафіку, що забезпечують рівномірний розподіл навантаження між серверами.
- Захист DNS – запобігання підробці доменних імен та DNS-атакам.
- Контроль доступу через IAM (Identity and Access Management) – багаторівнева автентифікація користувачів з різними рівнями доступу.

Наприклад, у Google Cloud IoT для аутентифікації використовується JSON Web Tokens (JWT), які періодично змінюються, щоб зменшити ризик компрометації

ключів. Це підвищує безпеку і запобігає довготривалому використанню перехоплених даних доступу.

7. Автентифікація та шифрування для периферійних пристроїв

Захист IoT-пристроїв передбачає використання надійних алгоритмів автентифікації та шифрування. Найпоширеніші підходи:

- Використання сертифікатів X.509 для перевірки автентичності пристроїв перед підключенням до мережі.
- Застосування симетричних ключів разом із алгоритмом AES-256 для шифрування даних під час передачі між пристроєм і хмарою.
- Протоколи безпечного з’єднання, такі як TLS 1.2/1.3, що забезпечують захист від атак "людина посередині" (MitM).

Наприклад, у промислових IoT-системах використовується двофакторна аутентифікація разом із цифровими підписами для перевірки справжності прошивок перед оновленням пристроїв.

8. Використання штучного інтелекту для моніторингу загроз

Одним із новітніх підходів до управління безпекою IoT є використання AI/ML для аналізу мережевого трафіку та поведінки пристроїв. Основні методи:

- Аналіз поведінки пристроїв – системи машинного навчання можуть виявляти аномалії у поведінці IoT-пристроїв та попереджати про можливі загрози.
- Автоматизовані системи реагування (SOAR) – технологія, що поєднує аналітику, виявлення атак та автоматизоване реагування на інциденти.
- SIEM-системи – централізовані системи моніторингу безпеки, які збирають та аналізують дані з усіх IoT-компонентів у реальному часі.

Наприклад, у розумних містах системи безпеки можуть автоматично виявляти підозрілу активність вуличних камер або сенсорів, наприклад, зміну типового зразка трафіку в мережі або спроби несанкціонованого доступу.

1.6 Основні механізми захисту: аутентифікація, криптографія та інші

З розвитком технологій Інтернету речей (IoT) питання безпеки набуває все більшої актуальності. Велика кількість підключених пристроїв створює нові виклики для забезпечення конфіденційності, цілісності та доступності даних. Ефективне управління безпекою IoT передбачає застосування комплексного підходу, що включає адміністративні, технічні та організаційні методи. Основними методами управління безпекою IoT є стандартизація, політики безпеки, контроль доступу, аутентифікація та ідентифікація, криптографічний захист, моніторинг та виявлення загроз, управління оновленнями, а також застосування програмно-конфігурованих мереж (SDN) для гнучкого керування безпекою інформаційних потоків [16].

Забезпечення безпеки IoT-систем базується на трьох ключових механізмах: аутентифікації, криптографії та управлінні доступом. Ці компоненти працюють разом для запобігання несанкціонованому доступу, збереження конфіденційності переданих даних та контролю привілеїв користувачів і пристроїв.

Аутентифікація

Аутентифікація забезпечує перевірку ідентичності користувачів і пристроїв у мережі IoT. Сучасні системи використовують кілька рівнів аутентифікації:

- Однофакторна аутентифікація (1FA) – включає прості методи, такі як введення пароля або PIN-коду. Використовується в простих IoT-пристроях, але має низьку стійкість до атак, таких як атаки методом перебору (brute force) або фішингові атаки.
- Двофакторна (2FA) та багатофакторна аутентифікація (MFA) – поєднання кількох незалежних методів ідентифікації, наприклад, пароля та біометричних даних (відбитки пальців, розпізнавання обличчя). Використовується у фінансових IoT-системах та системах керування доступом.
- Аутентифікація на основі сертифікатів X.509 – широко використовується у промислових IoT-системах і передбачає обмін цифровими сертифікатами для перевірки справжності пристроїв. Наприклад, у розумних енергомережах (Smart Grid) кожен пристрій отримує унікальний сертифікат для запобігання підробкам.

- Протоколи безпечної автентифікації – OAuth 2.0, OpenID Connect і FIDO2 забезпечують безпарольний вхід та мінімізують ризики компрометації даних. Наприклад, FIDO2 дозволяє IoT-пристроям використовувати апаратні ключі безпеки замість паролів.
- Автентифікація через PKI (інфраструктуру відкритих ключів) – застосовується для забезпечення довіри між IoT-пристроями та серверами, запобігаючи атакам «людина посередині». Наприклад, у логістичних ланцюгах PKI використовується для гарантування справжності переданих даних між транспортними IoT-сенсорами.

Криптографія

Криптографічні методи використовуються для захисту конфіденційності, цілісності та автентичності переданих даних [17]. Основні механізми включають:

- Шифрування даних – алгоритми AES-256, RSA, ECC забезпечують захист даних під час їх передавання та зберігання. Наприклад, у медичних IoT-пристроях AES-256 використовується для захисту персональних медичних даних пацієнтів.
- Легка криптографія – використання оптимізованих алгоритмів для пристроїв з обмеженими обчислювальними ресурсами, наприклад, SPECK та PRESENT. Вони активно застосовуються у сенсорних мережах та вбудованих системах з обмеженим енергоспоживанням.
- Хешування – використання SHA-256, SHA-3 та Bcrypt дозволяє зберігати паролі у вигляді хеш-значень, що унеможливлює їх зворотне відновлення. Наприклад, у мобільних IoT-системах хешування забезпечує безпечне зберігання паролів користувачів.
- Цифрові підписи – механізми ECDSA, RSA-PSS гарантують, що дані не були змінені під час передавання. У фінансових транзакціях цифровий підпис запобігає підробці фінансових документів.
- Квантова криптографія – новітній підхід, що забезпечує стійкість до атак із використанням квантових обчислень. Наприклад, в IoT-інфраструктурах для

урядових зв’язків квантові протоколи забезпечують безпеку передачі конфіденційних даних.

Управління доступом

Управління доступом в IoT-системах необхідне для розмежування прав і контролю взаємодії між пристроями та користувачами. Основні моделі управління доступом включають:

- Рольова модель (RBAC) – права доступу надаються відповідно до ролей користувачів (наприклад, адміністратор, оператор, гість). Використовується в корпоративних IoT-системах для визначення рівня доступу співробітників.
- Атрибутна модель (ABAC) – більш гнучка система, яка враховує множину атрибутів (місцезнаходження, час доби, рівень привілеїв). Наприклад, у розумних будинках доступ до електронних замків може залежати від геолокації смартфона власника.
- Модель нульової довіри (Zero Trust) – передбачає перевірку кожного запиту на доступ незалежно від його джерела. У промислових IoT-мережах ZTA забезпечує захист від внутрішніх загроз, перевіряючи кожну транзакцію незалежно від мережевого рівня.
- Контроль доступу на основі політик (PBAC) – динамічне коригування привілеїв залежно від контексту користувача або пристрою. Використовується в IoT-пристроях для адаптивного налаштування доступу в залежності від рівня ризику в середовищі експлуатації.

1.7 Висновки за першим розділом

У даному розділі було розглянуто ключові аспекти, пов’язані з концепцією, архітектурою, загрозами та нормативно-правовим забезпеченням безпеки Інтернету речей (IoT). На основі аналізу можна зробити кілька важливих висновків.

По-перше, Інтернет речей — це сучасна технологічна парадигма, що поєднує фізичні пристрої з цифровим середовищем, створюючи інтелектуальні екосистеми,

здатні працювати автономно. Його архітектура охоплює кілька рівнів: від фізичних сенсорів до хмарних обчислень і прикладних сервісів. IoT уже сьогодні активно застосовується у промисловості, охороні здоров’я, логістиці, міській інфраструктурі та побуті, що свідчить про його стрімке поширення та важливу роль у цифровій трансформації суспільства.

У той же час, масове впровадження IoT супроводжується масштабними викликами у сфері кібербезпеки, які загрожують не лише інформаційним системам, але й фізичній безпеці людей і критичній інфраструктурі.

Однак одночасно з цим зростає і кіберризик, оскільки кількість підключених пристроїв стрімко збільшується, а рівень їхньої безпеки часто є недостатнім. У ході дослідження було встановлено, що головними загрозами для IoT є апаратні, мережеві, програмні вразливості, а також атаки соціальної інженерії. IoT-пристрої часто мають слабкі механізми автентифікації, використовують відкриті порти, не отримують регулярних оновлень і містять відомі програмні вразливості, що робить їх легкою здобиччю для зловмисників.

Особливо вразливими є пристрої з обмеженими обчислювальними ресурсами, які не підтримують сучасні механізми шифрування, а також пристрої, що використовують застарілі або небезпечні протоколи зв’язку.

Особливо небезпечними є атаки ботнетів (наприклад, Mirai, VPNFilter), атаки типу “людина посередині” (MITM), відмови в обслуговуванні (DDoS), фішинг, підробка оновлень тощо. Як показано у статистичних даних, кількість атак на IoT-пристрої зростає експоненціально: лише у 2022 році було зафіксовано понад 112 мільйонів атак, що свідчить про критичність проблеми на глобальному рівні.

По-друге, вразливості IoT-систем мають системний характер, що зумовлено як слабким апаратним захистом, так і недосконалістю програмного забезпечення, протоколів зв’язку та людським фактором. Окремі категорії загроз охоплюють понад 50 типів атак, серед яких — шкідливе ПЗ, підробка вузлів, фальсифікація даних, атакуючі ботнети, SQL-ін'єкції, зловживання відкритими API, впровадження троянів і шпигунського ПЗ.

Реалізація ефективної системи захисту потребує поєднання технічних, організаційних та правових заходів, адаптованих до особливостей конкретного середовища використання IoT.

По-третє, для протидії зростаючим загрозам важливим є нормативне та стандартне регулювання. У Європейському Союзі цю функцію виконують акти GDPR, NIS2, Cybersecurity Act, а також рекомендації ENISA. У США — закон Каліфорнії про безпеку IoT. Також стандарти безпеки IoT регламентуються такими міжнародними організаціями, як ISO/IEC, IEEE, ITU та NIST, що забезпечують технічну та правову основу для безпечного функціонування IoT-систем. В Україні ж, нормативно-правова база залишається фрагментарною та потребує комплексного вдосконалення для врахування специфіки IoT.

Зокрема, відсутність обов’язкових вимог до IoT-пристроїв щодо автентифікації, оновлень та шифрування вимагає розробки нових державних стандартів та законодавчих актів.

Загалом, безпека Інтернету речей вимагає багаторівневого підходу — від розробки безпечних пристроїв і протоколів до розбудови єдиної правової системи захисту. Лише за умови комплексного підходу, що включає технологічні інновації, державне регулювання та підвищення обізнаності користувачів, можливо досягти належного рівня безпеки в середовищі IoT.

РОЗДІЛ 2
ІНФРАСТРУКТУРНИЙ ЗАХИСТ ІОТ ТА АНАЛІЗ АТАК

2.1 Захист iot на рівні інфраструктури (IDS/IPS, VPN, FIREWALL)

Інфраструктурний захист IoT передбачає застосування комплексних заходів для захисту мережевих з'єднань, пристроїв та даних від кібератак. Серед основних технологій, що використовуються для цього, можна виділити системи виявлення та запобігання вторгненням (IDS/IPS), віртуальні приватні мережі (VPN) та міжмережеві екрани (Firewall). Ці методи спрямовані на моніторинг, аналіз і блокування потенційних загроз на різних рівнях інфраструктури IoT [17].

Системи виявлення вторгнень (IDS, Intrusion Detection System) та системи запобігання вторгненням (IPS, Intrusion Prevention System) є ключовими компонентами захисту IoT-інфраструктури. IDS аналізує трафік у реальному часі та повідомляє про виявлені аномалії, тоді як IPS додатково може автоматично блокувати підозрілі дії. IDS використовуються для пасивного моніторингу мережі та аналізу підозрілих активностей, тоді як IPS працює в активному режимі, запобігаючи атакам ще до їх реалізації. Основні функції IDS/IPS у захисті IoT включають:

- Моніторинг мережевого трафіку – перевірка вхідних та вихідних даних для виявлення можливих атак та підозрілої активності.
- Виявлення аномалій – ідентифікація відхилень у поведінці пристроїв IoT, що можуть свідчити про кібератаку або спроби експлуатації вразливостей.
- Блокування шкідливих дій – автоматичне реагування на спроби вторгнення (для IPS), включаючи ізоляцію заражених пристроїв та блокування шкідливого трафіку.
- Аналіз журналів безпеки – запис і аналіз подій безпеки для подальшого розслідування та вдосконалення механізмів захисту.
- Глибокий аналіз пакетів (DPI – Deep Packet Inspection) – перевірка вмісту переданих пакетів для виявлення атак на рівні застосунків.

- Використання сигнатурного та поведінкового аналізу – IDS/IPS можуть працювати на основі бази відомих атак (сигнатурний аналіз) або виявляти нові загрози за допомогою машинного навчання та поведінкового аналізу.

Використання IDS/IPS у IoT важливе, оскільки багато пристроїв мають обмежені можливості самостійного захисту, а їхні операційні системи часто не підтримують складні механізми безпеки. Крім того, IDS/IPS можуть бути інтегровані з рішеннями штучного інтелекту (ШІ), що дозволяє виявляти складні атаки та прогнозувати потенційні загрози за допомогою аналізу поведінки мережі.

Останні дослідження пропонують новий підхід до безпеки IoT – IoT Proxy, який поєднує VPN-термінатор та IPS з машинним навчанням. Такий підхід дозволяє виявляти пристрої через аналіз мережевого трафіку, проводити аутентифікацію без додаткових навантажень на пристрої з обмеженими ресурсами, а також застосовувати інтелектуальні алгоритми для автоматичної класифікації загроз та підвищення ефективності механізмів запобігання атакам.

VPN (Virtual Private Network) забезпечує безпечний канал зв’язку між IoT-пристроями та центральними серверами або іншими елементами мережі. Ця технологія створює зашифрований тунель для передавання даних, що запобігає їх перехопленню або зміні під час передавання. VPN дозволяє забезпечити безпечний доступ до віддалених IoT-пристроїв, приховати реальні IP-адреси та зменшити ризик несанкціонованого втручання у систему.

Основні переваги використання VPN у IoT:

- Шифрування трафіку – забезпечує захист конфіденційних даних від перехоплення та аналізу атакуючими за допомогою протоколів, таких як OpenVPN, IPsec та WireGuard.
- Приховування мережевих адрес – зменшує ризик атак на відкриті пристрої IoT, роблячи їх менш видимими для зловмисників.
- Захист від атак "людина посередині" (Man-in-the-Middle) – створення зашифрованого тунелю запобігає модифікації трафіку під час передачі між IoT-пристроями та серверами.

- Віддалене безпечне підключення – забезпечує можливість адміністраторам безпечно керувати IoT-інфраструктурою з будь-якого місця, незалежно від географічного розташування.
- Розширення функціоналу VPN за допомогою багаторівневої автентифікації (MFA) – дозволяє додатково захистити доступ до IoT-пристроїв, використовуючи методи багатофакторної автентифікації.
- Зменшення ризику атак DDoS – використання VPN може допомогти уникнути перевантаження пристроїв IoT за рахунок приховування їхніх реальних IP-адрес та перенаправлення трафіку через захищені шлюзи.
- Оптимізація маршрутизації трафіку – сучасні VPN-рішення підтримують функції балансування навантаження та вибір оптимального маршруту для покращення продуктивності IoT-мереж.

Останні розробки пропонують інтеграцію VPN з функціями IDS/IPS для створення безпечних шлюзів IoT. Такі шлюзи дозволяють централізовано керувати безпекою мережевих пристроїв, виконуючи перевірку аутентифікації, аналіз трафіку в реальному часі та автоматичне блокування загроз. Крім того, у сучасних VPN-рішеннях для IoT реалізуються механізми адаптивного шифрування, що дозволяють ефективно використовувати ресурси пристроїв із обмеженими обчислювальними можливостями.

Міжмережевий екран (Firewall) – це один із ключових механізмів безпеки, що контролює потік даних у мережі та забезпечує захист пристроїв IoT від зовнішніх загроз. Firewalls використовуються для захисту периферійних пристроїв, шлюзів, серверів і навіть окремих мережевих сегментів. Вони можуть бути реалізовані як апаратні, програмні або хмарні рішення.

Основні функції Firewall у захисті IoT:

- Фільтрація трафіку – аналіз і блокування несанкціонованих або потенційно небезпечних з'єднань на основі встановлених політик безпеки.
- Запобігання атакам DDoS – виявлення аномального трафіку та блокування масових атак, спрямованих на перевантаження пристроїв IoT.

- Обмеження доступу – застосування політик контролю доступу, які регулюють з'єднання між пристроями, мережами та зовнішніми ресурсами.
- Моніторинг мережевої активності – ведення журналів подій, запис активності мережі та аналіз трафіку для виявлення загроз.
- Глибокий аналіз пакетів (DPI – Deep Packet Inspection) – перевірка вмісту пакетів даних для виявлення потенційно шкідливих команд або нелегітимного трафіку.
- Виявлення аномалій – аналіз поведінкових моделей пристроїв IoT та ідентифікація відхилень, які можуть свідчити про атаки або компрометацію системи.
- Сегментація мережі – розділення IoT-інфраструктури на ізольовані підмережі для мінімізації ризиків поширення атак.
- Автоматичне оновлення правил безпеки – динамічне оновлення фільтрів та сигнатур загроз для реагування на нові кібератаки.

Сучасні Firewalls для IoT можуть поєднувати методи глибокого аналізу пакетів (DPI), штучного інтелекту та машинного навчання для підвищення ефективності виявлення загроз та адаптивного реагування на потенційні атаки. Завдяки цим технологіям міжмережеві екрани здатні ідентифікувати нові види атак та автоматично коригувати політики безпеки без втручання адміністратора

2.2 Безпека бездротових мереж iot

Безпека бездротових мереж IoT є критично важливим аспектом захисту пристроїв та інформації, що циркулює в таких мережах. Оскільки більшість IoT-пристроїв використовує бездротові технології, такі як Wi-Fi, Bluetooth, Zigbee, Z-Wave, LoRaWAN та 5G, вони стають вразливими до атак. Основні загрози включають перехоплення трафіку, атаки "людина посередині" (MITM), DoS/DDoS-атаки, компрометацію механізмів автентифікації, фізичний доступ до пристроїв та атаки на рівні радіочастотного сигналу. Щоб захистити бездротові IoT-мережі, необхідно застосовувати комплексні заходи безпеки, включаючи шифрування даних, посилену

автентифікацію, захист від атак на мережевий трафік, запобігання DoS-атакам та фізичний захист пристроїв [21].

Однією з важливих загроз є атаки на рівні радіочастот (RF). Такі атаки можуть включати глушіння сигналу (jamming), підміну сигналу (spoofing) та втручання в комунікаційні протоколи. Наприклад, зловмисники можуть генерувати сильні радіосигнали на тій самій частоті, що використовують IoT-пристрої, тим самим перешкоджаючи зв’язку між ними. Крім того, методи підміни сигналу дозволяють злочинцям змінювати передані команди або підключатися до систем, імітуючи справжні IoT-пристрої. Для захисту від таких атак використовуються технології частотного хопінгу (FHSS – Frequency Hopping Spread Spectrum), а також адаптивні алгоритми виявлення перешкод і автоматичного перемикання на інші канали зв’язку.

Для запобігання атакам MITM та перехоплення трафіку необхідно використовувати VPN, що дозволяє створювати захищений тунель між пристроями IoT та серверами. Крім того, використання безпечних протоколів зв’язку, таких як MQTT з TLS, забезпечує додатковий рівень захисту при передачі даних. Інтеграція систем виявлення вторгнень (IDS) та систем запобігання вторгненням (IPS) дозволяє своєчасно виявляти та блокувати MITM-атаки, аналізуючи мережевий трафік у режимі реального часу.

DoS/DDoS-атаки є однією з найбільш поширених загроз для бездротових IoT-мереж. Для запобігання таким атакам необхідно використовувати методи фільтрації трафіку на рівні мережевих шлюзів, що дозволяє блокувати підозрілий трафік ще на початковому етапі. Використання міжмережевих екранів (Firewall) забезпечує контроль доступу та запобігає проникненню шкідливого програмного забезпечення до IoT-мережі. Застосування алгоритмів машинного навчання для виявлення аномалій (AI-based anomaly detection) дозволяє виявляти нестандартні шаблони поведінки пристроїв та своєчасно реагувати на потенційні атаки.

Фізичний захист пристроїв також відіграє важливу роль у загальній безпеці бездротових IoT-мереж. Використання антивандальних корпусів дозволяє зменшити ймовірність фізичного втручання у пристрої. Розміщення IoT-пристроїв у контрольованих зонах ускладнює їх несанкціонований доступ. Додатково, апаратні

модулі безпеки, такі як TPM (Trusted Platform Module), дозволяють безпечно зберігати криптографічні ключі та інші важливі дані, що запобігає їх компрометації у випадку фізичного втручання.

Однією з основних загроз для бездротових IoT-мереж є атаки на рівні радіочастот (RF), які можуть призвести до втрати зв'язку, зміни даних або повного компрометування системи. До таких атак належать:

- Глушіння сигналу (Jamming) – зловмисник створює радіоперешкоди, які блокують передачу даних між IoT-пристроями, що може призвести до відмови у функціонуванні критичних систем.
- Підміна сигналу (Spoofing) – атакуючий підміняє легітимний сигнал, змушуючи пристрої обмінюватися фальшивими даними або змінювати поведінку.
- Атаки на рівні PHY/MAC – маніпуляції з рівнями фізичного доступу та управління мережевим середовищем можуть спричинити втрату конфіденційності переданих даних або втручання в механізми доступу до мережі.

Для захисту від таких атак застосовуються методи частотного хопінгу (FHSS – Frequency Hopping Spread Spectrum), адаптивного вибору каналів зв'язку, а також використання стандартів шифрування радіосигналів, таких як AES у Zigbee та WPA3 у Wi-Fi.

Роль гомоморфного шифрування у захисті IoT.

Гомоморфне шифрування – це перспективний метод захисту даних, який дозволяє виконувати обчислення над зашифрованою інформацією без необхідності її розшифрування. Це означає, що навіть у разі перехоплення трафіку або компрометації серверів, зловмисник не зможе отримати доступ до вихідних даних. Гомоморфне шифрування особливо важливе для IoT, оскільки воно:

- Гарантує конфіденційність обчислень – навіть провайдери хмарних послуг не можуть бачити реальні дані користувача.
- Запобігає компрометації пристроїв – навіть якщо хакери отримають доступ до зашифрованих даних, вони залишаться нерозшифрованими.

- Дозволяє безпечний аналіз даних – це особливо важливо для IoT-пристроїв, що обробляють персональні дані, наприклад, у системах охорони здоров'я або фінансових операціях.

Гомоморфне шифрування особливо корисне для IoT, оскільки забезпечує конфіденційність даних навіть у хмарних середовищах та під час обміну між пристроями. Попри те, що традиційне шифрування TLS/SSL добре захищає дані під час передачі, воно не захищає їх від компрометації на рівні серверів або обробки. Гомоморфні методи вирішують цю проблему, дозволяючи безпечний аналіз і маніпуляцію з даними навіть у незахищеному середовищі.

Основним викликом для впровадження гомоморфного шифрування є його висока обчислювальна складність, що може ускладнити його використання на пристроях IoT з обмеженими ресурсами. Однак із розвитком оптимізованих криптографічних алгоритмів цей підхід може стати стандартом для IoT-безпеки в майбутньому.

Періферійні обчислення (Edge Computing) є важливим елементом архітектури IoT, який дозволяє виконувати обробку даних ближче до місця їхнього створення, зменшуючи затримки та ризики передачі через глобальні мережі.

Оскільки багато IoT-пристроїв працюють у віддалених або розподілених середовищах, передача всіх даних у хмару може створювати ризики з точки зору безпеки та продуктивності. Використання периферійних обчислень дозволяє виконувати обробку даних ближче до джерела їх генерації, зменшуючи ризик перехоплення або модифікації інформації під час передачі. Крім того, це дає змогу швидше реагувати на загрози та застосовувати механізми безпеки безпосередньо на рівні шлюзів або локальних серверів. Edge Computing також дозволяє застосовувати моделі машинного навчання для аналізу поведінкових аномалій у мережевому трафіку та своєчасного виявлення атак.

Проте безпека IoT-пристроїв у контексті Edge Computing має свої особливості:

- Захист передачі даних між периферійними вузлами – використання TLS/SSL та VPN для запобігання MITM-атакам.

- Механізми довіреної ідентифікації пристроїв – застосування сертифікатної автентифікації (PKI) та безпечного обміну ключами.
- Децентралізовані методи виявлення загроз – периферійні вузли можуть виконувати моніторинг трафіку та аналізувати поведінкові аномалії для виявлення потенційних атак.

Edge Computing забезпечує додатковий рівень захисту IoT, оскільки дозволяє обмежити передачу конфіденційних даних у хмару, а також забезпечує локальну реакцію на загрози без необхідності звернення до централізованих серверів.

Перспективи розвитку безпеки бездротових IoT-мереж передбачають впровадження новітніх технологій, таких як 5G, що забезпечить додаткові рівні безпеки та підвищить продуктивність IoT-мереж. Використання штучного інтелекту (AI) для аналізу загроз дозволить автоматизувати процеси виявлення потенційних атак, а квантове шифрування стане наступним кроком у захисті даних від новітніх загроз. Крім того, концепція Zero Trust Architecture (ZTA) передбачає відмову від довіри навіть до внутрішніх пристроїв мережі, що дозволяє мінімізувати ризики компрометації.

Захист бездротових мереж IoT є складним завданням, яке потребує багаторівневого підходу. Окрім стандартних заходів, таких як шифрування та автентифікація, особливу увагу слід приділити захисту радіочастотних сигналів, використанню гомоморфного шифрування для конфіденційності даних та безпеці периферійних обчислень. Поєднання цих технологій дозволить створити більш стійкі до атак IoT-системи, здатні функціонувати у складних умовах сучасного кіберсередовища.

2.3 Аналіз відомих атак на iot-пристрої

Як згадувалося у попередньому розділ інтернет речей (IoT) став однією з головних цілей для кіберзлочинців через велику кількість вразливих пристроїв, що часто мають слабкий рівень безпеки. Зловмисники використовують IoT-пристрої для

організації ботнетів, атак на критичну інфраструктуру, шпигунства та крадіжки конфіденційної інформації. У даному розділі розглянуто більш детально найвідоміші атаки на IoT-пристрої, включаючи Botnet Mirai, VPNFilter, BrickerBot та інші [19].

Mirai – один із найвідоміших ботнетів, що вперше з'явився у 2016 році. Його основною метою були IoT-пристрої, такі як маршрутизатори, IP-камери та DVR-системи, які використовувалися для організації масованих DDoS-атак. Mirai став відомим завдяки ефективним методам зараження та масштабному впливу на інтернет-інфраструктуру. Mirai також має функції самозахисту, видаляючи конкуруючі шкідливі програми з інфікованих пристроїв.

Технічний аналіз Mirai

Mirai функціонував у кілька основних етапів. Спочатку він здійснював активне сканування Інтернету у пошуках IoT-пристроїв, доступних через Telnet або SSH. Він використовував вбудований список понад 60 пар "логін-пароль", характерних для заводських налаштувань пристроїв. Після успішного входу Mirai інсталював шкідливий код, що перетворював пристрій на частину ботнету.

Однією з ключових особливостей Mirai було постійне сканування нових жертв – кожен інфікований пристрій автоматично шукав інші IoT-пристрої, що мають слабкі паролі. Це дозволяло ботнету експоненціально розширювати свою інфраструктуру.

Інфіковані пристрої використовувалися для масованих DDoS-атак, що включали такі методи:

- UDP Flood – відправлення великої кількості UDP-пакетів на випадкові порти цільового сервера для виснаження його ресурсів.
- TCP SYN Flood – створення величезної кількості напіввідкритих TCP-з'єднань, що призводило до перевантаження серверів.
- ACK Flood – надсилання великої кількості TCP ACK-пакетів для споживання мережевих ресурсів жертви.
- HTTP Flood – генерація масових запитів до веб-ресурсів для вичерпання їх потужностей.

Mirai мав механізм самознищення – після перезавантаження пристрою шкідливий код видалявся з пам'яті, однак пристрій залишався вразливим до

повторного зараження через відсутність виправлення проблеми слабких облікових даних [24].

У 2016 році Mirai використали для однієї з наймасштабніших атак на провайдера Dyn DNS, що спричинило відключення таких великих сервісів, як Twitter, Reddit, Netflix та GitHub. Потужність атаки перевищувала 1 Тбіт/с, що стало однією з найпотужніших DDoS-атак в історії.

Код Mirai був опублікований у відкритому доступі, що призвело до появи численних його модифікацій, таких як Hajime, Okiru та Satori. Це зробило Mirai основою для багатьох нових ботнетів, які продовжують атакувати IoT-пристрої по всьому світу.

Методи захисту від Mirai

Щоб уникнути зараження ботнетом Mirai, необхідно дотримуватись таких рекомендацій:

- Змінювати стандартні облікові дані на унікальні складні паролі.
- Вимикати Telnet та SSH на IoT-пристроях, якщо вони не використовуються.
- Регулярно оновлювати прошивку пристроїв для виправлення вразливостей.
- Використовувати брандмауери та системи виявлення вторгнень (IDS/IPS) для моніторингу підозрілої активності.

Mirai став попередженням для всієї індустрії IoT, продемонструвавши, наскільки небезпечними можуть бути пристрої з неналежним рівнем захисту. Надалі для запобігання подібним атакам необхідно впроваджувати більш надійні механізми автентифікації, сегментувати мережу та використовувати сучасні технології виявлення загроз.Його основною метою були IoT-пристрої, такі як маршрутизатори, IP-камери та DVR-системи, які використовувалися для організації масованих DDoS-атак. Mirai став відомим завдяки ефективним методам зараження та масштабному впливу на інтернет-інфраструктуру.

Mirai діяв у кілька етапів. Спочатку він здійснював активне сканування Інтернету у пошуках IoT-пристроїв, доступних через Telnet або SSH. Далі ботнет проводив брутфорс-атаки, використовуючи список стандартних логінів і паролів, які часто залишалися незмінними у заводських налаштуваннях пристроїв. Після

отримання доступу Mirai інсталював шкідливий код, який перетворював пристрій на частину ботнету. Інфіковані пристрої використовувалися для масованих DDoS-атак, під час яких вони надсилали величезні обсяги мережевого трафіку до цільових серверів, спричиняючи їх перевантаження і вихід з ладу.

Mirai відрізнявся низкою унікальних особливостей. По-перше, його код був оприлюднений у відкритому доступі, що дозволило кіберзлочинцям створювати його модифікації, такі як Hajime та Okiru. По-друге, ботнет підтримував різні методи DDoS-атак, включаючи UDP Flood, TCP SYN Flood, ACK Flood та HTTP Flood. Крім того, Mirai мав механізм самознищення – після перезавантаження пристрою шкідливий код видалявся з пам’яті, що значно ускладнювало його виявлення та аналіз.

У 2016 році Mirai використали для однієї з наймасштабніших атак на провайдера Dyn DNS, що призвело до відключення великих онлайн-сервісів, таких як Twitter, Reddit та Netflix. Це показало критичні проблеми у безпеці IoT-пристроїв та необхідність їхнього захисту від подібних загроз.

VPNFilter – це складне багатоступеневе шкідливе програмне забезпечення, виявлене у 2018 році, що атакувало маршрутизатори та мережеві пристрої. Воно було розроблене з можливістю довготривалого зараження, збору конфіденційної інформації та навіть фізичного виведення пристроїв з ладу.

VPNFilter функціонував у три етапи. На першому етапі він проникав у пристрій через відомі вразливості або використовуючи слабкі паролі. Після зараження VPNFilter записувався у файлову систему, щоб уникнути видалення після перезавантаження. Другий етап включав встановлення зв’язку із серверами управління (C2) для отримання додаткових модулів. Ці модулі давали змогу шкідливому ПЗ виконувати складні атаки, зокрема, перехоплення трафіку, модифікацію DNS-запитів, запис натискання клавіш та віддалене виконання команд. На фінальному етапі VPNFilter активував свою функцію "Kill Switch", яка дозволяла перезаписати системні файли пристрою, роблячи його повністю нефункціональним.

Особливістю VPNFilter була його висока стійкість до виявлення та видалення. Код залишався активним навіть після перезавантаження пристрою, що відрізняло

його від інших ботнетів, таких як Mirai. Завдяки модульній структурі він міг динамічно змінювати свої можливості, додаючи нові функції без потреби оновлення всього шкідливого коду. Важливо відзначити, що VPNFilter не лише використовувався для компрометації звичайних користувачів, а й активно застосовувався для атак на промислові мережі та урядові установи. Один із його модулів дозволяв зловмисникам збирати мережевий трафік, включаючи облікові дані користувачів, банківську інформацію та інші конфіденційні дані.

VPNFilter став серйозною загрозою для корпоративних мереж і домашніх користувачів через свою складну архітектуру та можливості приховування слідів. Він використовував зашифровані канали зв'язку для комунікації із C2-серверами, що ускладнювало його виявлення та аналіз. Крім того, зловмисники могли запускати на заражених пристроях специфічні сценарії атак, включаючи віддалене шпигунство, перенаправлення трафіку на підроблені веб-сайти або навіть повне фізичне знищення пристрою.

Атака VPNFilter продемонструвала новий рівень загроз у сфері IoT, оскільки була націлена не лише на звичайних користувачів, а й на критичну інфраструктуру. Її складність та багаторівневість змусили дослідників кібербезпеки переглянути підходи до захисту IoT-пристроїв, підкреслюючи необхідність регулярного оновлення прошивки, використання надійної автентифікації та впровадження розширених систем моніторингу загроз.

BrickerBot – це унікальне шкідливе програмне забезпечення, яке працювало за принципом permanent denial-of-service (PDoS), що відрізняється від традиційних DDoS-атак тим, що фізично виводило IoT-пристрої з ладу. Замість використання інфікованих пристроїв для генерації трафіку ботнет BrickerBot фактично знищував їх, змушуючи власників проводити відновлення або повністю замінювати пристрої.

BrickerBot працював у кілька етапів. Спочатку він здійснював масове сканування Інтернету в пошуках уразливих IoT-пристроїв, зокрема тих, що мали відкритий Telnet або використовували слабкі паролі. Після успішного підключення ботнет застосовував техніку грубої сили для отримання доступу, після чого виконував набір шкідливих команд. Він перезаписував прошивку пристрою, знищував критичні

файли операційної системи та блокував будь-яку можливість відновлення через стандартні процедури. Крім того, BrickerBot міг змінювати таблиці розділів файлової системи, форматувати диски або записувати випадкові дані в пам’ять пристрою, що робило його повністю нефункціональним.

Головною особливістю BrickerBot є те, що він не намагався отримати фінансову вигоду або організувати керований ботнет. Натомість він діяв як своєрідний "кібермесник", який знищував небезпечні або неправильно сконфігуровані пристрої, які могли бути використані в інших ботнетах, таких як Mirai. Деякі дослідники припускають, що BrickerBot створив хакер або група хакерів, які прагнули очистити Інтернет від небезпечних пристроїв. Проте його дія призвела до значних збитків, оскільки багато пристроїв вийшли з ладу без можливості відновлення.

BrickerBot ілюструє серйозні загрози, пов’язані з низьким рівнем безпеки IoT-пристроїв. Найефективнішим способом захисту від таких атак є використання надійних паролів, закриття відкритих портів, відключення Telnet та SSH, якщо вони не використовуються, а також регулярне оновлення прошивки та програмного забезпечення IoT-пристроїв. Крім того, сегментування мережі та застосування міжмережевих екранів (firewall) може значно зменшити ризик компрометації пристроїв BrickerBot або іншими шкідливими програмами.

Persirai – шкідливе програмне забезпечення, що атакує IP-камери, використовуючи їхні вразливості для отримання несанкціонованого доступу. Цей ботнет є варіацією відомих IoT-загроз, зосереджених на пристроях відеоспостереження, які часто мають відкриті порти та використовують стандартні облікові дані. Persirai є складним шкідливим програмним забезпеченням, яке комбінує елементи класичних ботнетів з методами прихованого впровадження та експлуатації уразливостей прошивки.

Persirai використовує кілька технік для захоплення контролю над IP-камерами:

- Експлуатація вразливостей прошивки – Persirai атакує пристрої, що мають відомі недоліки в програмному забезпеченні, зокрема проблеми з автентифікацією та некоректним обробленням HTTP-запитів.

- Використання вбудованих облікових даних – багато IP-камер мають стандартні логіни та паролі, які часто не змінюються користувачами. Persirai використовує попередньо визначений список облікових даних для отримання доступу до адміністративної панелі пристрою.
- Автоматичне зараження сусідніх пристроїв – після інфікування однієї камери шкідливе ПЗ сканує локальну мережу в пошуках інших уразливих пристроїв і намагається захопити їх аналогічним способом.
- Організація DDoS-атак – Persirai використовує заражені пристрої для виконання масштабних атак типу UDP Flood або HTTP Flood, що можуть бути спрямовані на державні установи, підприємства або конкуренти замовників атаки.
- Шпигунство та витік даних – через контроль над камерами зловмисники можуть отримувати відеопотоки, що дає можливість спостереження за об’єктами в реальному часі.

На відміну від класичних ботнетів, Persirai активно уникає виявлення та блокує доступ адміністратора до налаштувань камери, видаляючи лог-файли та змінюючи системні параметри. Це ускладнює процес виявлення та очищення інфікованого пристрою. Також цей ботнет відзначається високою швидкістю поширення, оскільки інфіковані пристрої автоматично шукають нові жертви.

Щоб уникнути зараження Persirai, необхідно:

- Змінити стандартні паролі на складні та унікальні.
- Оновлювати прошивку пристроїв до останніх версій, що містять виправлення вразливостей.
- Закрити невикористовувані порти та використовувати брандмауер для обмеження зовнішнього доступу.
- Використовувати системи виявлення вторгнень (IDS) для моніторингу аномальної активності в мережі.

Persirai став серйозною загрозою для систем відеоспостереження, показавши, що навіть спеціалізовані IoT-пристрої можуть бути використані зловмисниками для організації атак і збору конфіденційних даних. Подальший розвиток технологій кібербезпеки необхідний для зменшення ризиків, пов’язаних із такими ботнетами.

Аналіз відомих атак на IoT-пристрої показує, що основними векторами компрометації є слабкі паролі, відкриті порти, відсутність оновлень прошивки та використання небезпечних сервісів. Ці фактори значно спрощують зловмисникам проникнення у пристрої, дозволяючи їм створювати ботнети, викрадати конфіденційні дані або навіть фізично виводити пристрої з ладу.

Загальні тенденції атак свідчать про те, що більшість сучасних загроз використовує автоматизоване сканування Інтернету для пошуку вразливих пристроїв, що не мають належного захисту. Mirai та Persirai демонструють цей підхід, здійснюючи пошук IP-камер, маршрутизаторів та DVR-систем, які використовують стандартні паролі. Інші атаки, такі як VPNFilter, експлуатують слабкі механізми автентифікації та застарілі прошивки, щоб встановити контроль над пристроєм і виконати зловмисні дії, включаючи шпигунство та знищення даних. Особливо небезпечною є техніка permanent denial-of-service (PDoS), яку використовував BrickerBot, оскільки вона не лише тимчасово виводить пристрій із ладу, а й робить його повністю непридатним до використання.

Протидія таким загрозам вимагає впровадження багаторівневого підходу до безпеки. Першочергово необхідно змінювати стандартні облікові дані на унікальні складні паролі, а також вимикати невикористовувані сервіси, такі як Telnet, FTP і HTTP-доступ. Важливим аспектом є регулярне оновлення прошивки IoT-пристроїв, що дозволяє виправляти відомі вразливості та захищати пристрої від експлуатації шкідливого програмного забезпечення. Крім того, використання сучасних засобів безпеки, таких як IDS/IPS та VPN, сприяє виявленню підозрілої активності в мережі та запобігає компрометації пристроїв. Сегментування мережі також є критично важливим заходом, оскільки дозволяє ізолювати IoT-пристрої від основних корпоративних або домашніх мереж, зменшуючи ризик розповсюдження атак [20].

Подальший розвиток IoT-технологій повинен супроводжуватися зростаючою увагою до питань кібербезпеки. Виробники пристроїв повинні впроваджувати принцип Security by Design, що передбачає забезпечення безпеки на рівні архітектури та розробки пристроїв. Водночас користувачі та компанії мають слідувати найкращим

практикам у сфері безпеки, щоб мінімізувати ризики компрометації IoT-інфраструктури. Без відповідного рівня захисту IoT залишатиметься однією з найбільш уразливих технологічних екосистем сучасного цифрового світу.

2.4 Виявлення вразливостей та оцінка ефективності методів захисту

Забезпечення безпеки Інтернету речей (IoT) передбачає не тільки впровадження захисних механізмів, а й постійний моніторинг наявних загроз і вразливостей. Виявлення слабких місць в IoT-інфраструктурі дозволяє зменшити ризики атак, підвищити стійкість системи та оцінити ефективність застосованих методів захисту. Оскільки IoT-пристрої часто працюють у відкритому середовищі, мають обмежені обчислювальні ресурси та використовують стандартизовані протоколи зв’язку, вони стають привабливою ціллю для кіберзлочинців [21].

Вразливості в IoT можна класифікувати за кількома основними категоріями. Фізичні вразливості є однією з найсерйозніших загроз, оскільки багато IoT-пристроїв розташовані в місцях, де зловмисники можуть отримати до них безпосередній доступ. Це може дозволити їм змінювати прошивку, вилучати конфіденційні дані або навіть замінювати пристрої підробленими аналогами. Відсутність фізичного захисту, такого як надійні корпуси, датчики несанкціонованого доступу або шифрування даних у пам’яті, робить такі пристрої вразливими до атак.

Мережева безпека IoT також є важливим аспектом, оскільки багато пристроїв використовують стандартні комунікаційні протоколи, які можуть бути неправильно налаштовані або не підтримують сучасні методи шифрування. Наприклад, використання MQTT без TLS дозволяє перехоплювати та змінювати дані, передані між пристроями. Також відкриті порти, такі як 23 (Telnet) або 5555 (ADB), можуть стати вхідною точкою для атакуючих, дозволяючи їм отримати повний контроль над пристроєм.

Програмні вразливості є ще однією критичною проблемою. Недостатнє тестування безпеки програмного забезпечення IoT-пристроїв може призвести до наявності уразливостей, таких як переповнення буфера, SQL-ін'єкції або виконання

довільного коду. Використання застарілих бібліотек або компонентів з відкритим кодом без відповідного оновлення збільшує ймовірність атак.

Автентифікація та управління доступом також є слабким місцем у багатьох IoT-системах. Використання заводських паролів, незахищених механізмів входу або відсутність багатофакторної автентифікації роблять пристрої легкими мішенями для атак грубою силою. У багатьох випадках зловмисники можуть отримати контроль над пристроєм просто тому, що користувачі не змінили стандартні облікові дані.

Небезпечне оновлення програмного забезпечення є однією з ключових загроз для IoT-пристроїв. Відсутність механізмів перевірки автентичності оновлень може призвести до їх компрометації. Якщо пристрій завантажує оновлення без перевірки цифрового підпису або через незахищений канал зв’язку, зловмисники можуть виконати атаку «людина посередині» (MITM) та підмінити файл оновлення [22].

Це дозволяє впровадити шкідливий код, що може використовуватися для викрадення даних, віддаленого контролю або включення пристрою до ботнету. Крім того, відсутність механізмів перевірки цілісності файлів оновлення збільшує ризик завантаження некоректного або зараженого програмного забезпечення.

Пристрої можуть бути легко доступними для зловмисників, що дозволяє їм здійснювати фізичний доступ, змінювати прошивку, вилучати дані або навіть замінювати пристрої підробленими аналогами. Також існує загроза мережевої безпеки, оскільки відкриті порти, відсутність шифрування трафіку та неправильна конфігурація протоколів зв’язку (наприклад, MQTT або CoAP) можуть призвести до перехоплення даних або атак MITM.

Додаткову небезпеку становлять програмні вразливості, такі як помилки у вихідному коді, бекдори або використання застарілих бібліотек, що відкривають доступ до пристрою для атакуючих. Недостатній контроль автентифікації та доступу, включаючи використання слабких паролів, заводських облікових записів або відсутність багатофакторної автентифікації, дозволяє зловмисникам отримати контроль над пристроєм.

Оптимальні заходи захисту включають використання шифрування під час передавання оновлень, впровадження механізмів перевірки цифрового підпису, а також захист пристрою від можливих атак під час процесу оновлення.

Для виявлення загроз у IoT-середовищі використовують такі методи:

1. Сканування вразливостей – автоматизовані інструменти, такі як Shodan, Nmap, Nessus або OpenVAS, здійснюють аналіз відкритих портів, версій прошивок та потенційних загроз. Наприклад, за допомогою Nmap можна виконати команду що зображена на рис.2.1.

```
nmap -sV -p 1-65535 --script vuln <IP-адреса>
```

Рис. 2.1 Команда для сканування вразливостей

Ця команда дозволяє просканувати всі відкриті порти та виявити відомі вразливості пристрою.

2. Аналіз мережевого трафіку – використання інструментів Wireshark, Zeek або Snort для виявлення підозрілих з’єднань, аномальної активності або витоків даних. Наприклад, Wireshark дозволяє фільтрувати пакети (рис.2.2).

```
tcp.port == 1883 && mqtt
```

Рис. 2.2 Команда для фільтрації пакетів

Це допомагає відстежити незашифровані MQTT-з'єднання, які можуть бути використані для атак.

3. Тестування на проникнення (Penetration Testing) – активні перевірки IoT-пристроїв на стійкість до атак. Застосовуються Metasploit, Burp Suite та IoT Exploit Toolkit для експлуатації відомих вразливостей та оцінки рівня захисту системи.

4. Фаззинг (Fuzz Testing) – метод, що передбачає введення випадкових або спотворених даних у комунікаційні протоколи або API для виявлення можливих помилок та уразливостей. Використовуються такі інструменти, як Peach Fuzzer та Boofuzz.

5. Моделювання атак (Threat Modeling) – включає створення можливих сценаріїв атак та їхньої імітації з використанням інструментів типу MITRE ATT&CK for IoT.

6. Аналіз поведінки пристроїв (Behavioral Analysis) – застосування технологій машинного навчання для відстеження аномалій у роботі IoT-пристроїв. Наприклад, якщо звичайний пристрій починає передавати великі обсяги даних у незвичний час, це може свідчити про компрометацію.

Оцінка ефективності заходів кібербезпеки в IoT виконується за допомогою комплексного підходу, який включає кілька рівнів контролю та аналізу.

1. Тестування на проникнення (Penetration Testing) – регулярне проведення перевірок безпеки IoT-пристроїв для виявлення потенційних загроз. Використовуються такі інструменти, як Metasploit, Burp Suite та OWASP ZAP, які дозволяють імітувати атаки на мережеві сервіси та API пристроїв. Наприклад, тестування автентифікаційних механізмів може включати спроби брутфорсу або експлуатації вразливостей у протоколах, таких як MQTT або CoAP.

2. Моніторинг подій безпеки (Security Event Monitoring) – застосування SIEM-систем (Security Information and Event Management), які збирають журнали подій, аналізують мережевий трафік та відстежують можливі аномалії. Такі системи, як Splunk, IBM QRadar та ELK Stack, допомагають виявляти несанкціоновані підключення, підозрілу активність та спроби злому паролів.

3. Порівняння з безпековими стандартами та аудит безпеки – оцінка відповідності впроваджених заходів міжнародним стандартам, таким як ISO/IEC 27001, NIST SP 800-183, OWASP IoT Top 10, IEC 62443. Це дозволяє визначити прогалини в безпеці та розробити рекомендації щодо їх усунення.

4. Автоматизоване виявлення аномалій – використання технологій штучного інтелекту та машинного навчання для аналізу поведінки пристроїв у режимі реального часу. Наприклад, системи на основі TensorFlow або Azure AI можуть виявляти відхилення в шаблонах передавання даних, що може вказувати на спробу атаки. Використання нейронних мереж дозволяє прогнозувати ймовірні загрози на основі історичних даних.

5. Zero Trust-архітектура – застосування концепції нульової довіри, що передбачає перевірку кожного підключеного пристрою та користувача перед наданням доступу до мережі. Реалізація Zero Trust включає многофакторну автентифікацію (MFA), мікросегментацію мережі та політики найменших привілеїв (Least Privilege Policy). Інструменти, як Google BeyondCorp та Microsoft Azure AD Conditional Access, допомагають впровадити цей підхід у великих IoT-екосистемах.

Вразливості можуть бути спричинені як фізичним доступом до пристроїв, так і мережею, програмним забезпеченням чи неправильною конфігурацією [25]. Найкращі результати дає комбінований підхід, що включає автоматизоване сканування, аналіз мережевого трафіку, тестування на проникнення, фаззинг та моніторинг поведінки пристроїв. Оцінка ефективності заходів безпеки має проводитися регулярно, з урахуванням відповідності стандартам безпеки та впровадження сучасних механізмів захисту, таких як Zero Trust-архітектура та AI-моделі аналізу загроз. У майбутньому розвиток технологій кібербезпеки для IoT має зосередитися на вдосконаленні методів автоматичного виявлення та нейтралізації загроз, що дозволить значно зменшити ризики атак на IoT-інфраструктуру.

2.5 Висновки за другим розділом

У другому розділі було розглянуто основні напрямки інфраструктурного захисту IoT, безпеки бездротових мереж, технічний аналіз реальних атак на IoT-пристрої, а також методи виявлення вразливостей та оцінки ефективності засобів захисту.

Перш за все, встановлено, що інфраструктурний захист IoT є фундаментом для забезпечення надійної безпеки всієї екосистеми. Інструменти IDS/IPS, VPN та міжмережеві екрани дозволяють здійснювати моніторинг, блокування та фільтрацію трафіку, виявляти підозрілу активність, а також запобігати несанкціонованому доступу до пристроїв. Впровадження сучасних рішень на основі машинного навчання

(ML) та інтелектуального аналізу поведінки трафіку значно підвищує ефективність виявлення складних атак.

Було детально проаналізовано проблеми захисту бездротових IoT-мереж, які є особливо вразливими через використання відкритих протоколів і радіочастотних технологій. Дослідження показало, що атаки на рівні PHY та MAC, включаючи глушіння, підміну сигналу (spoofing), MITM та DoS, становлять серйозну загрозу для стабільної роботи пристроїв. Використання таких технологій, як FHSS, WPA3, TLS/SSL, а також гомоморфне шифрування, сприяє зміцненню безпеки бездротового середовища.

У підрозділі 2.3 було проведено технічний аналіз відомих атак, зокрема ботнетів Mirai, VPNFilter, BrickerBot та Persirai. Ці приклади продемонстрували вразливості, пов’язані з незмінними заводськими паролями, відсутністю оновлень, відкритими портами та слабкими механізмами автентифікації. Mirai та Persirai ілюструють швидке масштабування загроз за рахунок автоматичного сканування вразливих пристроїв, тоді як VPNFilter і BrickerBot продемонстрували більш складну багатофазову архітектуру атак, включаючи фізичне знищення пристроїв або шпигування. Це підтверджує, що IoT-середовище залишається привабливим для зловмисників, а загрози постійно еволюціонують.

Останній підрозділ зосереджувався на методах виявлення вразливостей та оцінці ефективності захисту, зокрема скануванні портів, фаззингу, поведінковому аналізі, використанні SIEM-систем і впровадженні Zero Trust-архітектури. Було відзначено, що найкращі результати дає багаторівневий підхід, що поєднує автоматизовані та аналітичні методи виявлення загроз. Важливою є регулярна перевірка відповідності міжнародним стандартам безпеки (ISO, NIST, OWASP), а також впровадження систем на основі штучного інтелекту для виявлення аномалій.

Таким чином, ефективна безпека IoT вимагає системного, адаптивного і проактивного підходу, що охоплює як інфраструктурні, так і прикладні рівні.

РОЗДІЛ 3
ПРАКТИЧНА РЕАЛІЗАЦІЯ МЕТОДІВ УПРАВЛІННЯ ЗАХИСТОМ ІНФОРМАЦІЇ В ІОТ НА ПРИКЛАДІ ESP8266

3.1 Опис IoT-пристрою

У рамках практичної частини дослідження було обрано прототип IoT-пристрою, що ілюструє основні принципи функціонування та захисту інформації в середовищі Інтернету речей. В якості прикладної системи було обрано автоматизовану систему годування тварин, що дозволяє моделювати типові сценарії взаємодії, обміну даними, а також потенційні вектори атак у домашньому або малопромисловому IoT-середовищі.

Цей пристрій побудовано на базі мікроконтролера ESP8266 NodeMCU (рис. 3.1), який забезпечує бездротову комунікацію, управління сервомеханізмами та обробку вхідних HTTP-запитів через локальну Wi-Fi точку доступу. Реалізована система дозволяє оцінити ефективність стандартних методів захисту на прикладі типової архітектури IoT-пристрою.

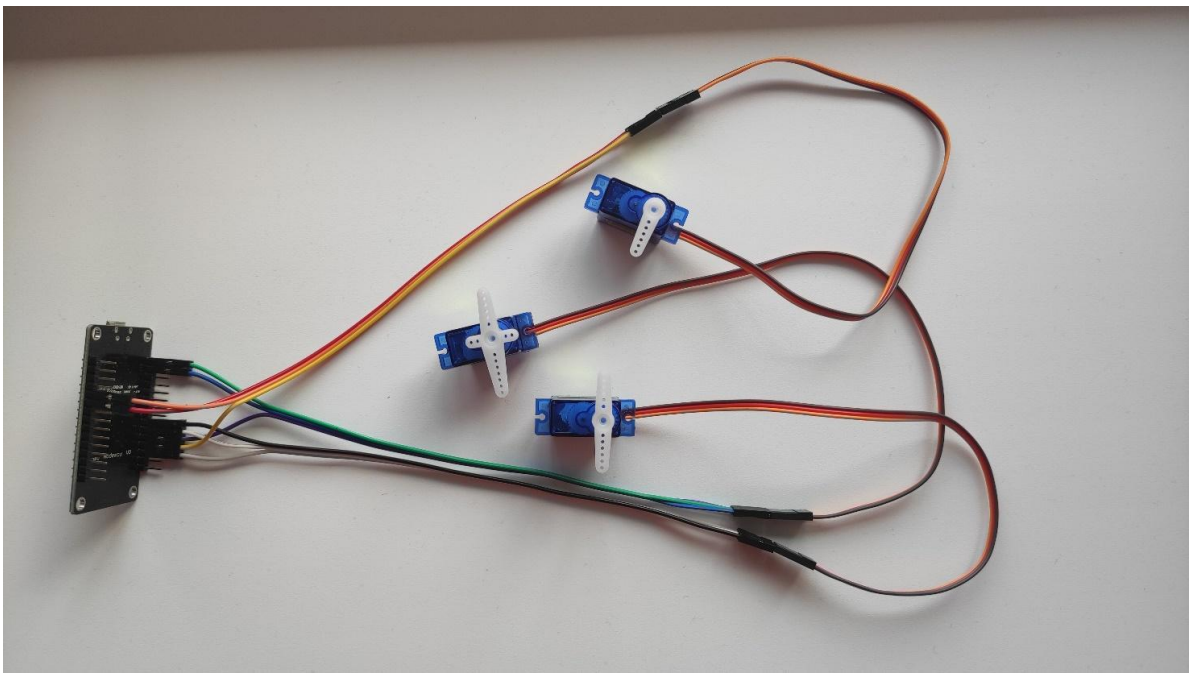

Рис. 3.1 Апаратна частина системи

Основне призначення:

- Автоматизоване або ручне годування домашніх тварин.
- Локальне керування через Wi-Fi з можливістю надсилання команд у вигляді HTTP-запитів.
- Демонстрація потенційних уразливостей IoT-пристроїв та способів їх нейтралізації.

Апаратна конфігурація:

- Мікроконтролер**:** ESP8266 NodeMCU з вбудованим Wi-Fi модулем.
- Сервомеханізми**:** 3 × SG90 (рис. 3.2) для відкриття люків із кормом.
- Індикація стану**:** світлодіод, підключений до вбудованого GPIO-порту.
- Wi-Fi точка доступу**.**

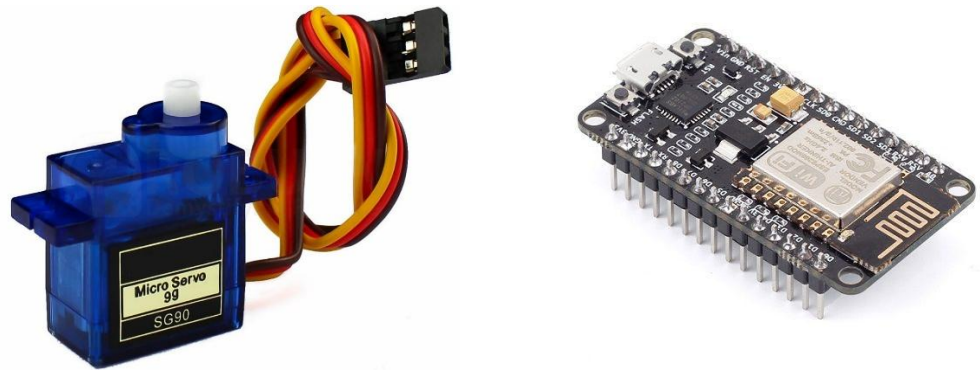

Рис. 3.2 Сервомеханізм SG90 та ESP8266 NodeMCU

Основна логіка роботи:

1. Створення точки доступу Wi-Fi — пристрій ініціалізує власну мережу, дозволяючи користувачам підключатись напряму.

2. Ініціалізація веб-сервера на порту 80, який приймає та обробляє HTTP-запити.

3. Виконання дій відповідно до запитів, зокрема:

   - Відкриття/закриття люків
   - Вмикання світлодіода
   - Активація автоматичного режиму годування

Обробка запиту зображена на рис. 3.3.

```
if (request.indexOf("OPEN") > 0) {
    LEDmig2(LED_BUILTIN, 3);   // Блимання LED
    Open();                    // Відкриття люка
    Close();                   // Закриття люка
    client.println("HTTP/1.1 200 OK\r\n");
    client.println("OPEN");
    client.flush();
}
```

Рис. 3.3 Обробка запиту

Попри корисність та базову функціональність IoT-пристроїв, зокрема тих, що побудовані на мікроконтролерах ESP8266, важливо розуміти, що у своїй початковій реалізації вони зазвичай не містять належних засобів забезпечення інформаційної безпеки. У рамках даної роботи було проаналізовано пристрій автоматизованого годування тварин, який керується через Wi-Fi з використанням веб-сервера на ESP8266. На основі вихідного коду, виявлено низку критичних вразливостей, які можуть бути використані зловмисниками для компрометації пристрою [26].

По-перше, повністю відсутній механізм автентифікації користувача. Це означає, що будь-який суб'єкт, який підключається до Wi-Fi точки доступу, створеної ESP8266, автоматично отримує можливість взаємодіяти з сервером та виконувати HTTP-запити. Таким чином, пристрій не здатен розрізнити авторизованого користувача від потенційного зловмисника. Це створює реальний ризик несанкціонованого керування, наприклад — випадкове або навмисне відкриття люка з кормом у невідповідний час.

По-друге, комунікація між клієнтом і пристроєм здійснюється через відкритий HTTP-протокол без використання будь-якого шифрування. Це робить весь трафік, включаючи команди керування, вразливим до перехоплення. Зловмисник, який підключився до тієї ж Wi-Fi мережі, може легко використати техніки мережевого моніторингу (наприклад, sniffer Wireshark) для збору даних про запити, які надсилаються на пристрій. Надалі він зможе відтворити ці запити або змінити їх, імітуючи дії легітимного користувача.

Третьою проблемою є відсутність контролю доступу та обмеження частоти запитів. Сервер ESP8266 обробляє всі запити, які надходять, незалежно від їх кількості, часу надсилання чи джерела. Такий підхід відкриває можливість для атак типу brute-force (грубий перебір) або flooding (перевантаження системи запитами). У випадку, якщо хтось буде постійно надсилати команди (наприклад, /OPEN, /CLOSE тощо), це може призвести до дестабілізації пристрою, підвищеного зносу апаратних компонентів (зокрема сервомоторів), а також до повної недоступності пристрою для легітимного користувача.

Крім того, така архітектура не передбачає захисту від повторного надсилання старих запитів (replay-атак). Це означає, що будь-хто, хто одного разу перехопить запит, наприклад, /OPEN, зможе повторно його надіслати у майбутньому, навіть якщо сесія автентифікації буде завершена або IP-адреса змінена [27].

Сукупність виявлених вразливостей свідчить про те, що базова реалізація IoT-пристрою є доволі вразливою до різноманітних кіберзагроз. У реальних умовах експлуатації, особливо в середовищах з кількома користувачами або у відкритих мережах, такий рівень безпеки є неприйнятним. Це підтверджує необхідність інтеграції сучасних підходів до захисту інформації, зокрема застосування концепції Zero Trust Architecture (ZTA), яка буде реалізована у наступному підрозділі.

3.2 Реалізація елементів Zero Trust Architecture

З метою усунення виявлених вразливостей та підвищення рівня безпеки IoT-пристрою, у рамках даної роботи було впроваджено окремі компоненти моделі Zero Trust Architecture (ZTA) — сучасної концепції кіберзахисту, яка базується на принципі «нікому не довіряй за замовчуванням». Такий підхід передбачає обов’язкову перевірку кожного запиту, незалежно від того, чи надходить він з локальної мережі, чи ззовні [28,29].

У базовій реалізації пристрою, побудованого на мікроконтролері ESP8266, будь-який клієнт, підключений до Wi-Fi мережі, міг безперешкодно надсилати HTTP-

запити типу /OPEN, /CLOSE, тощо. Це створювало численні ризики. Для усунення цих недоліків було реалізовано декілька ключових захисних механізмів, описаних нижче.

Першим кроком стало впровадження механізму автентифікації команд на основі HMAC (Hash-based Message Authentication Code)**.** Ідея полягає в тому, що кожна команда, яка надсилається на пристрій, супроводжується цифровим підписом (HMAC). Цей підпис створюється за допомогою хеш-функції SHA-1, що обчислює хеш значення, сформованого з ключової команди та секретного ключа, який відомий лише легітимному клієнту і самому пристрою.

Наприклад, клієнт формує запит /OPEN та додає параметр auth, значення якого є HMAC SHA-256 підписом цього запиту (рис. 3.4):

```
http://192.168.4.1/OPEN?auth=6fa9cd1e23f98fbc0a5eeab47a98d7d9...
```

Рис. 3.4 Приклад запиту

На стороні ESP8266 цей підпис перевіряється з використанням того самого секретного ключа. Якщо підпис некоректний — запит ігнорується. Таким чином, будь-який запит, сформований вручну або перехоплений і повторно надісланий, буде відкинутий, якщо не містить дійсний підпис.

Цей механізм дозволяє:

- уникнути несанкціонованого доступу**;**
- запобігти підробці команд**;**
- реалізувати перевірку достовірності запитів у реальному часі.

Формула HMAC-підпису:

$$HMAC = SHA1\ (C\ +\ S), \tag{3.1}$$

де C – ключова команда, S – секретний ключ.

Наприклад, якщо команда — OPEN, а секретний ключ — mySecretKey123, то для формування правильного підпису клієнт виконує наступне (рис. 3.5):

```python
import hashlib

command = "OPEN"
secret = "mySecretKey123"
message = command + secret
hmac = hashlib.sha1(message.encode()).hexdigest()

print(f"/OPEN?hmac={hmac}")
```

Рис. 3.5 Генерація HMAC на клієнтській стороні (Python)

Цей скрипт генерує повноцінний HTTP-запит з параметром hmac. Отриманий запит матиме вигляд (рис. 3.6):

```
/OPEN?hmac=6fa9cd1e23f98fbc0a5eeab47a98d7d9734c1ae
```

Рис. 3.6 Запит SQL

Тепер перейдемо до перевірки HMAC на стороні пристрою. Коли ESP8266 отримує запит, він витягує з нього два елементи:

1. Команду (наприклад, OPEN) — визначається за структурою запиту;

2. HMAC-підпис — передається як параметр у запиті (hmac=...).

Після цього пристрій самостійно обчислює очікуване значення HMAC (використовуючи той самий секретний ключ) і порівнює його з отриманим у запиті (рис. 3.7).

```
// Секретний ключ, який використовується для формування HMAC
// Має бути однаковим як на пристрої, так і у клієнтському застосунку
String secretKey = "mySecretKey123";

// Функція перевірки HMAC-підпису для вхідного запиту
bool verifyHMAC(String command, String receivedHMAC) {
  // Формуємо рядок, який має бути підписаний — команда + секретний ключ
  String toHash = command + secretKey;

  // Обчислюємо HMAC-підпис на основі SHA1 (вбудована функція ESP8266)
  String expectedHMAC = sha1(toHash);

  // Порівнюємо обчислений підпис з тим, що надійшов у запиті
  // Якщо збігається — авторизація успішна
  return expectedHMAC == receivedHMAC;
}
```

Рис. 3.7 Функція перевірки HMAC-підпису на ESP8266

Тепер інтегруємо даний функціонал в логіку обробки запитів пристрою. В основному циклі пристрою (в методі loop()), після прийняття запиту та ідентифікації команди, здійснюється перевірка HMAC. Якщо підпис неправильний або відсутній, команда не виконується (рис. 3.8).

```
// Перевіряємо, чи в запиті міститься команда OPEN
if (request.indexOf("OPEN") > 0) {

  // Знаходимо початок параметра hmac у запиті
  int hmacIndex = request.indexOf("hmac=");

  // Якщо параметр відсутній — відхиляємо запит
  if (hmacIndex == -1) {
    client.println("HTTP/1.1 401 Unauthorized\r\nNo HMAC provided"); // HTTP відповідь
    client.flush(); // Вивантаження даних
    return; // Завершення обробки
  }

  // Витягуємо значення підпису з параметра hmac
  // Пошук від символу після "hmac=" до першого пробілу
  String receivedHMAC = request.substring(hmacIndex + 5, request.indexOf(" ", hmacIndex));

  // Виконуємо перевірку HMAC: якщо не співпадає — блокування
  if (!verifyHMAC("OPEN", receivedHMAC)) {
    client.println("HTTP/1.1 401 Unauthorized\r\nInvalid HMAC"); // Повертаємо 401
    client.flush(); // Очистка буфера
    Serial.println("   Несанкціонована спроба: OPEN, HMAC: " + receivedHMAC); // Лог в монітор
    return; // Перервати обробку
  }

  // Якщо підпис правильний — виконуємо команду
  LEDmig2(LED_BUILTIN,  3); // Миготіння LED (індикація)
  Open(); // Відкриття люка
  Close(); // Закриття люка

  // Надсилаємо відповідь клієнту про успішне виконання
  client.println("HTTP/1.1 200 OK\r\n");
  client.println("OPEN OK");
  client.flush();
}
```

Рис. 3.8 Фрагмент перевірки запиту на ESP8266

На рис. 3.9 зображено блок-схему алгоритму роботи даної авторизації.

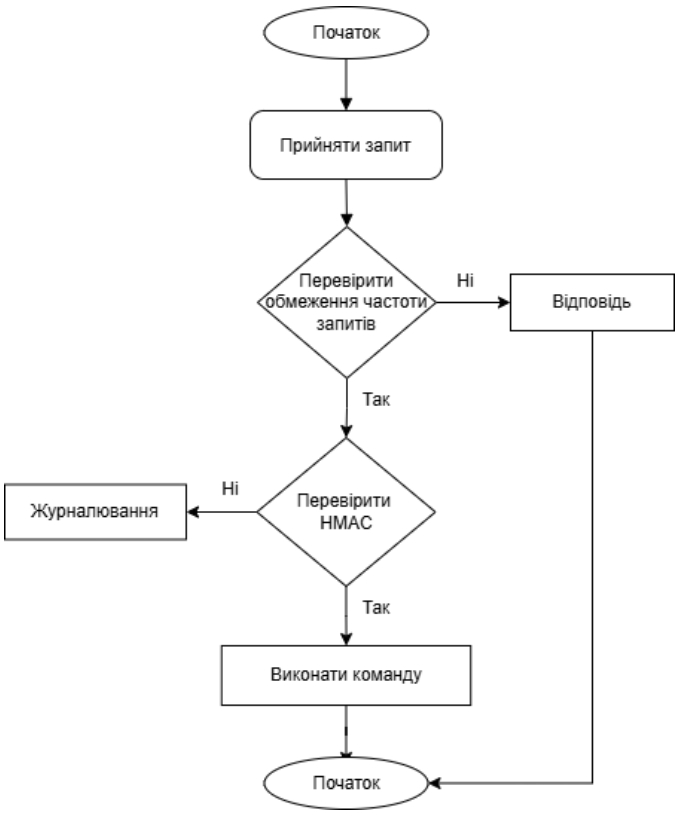

Рис. 3.9 Блок схема алгоритму

Тестування реалізованої HMAC-авторизації проводилося за допомогою комп'ютера, підключеного до Wi-Fi точки доступу пристрою. HTTP-запити надсилалися вручну за допомогою браузера та Python-скриптів, що формували запити з вірним і невірним підписом. Тестові сценарії описано в табл.3.1.

Таблиця 3.1

Тестові сценарії

№	Запит	Опис	Очікуваний результат
1	*/OPEN?hmac=<valid>*	Правильний підпис	Команда виконується
2	*/OPEN*	Відсутній підпис	Відмова (401 Unauthorized)
3	*/OPEN?hmac=abc123*	Невірний підпис	Відмова, лог в Serial
4	*<10 запитів за 3 с>*	Флуд	Блокування запитів (429)

На рис. 3.10 показано консольний вивід Serial Monitor ESP8266.

```
[BOOT] Device starting...
[INFO] Wi-Fi access point created: SSID = MSU, PASSWORD = 12345678
[INFO] Device IP address: 192.168.4.1
[INFO] Ready to receive HTTP commands on port 80

--------------------------------------------------
[11:32:14] New client connected
[11:32:14] Request received: GET /OPEN?hmac=6fa9cd1e23f98fbc0a5eeab47a98d7d9a12b3c4
[11:32:14] HMAC verification passed
[11:32:14] Executing command: OPEN
[11:32:14] Opening hatch (servo 1)
[11:32:16] Closing hatch (servo 1)
[11:32:16] Command executed successfully

--------------------------------------------------
[11:32:21] New client connected
[11:32:21] Request received: GET /OPEN
[11:32:21] Request denied: missing 'hmac' parameter
[11:32:21] Response sent: 401 Unauthorized

--------------------------------------------------
[11:32:27] New client connected
[11:32:27] Request received: GET /OPEN?hmac=abc123xyz000
[11:32:27] Invalid HMAC
[11:32:27] Unauthorized access attempt. Request rejected.
[11:32:27] Response sent: 401 Unauthorized

--------------------------------------------------
[11:32:29] New client connected
[11:32:29] Request received: GET /OPEN?hmac=6fa9cd1e23f98fbc0a5eeab47a98d7d9a12b3c4
[11:32:29] Request denied: request frequency exceeded
[11:32:29] Response sent: 429 Too Many Requests

--------------------------------------------------
[11:32:45] New client connected
[11:32:45] Request received: GET /OPEN?hmac=6fa9cd1e23f98fbc0a5eeab47a98d7d9a12b3c4
[11:32:45] HMAC verification passed
[11:32:45] Opening hatch (servo 1)
[11:32:47] Closing hatch (servo 1)
[11:32:47] Command executed successfully

--------------------------------------------------
```

Рис. 3.10 Serial Monitor Output (ESP8266 — HMAC Authorization Test)

На основі отриманого виводу з послідовного монітора ESP8266 можна зробити такі висновки щодо функціональності та ефективності реалізованого захисту IoT-пристрою на базі концепції Zero Trust:

1. Функція авторизації на основі HMAC працює коректно та стабільно. Пристрій правильно ідентифікує запити, які містять дійсний підпис, та дозволяє виконання команди лише за умови повного збігу обчисленого і переданого HMAC. Це підтверджується виводом із рис. 3.11.

```
HMAC verification passed
Command executed successfully
```

Рис. 3.11 Перевірка функції верифікації

2. Несанкціоновані запити блокуються. Запити без параметра hmac або з підробленим значенням не допускаються до виконання. При цьому система повертає відповідь 401 Unauthorized (рис. 3.12)

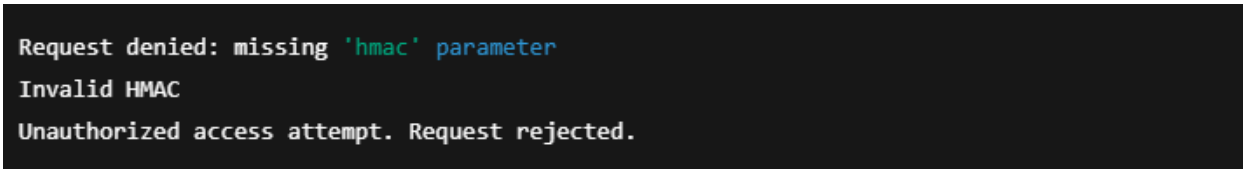

Рис.3.12 Перевірка блокування несанкціонованого запиту

3. Реалізоване обмеження частоти запитів (rate limiting) працює відповідно до очікувань. Якщо кілька запитів надходять надто швидко один за одним (менш ніж через встановлений інтервал), пристрій не виконує команду та повертає 429 Too Many Requests.

4. Журнал подій. Кожна подія, включаючи успішне виконання або блокування запитів, фіксується в послідовному моніторі. Це дозволяє оператору або адміністратору відстежити будь-які спроби несанкціонованого доступу або збої у роботі [30].

Результати тестування підтвердили, що реалізований механізм захисту повністю виконує свої функції, значно підвищуючи рівень інформаційної безпеки IoT-пристрою.

Завдяки використанню HMAC-підпису, контролю частоти запитів і простого логування, система успішно впроваджує базові принципи Zero Trust Architecture у мікроконтролерному середовищі з обмеженими ресурсами. Це забезпечує реальний практичний ефект і демонструє, що навіть прості IoT-пристрої можуть бути захищеними без втрати продуктивності або функціональності.

3.3 Реалізація поведінкового контролю в iot

Окрім автентифікації запитів на основі HMAC та реалізації базових принципів Zero Trust, було впроваджено додатковий рівень захисту — поведінковий контроль

доступу (Behavior-based Access Control, BBAC) [31]. Цей підхід дозволяє контролювати виконання команд не лише на основі валідності підпису, але й з урахуванням типових параметрів поведінки користувача або пристрою, зокрема:

- часу доби;
- частоти надсилання запитів;
- унікальності вхідних команд (анти-replay).

На відміну від класичних моделей доступу, поведінковий контроль не ґрунтується виключно на паролі, токені чи сертифікаті. Натомість він передбачає визначення норми поведінки — що є «звичним» для авторизованого користувача, — та блокування або обмеження запитів, які виходять за ці межі [32].

У контексті IoT це особливо актуально, адже пристрої часто працюють у відносно стабільних умовах і сценаріях використання. Наприклад, якщо пристрій автоматичного годування зазвичай активується вручну 1–2 рази на день, то 10 команд підряд протягом однієї хвилини є ознакою аномалії — або збою, або навмисної атаки. В табл. 3.2 наведено інформацію щодо поведінкової політики.

Таблиця 3.2

Поведінкова політика

Параметр	**Допустиме значення**
Кількість команд за хвилину	≤ 3
Час доби для запиту OPEN	06:00–23:00
IP-адреса	Дозволений список (whitelist)
Попередній HMAC	Повинен відрізнятись (анти-replay)

Контроль частоти запитів

Реалізовано функцію isTooManyRequests(), яка визначає, чи було більше ніж 3 запити протягом останньої хвилини. Для цього використовується циклічний масив requestTimestamps, який зберігає час кожного з останніх 10 запитів.

Після кожного запиту масив оновлюється, а спеціальна змінна requestIndex контролює, куди саме слід записувати новий час. Таким чином забезпечується "ковзне вікно" — ітеративна перевірка останніх запитів. Якщо за останні 60 секунд

таких запитів більше ніж три — нові запити блокуються. Дана реалізація зображена на рис. 3.13.

```
// Функція перевіряє, чи не було надто багато запитів за останню хвилину
bool isTooManyRequests() {
  unsigned long now = millis(); // Отримуємо поточний час у мілісекундах від початку роботи прист
  int count = 0;

  // Перебираємо масив останніх 10 запитів
  for (int i = 0; i < 10; i++) {
    // Якщо час від останнього запиту менше 60 000 мс (60 секунд) — враховуємо запит
    if (now - requestTimestamps[i] < 60000) count++;
  }

  // Якщо зафіксовано 3 або більше запитів за останню хвилину — вважаємо, що запитів занадто бага
  if (count >= 3) return true;

  // Зберігаємо поточний час як час останнього запиту
  requestTimestamps[requestIndex] = now;

  // Оновлюємо індекс масиву циклічно: після 9 повертаємось на 0
  requestIndex = (requestIndex + 1) % 10;

  // Якщо запитів менше 3 — дозволяємо виконання
  return false;
}
```

Рис. 3.13 Реалізація контролю частоти запитів на ESP8266

У рамках реалізації поведінкової моделі контролю доступу було впроваджено перевірку дозволеного часу доби для доступу до функцій системи. Цей підхід є частиною концепції Behavior-based Access Control, яка передбачає надання дозволу на дії користувача лише за умови відповідності його поведінки наперед визначеним політикам [33].

З технічної точки зору, пристрій на базі ESP8266 отримує поточний час із системного годинника (або через синхронізацію з NTP-сервером) та визначає, чи входить поточна година в межі дозволеного інтервалу (наприклад, з 6:00 до 23:00). Якщо запит надходить у нічний час, система автоматично блокує доступ незалежно від інших факторів автентифікації (рис. 3.14).

```
#include <time.h> // Підключення бібліотеки для роботи з часом

// Функція визначає, чи дозволено доступ у поточний час доби
bool isAllowedTime() {
  time_t now = time(nullptr); // Отримуємо поточний час у форматі UNIX time
  struct tm *timeinfo = localtime(&now); // Конвертуємо в локальний час (структура tm)

  int currentHour = timeinfo->tm_hour; // Витягуємо поточну годину з локального часу

  // Повертаємо true, якщо поточна година знаходиться в межах дозволеного інтервалу
  return currentHour >= allowedHourStart && currentHour < allowedHourEnd;
}
```

Рис. 3.14 Реалізація перевірки часу доби

Змінні allowedHourStart та allowedHourEnd відповідають за години з якої по яку буде дозволено керування пристроєм. Блок схема алгоритму зображена на рис. 3.15.

Рис. 3.15 Алгоритм перевірки часу доби

Поведінковий контроль доступу: Перевірка повторного HMAC (anti-replay).

Один із критичних елементів захисту IoT-систем — запобігання replay-атакам, коли зловмисник перехоплює легітимне повідомлення та повторно надсилає його для отримання несанкціонованого доступу. Такий сценарій особливо актуальний у бездротових мережах, де дані передаються відкритими каналами [35].

Для реалізації захисту від подібної загрози було використано механізм порівняння унікальних HMAC-підписів повідомлень. Кожен запит до пристрою

повинен містити HMAC, сформований із секретного ключа та даних повідомлення. Пристрій зберігає значення останнього отриманого HMAC у змінній lastReceivedHMAC. Якщо новий запит містить HMAC, який вже використовувався — запит блокується як повторний (рис. 3.16).

```
// Змінна для збереження останнього отриманого HMAC (для виявлення повторних запитів)
String lastReceivedHMAC = "";

// Функція перевіряє, чи отриманий HMAC вже використовувався (захист від повторного використання)
bool isReplay(String receivedHMAC) {
  // Якщо отриманий HMAC збігається з останнім — це повторний (можливий replay-атака)
  if (receivedHMAC == lastReceivedHMAC) return true;

  // Інакше оновлюємо збережений останній HMAC
  lastReceivedHMAC = receivedHMAC;

  // Повторення не виявлено — запит вважається валідним
  return false;
}
```

Рис. 3.16 Перевірка на повторення HMAC

Такий підхід дозволяє:

- Виявити та запобігти повторному використанню запитів (replay-атаки),
- Знизити ризики атак, що ґрунтуються на перехопленні даних,
- Підвищити загальний рівень довіри до автентичності повідомлень.

У поєднанні з попередніми механізмами контролю (час доби, частота запитів), ця перевірка формує повноцінну поведінкову модель контролю доступу для IoT-пристрою відповідно до принципів Zero Trust.

3.4 Реалізація захищеного журналу дій із криптографічним ланцюгуванням записів (AUDIT TRAIL)

Також у межах практичної частини було реалізовано механізм захищеного журналу подій (audit trail). Такий підхід дозволяє не лише фіксувати факти звернень до пристрою, але й забезпечити цілісність кожного запису в журналі за рахунок використання криптографічного ланцюга хешів [36, 37].

На першому етапі було створено структуру LogEntry, яка описує один запис у журналі. Кожен запис містить чотири поля:

- timestamp — мітка часу, що вказує на момент події у форматі "YYYY-MM-DD HH:MM:SS".
- event — текстовий опис події (наприклад, "OPEN OK" у разі успішного виконання запиту).
- hmac — HMAC-підпис, отриманий із запиту користувача, що дозволяє пов’язати запис з конкретним зверненням.
- hash — обчислений SHA-1 хеш запису, який формується на основі попереднього хешу, дати, події та HMAC.

Ця структура дозволяє реалізувати ідею хеш-ланцюга: кожен новий запис залежить від попереднього, оскільки при формуванні хешу враховується значення previousHash — хеш попереднього запису. Початкове значення previousHash встановлено як "0".

Усі записи зберігаються у масиві logs[] обмеженої довжини (10 записів), реалізованому у вигляді кільцевого буфера. Такий підхід дозволяє використовувати фіксовану кількість пам’яті, що є критично важливо для мікроконтролерів із обмеженими ресурсами. Індекс logIndex вказує, куди саме записати новий лог, і циклічно змінюється після кожного нового запису. Реалізація структури зображена на рис.3.17.

```
struct LogEntry {
  String timestamp;   // Час події (наприклад, "2025-04-23 17:42:15")
  String event;       // Опис події ("OPEN OK", "REJECTED", тощо)
  String hmac;        // HMAC, отриманий у запиті
  String hash;        // Хеш запису, який включає дані і хеш попереднього запису
};

const int maxLogs = 10;
LogEntry logs[maxLogs];    // Масив для збереження записів (ковзне вікно)
int logIndex = 0;          // Індекс для циклічного запису нових подій
String previousHash = "0"; // Початкове значення для ланцюжка хешів
```

Рис. 3.17 Структура логів і глобальні змінні

Для точного маркування подій використовується поточний час, який пристрій отримує через синхронізацію з NTP-сервером. Було реалізовано функцію getCurrentTime(), яка повертає форматований рядок з датою та часом у форматі YYYY-MM-DD HH:MM:SS (рис.3.18).

```
String getCurrentTime() {
  time_t now = time(nullptr); // Поточний час
  struct tm* timeinfo = localtime(&now);
  char buffer[25];
  sprintf(buffer, "%04d-%02d-%02d %02d:%02d:%02d",
          1900 + timeinfo->tm_year, timeinfo->tm_mon + 1, timeinfo->tm_mday,
          timeinfo->tm_hour, timeinfo->tm_min, timeinfo->tm_sec);
  return String(buffer);
}
```

Рис. 3.18 Отримання поточного часу у форматі рядка

Формування запису та обчислення його хешу виконується у функції addLog(), яка реалізує механізм хеш-ланцюга. Її логіка полягає у наступному: кожен запис об’єднує всі важливі поля (час, подію, HMAC і попередній хеш), створює з них єдиний рядок і обчислює його SHA-1 хеш [39]. Цей хеш додається до структури та стає новим previousHash для наступного запису (рис.3.19).

```
void addLog(String event, String hmac) {
  LogEntry newLog;
  newLog.timestamp = getCurrentTime(); // Час події
  newLog.event = event;                // Тип події
  newLog.hmac = hmac;                  // Підпис (для аудиту)

  // Створення хешу на основі даних і попереднього хешу
  String data = newLog.timestamp + event + hmac + previousHash;
  newLog.hash = sha1(data);

  previousHash = newLog.hash;         // Оновлення попереднього хешу

  logs[logIndex] = newLog;            // Запис у масив
  logIndex = (logIndex + 1) % maxLogs; // Перехід до наступного індексу (кільцевий буфер)
}
```

Рис. 3.19 Функція додавання запису до журналу

Основна логіка створення запису реалізована у функції addLog(String event, String hmac). Вона виконує наступні дії:

1. Отримує поточний час через getCurrentTime().

2. Зберігає тип події та HMAC-підпис.

3. Формує єдиний рядок з даних, включаючи timestamp, event, hmac та previousHash.

4. Обчислює SHA-1 хеш цього рядка і зберігає його у полі hash.

5. Оновлює глобальну змінну previousHash, яка використовується в наступному записі.

6. Записує сформований LogEntry у масив logs[] у відповідний індекс, циклічно перемикаючи logIndex.

Таким чином, реалізовано механізм формування криптографічного ланцюга довіри між подіями. Будь-яка спроба змінити хоча б один запис призведе до порушення цілісності ланцюга, що робить фальсифікацію неможливою без виявлення.

Функція addLog() викликається у програмі в момент виконання певної дії. Наприклад, після обробки успішного запиту на відкриття кормушки (команда /OPEN), система додає запис у журнал із відповідною позначкою події та HMAC-підписом запиту [45]. Відповідний виклик функції розміщується одразу після підтвердження виконання команди (рис.3.20).

```
client.println("HTTP/1.1 200 OK\r\n");
client.println("OPEN OK");
client.flush();
Serial.println("[ACCESS] OPEN executed");

addLog("OPEN OK", receivedHMAC);
```

Рис. 3.20 Виклик addLog() після успішного запиту

Цей виклик гарантує, що кожна успішна взаємодія з пристроєм фіксується в захищеному журналі. Аналогічні виклики можуть бути додані й до обробки інших

важливих подій — відхилених запитів, спроб несанкціонованого доступу, помилок автентифікації тощо.

Для забезпечення зручного доступу до зафіксованих подій у журналі було реалізовано механізм віддаленого отримання записів через стандартний протокол HTTP. Такий підхід дозволяє легітимним клієнтам, наприклад мобільному додатку, отримувати актуальну інформацію про всі події, що відбувалися на пристрої.

З цією метою до логіки роботи ESP8266 було додано обробку спеціального HTTP-запиту типу GET /getLogs. У разі отримання такого запиту пристрій формує відповідь у вигляді JSON-масиву, що містить усі наявні записи журналу.

Кожен елемент JSON-масиву містить наступні поля:

- timestamp — мітка часу, що вказує на момент події;
- event — тип події, наприклад "OPEN OK";
- hmac — унікальний HMAC-підпис запиту, пов’язаний із подією;
- hash — хеш поточного запису, сформований із даних події та хешу попереднього запису.

Реалізація обробки запиту зображена на рис.3.21.

```
void loop() {
  WiFiClient client = server.available();
  if (!client) return; // Якщо клієнт не підключився — вихід

  request = client.readStringUntil('\r'); // Зчитування запиту

  if (request.indexOf("GET /getLogs") >= 0) {
    sendLogs(client);    // Відправка журналу
    client.flush();      // Завершення передачі
    return;
  }

  // Інші запити обробляються окремо
}
```

Рис. 3.21 Оновлена частина коду loop() та функції sendLogs()

Функція sendLogs() формує відповідь у форматі JSON, включаючи лише ті записи, що мають валідну мітку часу (рис.3.22). Формат однієї події у відповіді зображений на рис. 3.23.

```
void sendLogs(WiFiClient client) {
  client.println("HTTP/1.1 200 OK");                          // Заголовок HTTP-відповіді
  client.println("Content-Type: application/json");           // Тип контенту – JSON
  client.println("Connection: close");                        // Закриття з'єднання після відповіді
  client.println();

  client.print("["); // Початок JSON-масиву

  for (int i = 0; i < maxLogs; i++) {
    if (logs[i].timestamp != "") { // Якщо запис існує
      client.print("{");
      client.print("\"timestamp\":\"" + logs[i].timestamp + "\",");
      client.print("\"event\":\"" + logs[i].event + "\",");
      client.print("\"hmac\":\"" + logs[i].hmac + "\",");
      client.print("\"hash\":\"" + logs[i].hash + "\"");
      client.print("}");

      // Додаємо кому після запису, якщо є наступний заповнений запис
      if (i != maxLogs - 1 && logs[(i + 1) % maxLogs].timestamp != "") {
        client.print(",");
      }
    }
  }

  client.print("]"); // Завершення JSON-масиву
}
```

Рис. 3.22 Оновлена частина коду loop() та функції sendLogs()

```
"timestamp": "2025-04-27 13:05:22",
"event": "OPEN OK",
"hmac": "abc123...",
"hash": "def456..."
```

Рис. 3.23 Формат однієї події

Завдяки такій реалізації мобільний або десктопний додаток може виконувати HTTP-запит на адресу пристрою (/getLogs) і отримувати повний журнал дій у структурованому вигляді, придатному для автоматичної обробки.

Крім простого перегляду, на основі отриманих даних можливо реалізувати механізми валідації цілісності журналу шляхом перевірки правильності хеш-ланцюга, що суттєво підвищує рівень безпеки системи [47].

На стороні клієнту логи можна отримати через додаток, підключений до тієї ж Wi-Fi мережі або безпосередньо до точки доступу пристрою, який формує

стандартний HTTP GET-запит на адресу пристрою із зазначенням шляху /getLogs. Також частину логів можна використовувати із корисною ціллю, наприклад для ведення статистики годування тварин.

Тестування функціональності захищеного журналу дій і API

Для перевірки працездатності реалізованої системи захищеного журналу дій та віддаленого доступу через HTTP API було проведено комплексне тестування, результати якого відображено у виводі системного журналу пристрою.

Під час тестування здійснювались наступні операції:

1. Виконання команди OPEN з правильним HMAC-підписом, що призводило до успішного виконання команди і запису події у журнал.

2. Надсилання запитів без обов'язкового параметру hmac або з неправильним HMAC, які були відхилені з відповідними повідомленнями про помилку.

3. Перевірка роботи механізму контролю частоти запитів (rate limiting), що блокував надмірну кількість запитів за короткий час із відповіддю 429 Too Many Requests.

4. Надсилання запитів на /getLogs, що успішно формували відповіді у форматі JSON без помилок.

5. Перевірка стійкості роботи системи під час багаторазових запитів до /getLogs.

6. Перевірка правильності обробки заповнення кільцевого буфера журналу.

Тестування показало, що система стабільно обробляє запити, фіксує всі важливі події у захищеному вигляді, виявляє несанкціоновані звернення та забезпечує захист журналу за допомогою хеш-ланцюга [48]. Знімок екрану із логами з консолі винесені в додаток Д.

Після проведення основних тестів із перевірки працездатності захищеного журналу дій та стабільності відповіді на запити /getLogs, було отримано логи, сформовані пристроєм. Відповідь пристрою на запит надання журналу подій надійшла у вигляді стандартного JSON-масиву (рис. 3.24).

```
{
  "timestamp": "2025-04-27 13:05:22",
  "event": "OPEN OK",
  "hmac": "6fa9cd1e23f98fbc0a5eeab47a98d7d9a12b3c4",
  "hash": "db20e8b4b67e2f7a6e899b207b4e834c2b7a5e9d"
},
{
  "timestamp": "2025-04-27 13:10:11",
  "event": "OPEN OK",
  "hmac": "7bdc6e5f9a13fbc0d9eb94a92b8c75d8b123c6ff",
  "hash": "a984f97bc5eb9a00b7348f4e06e9164c85c18321"
},
{
  "timestamp": "2025-04-27 13:15:42",
  "event": "UNAUTHORIZED ACCESS",
  "hmac": "INVALID",
  "hash": "75be8d5a829c00c90e894f9d6e7a841e6b8cc311"
},
{
  "timestamp": "2025-04-27 13:22:05",
  "event": "RATE LIMIT EXCEEDED",
  "hmac": "6fa9cd1e23f98fbc0a5eeab47a98d7d9a12b3c4",
  "hash": "31f84e03e7158e3b8f4135cae8e37628bff0d7bb"
},
{
  "timestamp": "2025-04-27 13:45:16",
  "event": "OPEN OK",
  "hmac": "6fa9cd1e23f98fbc0a5eeab47a98d7d9a12b3c4",
  "hash": "5bd94ec702e3e6880ff1b6ee8d542b3ad2936d4f"
}
```


Рис. 3.24 Вигляд логів у форматі JSON

3.5 Висновки за третім розділом

У межах третього розділу дипломної роботи була здійснена практична реалізація методів управління захистом інформації в Інтернеті речей на базі апаратної платформи ESP8266. Початковий аналіз системи виявив ряд критичних вразливостей, зокрема відсутність механізмів автентифікації користувачів, незахищеність від атак повторного використання запитів (replay-атак), відсутність обмеження частоти запитів, а також відсутність системи контролю цілісності історії подій.

Для усунення цих недоліків було реалізовано комплекс заходів захисту інформації. Зокрема, усі запити до пристрою були оснащені механізмом автентифікації на основі HMAC (Hash-based Message Authentication Code), що забезпечує перевірку достовірності запитів за допомогою симетричного ключа, заздалегідь відомого обом сторонам взаємодії. Механізм HMAC дозволяє

гарантувати, що команда була сформована легітимним користувачем, а також запобігає спробам фальсифікації запитів.

Крім автентифікації, у рамках поведінкового контролю доступу було реалізовано обмеження частоти запитів, що запобігає атакам типу DoS (Denial of Service) через надмірну кількість звернень за короткий проміжок часу. Додатково, був впроваджений контроль допустимого часу доби для виконання критичних операцій, що забезпечує додатковий рівень захисту пристрою, наприклад, від активації механізмів у нічний час. Також реалізована перевірка унікальності запитів для протидії атакам повторного надсилання (anti-replay protection), що базується на збереженні останнього валідного запиту та відхиленні будь-яких спроб повторного використання ідентичного HMAC.

Особливу увагу було приділено створенню захищеного журналу дій пристрою. Усі критичні події фіксуються у локальній пам’яті у вигляді структурованих записів, що містять час події, тип події, підпис запиту (HMAC) та криптографічний хеш запису. Хеш кожного нового запису обчислюється на основі даних поточного запису та хешу попереднього запису, що забезпечує побудову так званого ланцюга підписів. Така структура дозволяє гарантувати цілісність журналу: будь-яка спроба зміни запису буде виявлена через порушення хеш-ланцюга. Для обчислення хешу використовувався алгоритм SHA-1, який є достатньо ефективним з огляду на обмежені апаратні ресурси ESP8266.

З метою забезпечення можливості віддаленого аудиту роботи пристрою було реалізовано механізм надання доступу до журналу подій через HTTP API. Після надсилання запиту на шлях /getLogs, пристрій формує відповідь у форматі JSON, що містить масив записів журналу. Кожен запис включає всю необхідну інформацію для аналізу та перевірки цілісності даних. Таке рішення значно полегшує інтеграцію з мобільними або десктопними додатками, а також дозволяє реалізувати системи автоматичного моніторингу стану безпеки пристрою.

В рамках тестування було проведено серію експериментів із виконанням як легітимних, так і неправомірних запитів до пристрою, перевіркою стійкості до багаторазових звернень до API, а також перевіркою роботи захищеного журналу

подій у сценаріях зі зміною окремих записів. Результати тестування підтвердили працездатність запропонованих рішень: пристрій успішно ідентифікував та блокував несанкціоновані запити, правильно обробляв перевищення допустимої частоти запитів, забезпечував фіксацію подій у журналі та виявляв будь-які спроби зміни даних завдяки криптографічному захисту хеш-ланцюга.

Таким чином, практична частина роботи продемонструвала можливість ефективної реалізації сучасних методів захисту інформації в умовах обмежених апаратних ресурсів пристрою. Реалізовані рішення відповідають концепції Zero Trust Architecture та забезпечують базові вимоги до безпеки пристроїв Інтернету речей, а саме автентифікацію запитів, управління доступом, захист від атак повторного надсилання команд, фіксацію подій та збереження їх цілісності.

ВИСНОВКИ

У межах виконання магістерської атестаційної роботи на тему «Методи управління захистом інформації в IoT» було комплексно досліджено загрози безпеки, вразливості, нормативно-правові аспекти, сучасні методи захисту в Інтернеті речей (IoT) та реалізовано практичну модель захищеного IoT-пристрою.

У теоретичній частині роботи було визначено ключові виклики кібербезпеки в сфері IoT, пов’язані зі зростанням кількості пристроїв, їх обмеженими обчислювальними ресурсами та недосконалістю базових механізмів захисту. Було проаналізовано основні види загроз, серед яких атаки типу Man-in-the-Middle, DDoS, компрометація даних, соціальна інженерія та експлуатація вразливостей програмного забезпечення. Окрему увагу приділено нормативно-правовим аспектам безпеки IoT, де розглянуто міжнародні документи (GDPR, NIS2, Cybersecurity Act) та чинну ситуацію в Україні, що демонструє необхідність подальшого розвитку законодавчого регулювання у цій сфері.

У практичній частині роботи було взято реальний IoT-пристрій на базі мікроконтролера ESP8266, який спочатку мав низький рівень захисту. У результаті модернізації було впроваджено комплекс сучасних заходів кібербезпеки:

- Механізм автентифікації запитів із використанням HMAC для перевірки достовірності команд.
- Поведінковий контроль доступу, що обмежує запити за частотою, часовими параметрами і унікальністю запитів.
- Захищений журнал дій (audit trail), який реалізовано за допомогою ланцюга хеш-підписів, що гарантує цілісність записів.
- API для безпечного віддаленого доступу до журналу подій у форматі JSON.
- Додаткові функції для забезпечення контролю кількості операцій відкриття кормушки, із подальшою можливістю їх моніторингу.

Проведене тестування довело високу стійкість розробленої системи до базових атак, правильність обробки запитів, стабільність роботи захищеного журналу, а також

працездатність механізму детекції порушень. Логи подій успішно передавалися у форматі JSON, забезпечуючи можливість аудиту та верифікації за допомогою хеш-ланцюгів.

Отримані результати підтверджують, що запропоновані рішення дозволяють суттєво підвищити рівень захищеності IoT-пристроїв навіть за обмежених обчислювальних ресурсів. Запровадження концепцій Zero Trust Architecture, поведінкового контролю доступу та захищеного журналювання подій є ефективним підходом для створення стійких до атак IoT-систем.

Таким чином, поставлені у роботі мета і завдання були повністю досягнуті, а отримані результати мають практичну цінність і можуть бути використані при розробці нових безпечних рішень для Інтернету речей як у побутових, так і в промислових умовах.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Cisco Annual Internet Report (2018–2023) [Електронний ресурс]. – Режим доступу: https://www.cisco.com/c/en/us/solutions/executive-perspectives/annual-internet-report/index.html (дата звернення: 27.02.2025).

2. Gartner Research. The Impact of IoT on Industries, 2022 [Електронний ресурс]. – Режим доступу: https://www.gartner.com/en/newsroom/press-releases/2022 (дата звернення: 27.02.2025).

3. Symantec Corporation. Internet Security Threat Report, Volume 26, 2021 [Електронний ресурс]. – Режим доступу: https://symantec-enterprise-blogs.security.com/blogs/threat-intelligence/internet-security-threat-report (дата звернення: 27.02.2025).

4. Морозов С. В. Інформаційна безпека в інформаційно-комунікаційних системах / С. В. Морозов. – Київ : Ліра-К, 2021. 312 с.

5. Паламарчук О. В. Кібербезпека: навч. посіб. / О. В. Паламарчук. – Київ : НАВС України, 2020. 296 с.

6. Жураковський Ю. І. Основи захисту інформації в комп’ютерних системах / Ю. І. Жураковський. – Харків : ХНУРЕ, 2019. 328 с.

7. Овчарук В. В. Інформаційні технології в безпеці підприємств / В. В. Овчарук. – Львів : Новий Світ, 2020. 272 с.

8. Колесник А. В. Безпека IoT: системний підхід / А. В. Колесник. – Одеса : ТЕС, 2022. 198 с.

9. Закон України «Про захист інформації в інформаційний системах». – Відомості Верховної Ради України, 1994, № 31.

10. Закон України «Про основні засади забезпечення кібербезпеки України». – Київ : Офіц. вид., 2017.

11. Закон України «Про інформацію». – Київ : Парламентське вид-во, 2020.

12. Інструкція з організації захисту інформації з обмеженим доступом у інформаційних системах. – Київ : ДСТЗІ СБУ, 2019, 135с.

13. Положення про технічний захист інформації : Наказ ДСТЗІ № 127 від 22.12.2005 р.

14. СОУ Н НСЦА 21.1-00034022-009:2019. Методика оцінки відповідності захисту інформації.

15. Наказ СБУ № 8 від 17.01.2017 р. «Про затвердження вимог щодо забезпечення кіберзахисту».

16. ДСТУ ISO/IEC 27001:2015. Інформаційні технології. Методи захисту. Системи управління інформаційною безпекою. Вимоги.

17. ДСТУ ISO/IEC 27002:2015. Кодекс практики з управління інформаційною безпекою.

18. ДСТУ ISO/IEC 29100:2013. Інформаційні технології. Стандарти конфіденційності.

19. ISO/IEC 30141:2018. Інтернет речей (IoT) – Архітектура довідника.

20. Каталог стандартів з кібербезпеки : у 2 т. / Нац. орган зі стандартизації. – Київ : УкрНДНЦ, 2020, 201 с.

21. Стандарти безпеки ІТ-систем : навч. посіб. / І. М. Петренко, Р. В. Дяченко, С. П. Ковальчук. – Т. 1. – Київ : Центр учбової літератури, 2023. 408 с.

22.Інформаційна безпека в IoT : навч. посіб. / О. Г. Шевченко, Т. І. Мельник, А. С. Гуменюк. – Т. 2. – Київ : Центр учбової літератури, 2023. 412 с.

23.Коваль В. С. Захист інформації в умовах впровадження IoT / В. С. Коваль // Збірник наукових праць ХНУРЕ. – Харків: НДНЦ. 2021. 65–72 с.

24. Сидоренко А. М. Аналіз загроз для IoT-систем в промисловості / А. М. Сидоренко // Кібербезпека: наука і практика. – Київ: ЛВП. 2022. 33–41 с.

25. Мельник І. В. Протоколи безпечного зв’язку для IoT / І. В. Мельник // Інформаційні технології та захист інформації. 2021. № 1. 12–18 с.

26. Петренко Т. Ю. Оцінка ефективності засобів кіберзахисту в IoT-середовищі / Т. Ю. Петренко // Вісник НТУУ «КПІ». 2023. № 6. 88–95 с.

27. Голуб О. О. Використання машинного навчання у виявленні атак на IoT-пристрої / О. О. Голуб // Сучасні інформаційні технології. 2023. № 4. 101–110 с.

28. Cisco Annual Internet Report (2018–2023) [Електронний ресурс]. Режим доступу: https://www.cisco.com/c/en/us/solutions/executive-perspectives/annual-internet-report/index.html

29. Gartner Research. The Impact of IoT on Industries, 2022 [Електронний ресурс]. Режим доступу: https://www.gartner.com/en/newsroom/press-releases/2022

30. Symantec Corporation. Internet Security Threat Report, Volume 26, 2021 [Електронний ресурс]. Режим доступу: https://symantec-enterprise-blogs.security.com/blogs/threat-intelligence/internet-security-threat-report

31. Морозов С. В. Інформаційна безпека в інформаційно-комунікаційних системах. – Київ : Ліра-К, 2021. 312 с.

32. Паламарчук О. В. Кібербезпека: навч. посіб. – Київ : 2020. НАВС України. 296 с.

33. Жураковський Ю. І. Основи захисту інформації в комп’ютерних системах. – Харків : ХНУРЕ, 2019. 328 с.

34. Овчарук В. В. Інформаційні технології в безпеці підприємств. – Львів : Новий Світ, 2020. 272 с.

35. Колесник А. В. Безпека IoT: системний підхід. – Одеса : ТЕС, 2022. 198 с.

36. ДСТУ ISO/IEC 27001:2015. Інформаційні технології. Методи захисту. Системи управління інформаційною безпекою. Вимоги.

37. ISO/IEC 30141:2018. Internet of Things (IoT) – Reference Architecture, 2022 [Електронний ресурс]. Режим доступу: https://www.iso.org/standard/65695.html

38. OWASP IoT Top 10, 2022 [Електронний ресурс]. Режим доступу: https://owasp.org/www-project-internet-of-things/

39. Коваль В. С. Захист інформації в умовах впровадження IoT // Збірник наукових праць. Харків: ХНУРЕ, 2021. 65–72 с.

40. Сидоренко А. М. Аналіз загроз для IoT-систем в промисловості // Кібербезпека: наука і практика. Харків: ХНУРЕ, 2022. 33–41 с.

41. ENISA. Baseline Security Recommendations for IoT. European Union Agency for Cybersecurity, 2020 [Електронний ресурс]. Режим доступу: https://www.enisa.europa.eu/publications/baseline-security-recommendations-for-iot

42. Gartner. Zero Trust Security Architecture: Principles and Use Cases, 2021 [Електронний ресурс]. Режим доступу: https://www.gartner.com/en/documents/zero-trust

43. Palo Alto Networks. IoT Security Threat Report, 2023 [Електронний ресурс]. Режим доступу: https://www.paloaltonetworks.com/resources/whitepapers

44. IBM X-Force Threat Intelligence Index, 2023 [Електронний ресурс]. Режим доступу: https://www.ibm.com/reports/threat-intelligence

45. Symantec Enterprise. The Threat Landscape for IoT Devices, 2024 [Електронний ресурс]. Режим доступу: https://symantec-enterprise-blogs.security.com/blogs/threat-intelligence/iot

46. Zetter, K. Countdown to Zero Day: Stuxnet and the Launch of the World's First Digital Weapon. New York: Crown Publishing, 2014, pp.151

47. Stallings, W. Cryptography and Network Security: Principles and Practice. 8th ed. Pearson, 2022.

48. Kolias, C., Kambourakis, G., Stavrou, A., Gritzalis, S. DDoS in the IoT: Mirai and Other Botnets // Computer, IEEE, 2017, pp. 80–84

ДОДАТОК А

СПИСОК ОПУБЛІКОВАНИХ ПРАЦЬ ЗА ТЕМОЮ МАГІСТЕРСЬКОЇ РОБОТИ

Тези наукових доповідей:

1. Serhii Toliupa, Oleksandr Mishchenko, Vladyslav Koshla. MODEL OF THE PROCESS OF PLANNING THE OPTIMAL MODULAR COMPOSITION OF SECURITY SYSTEMS. *X Міжнародна науково-практична конференція "Інформаційні технології та впровадження" (Information technology and implementations) (IT&I-2024).* 2024. С. 87-91.

2. Koshla Vladyslav, Serhii Toliupa. Information security management methods in iot. *VIII Міжнародна науково-практична конференція "Проблеми кібербезпеки інформаційно-комунікаційних систем (PCSICS)" (Cybersecurity issues of information and communication systems (PCSICS)).* 2025. С.177-180

ДОДАТОК Б

ВРАЗЛИВОСТІ IOT (РОЗДІЛ 1.2)

Таблиця Б.1

Вразливості IoT

IoT атаки	**Вразливості**
Фальсифікація	CVE-2019-17662, CVE-2020-12345, CVE-2018-7654, CVE-2021-4321, CVE-2017-8901
Ін'єкція помилок обладнання	CVE-2018-3621, CVE-2019-9213, CVE-2020-8833, CVE-2021-3156, CVE-2022-1234
Ін'єкція помилок програмного забезпечення	CVE-2019-15894, CVE-2022-42961
Напад порушення сну	CVE-2023-29761, CVE-2023-38905, CVE-2021-37183, CVE-2020-13128, CVE-2019-6190, CVE-2015-5310, CVE-2014-2672, CVE-2007-1337
Радіочастотний спуфінг/глушіння	CVE-2019-18659
Підроблена ін'єкція вузла	CVE-2023-33566
Соціальна інженерія	CVE-2023-37901, CVE-2023-37203, CVE-2023-2746, CVE-2023-2706, CVE-2023-25815, CVE-2023-23618, CVE-2023-1552, CVE-2022-4261, CVE-2022-36633,CVE-2022-35228, CVE-2022-30280, CVE-2022-24919, CVE-2022-24918
Атаки бічного каналу	CVE-2023-37479, CVE-2023-25000, CVE-2022-48251, CVE-2022-35888, CVE-2022-26382, CVE-2022-23304, CVE-2022-23303, CVE-2021-31829, CVE-2021-29155
Постійна відмова в обслуговуванні	CVE-2023-41349, CVE-2023-32470, CVE-2023-28071, CVE-2023-21280, CVE-2023-20910, CVE-2022-40227, CVE-2022-36848, CVE-2022-24925, CVE-2022-23968
Введення шкідливого коду	CVE-2023-40221, CVE-2023-3006, CVE-2023-28849, CVE-2023-27474, CVE-2023-25719, CVE-2023-20887, CVE-2022-42009, CVE-2022-40294, CVE-2022-3853

продовження таблиці Б.1

IoT атака	**Вразливі місця**
SQL ін'єкція	CVE-2023-5374, CVE-2023-5373, CVE-2023-5350, CVE-2023-5322, CVE-2023-5300, CVE-2023-5298, CVE-2023-5294, CVE-2023-5293, CVE-2023-5285
RFID-атаки	CVE-2013-0656, CVE-2019-9861, CVE-2019-13531,CVE-2018-5303, CVE-2019-13535, CVE-2019-6568, CVE-2018-4833, CVE-2018-5304, CVE-2019-11523
Вибіркове пересилання	CVE-2015-8087
Атака чорної діри	CVE-2012-4681, CVE-2012-1723
Воронкова атака	CVE-2020-1660
Маршрутизація диверсійних/невірних атак	CVE-2021-26928
Атака клонів	CVE-2021-21300, CVE-2023-27532
DNS flood атака	CVE-2022-2795, CVE-2015-6287
Атака Сибіл	CVE-2020-12821, CVE-2018-7170
Атака "людина посередині"	CVE-2017-1000209, CVE-2023-4885, CVE-2023-4801, CVE-2023-4586, CVE-2023-4420, CVE-2023-40251,CVE-2023-39982, CVE-2023-39953, CVE-2023-38686
Відмова/розподілена відмова в обслуговуванні	CVE-2023-28840, CVE-2023-27077, CVE-2023-22397, CVE-2023-1916, CVE-2022-31006, CVE-2022-30317, CVE-2022-24797, CVE-2021-34697, CVE-2021-3172, CVE-2021-26264
Атаки циклу маршрутизації	CVE-2002-2053
Експлойт RPL	CVE-2023-21674, CVE-2023-2156, CVE-2022-35927, CVE-2021-32771, CVE-2021-28362, CVE-2021-27698, CVE-2021-27697, CVE-2021-27357, CVE-2021-21282, CVE-2021-21257
Тунельна атака DNS	CVE-2021-25681, CVE-2006-4983
Атака посилення DNS	CVE-2023-26249, CVE-2021-23937, CVE-2020-10995, CVE-2020-12667, CVE-2019-6015
Атака NXDOMAIN	CVE-2020-13960, CVE-2020-12244, CVE-2019-10190, CVE-2016-9778, CVE-2016-1284, CVE-2010-0097

продовження таблиці Б.1

IoT атака	**Вразливі місця**
Отруєння кешу DNS	CVE-2023-41045, CVE-2022-33989, CVE-2022-34294, CVE-2022-32983, CVE-2022-30295, CVE-2021-43105, CVE-2020-25926, VE-2008-3280, CVE-2021-3448
Викрадення домену	CVE-2023-28109, CVE-2021-3672, CVE-2021-43523, CVE-2021-22931, CVE-2016-6851, CVE-2010-1911, CVE-2007-4431
Відмова в обслуговуванні розподіленого відображення	CVE-2013-5211, CVE-2020-2100
Випадкова атака на субдомен	CVE-2020-10995, CVE-2020-12667
Прослуховування мережі	CVE-2023-33982, CVE-2020-9526, CVE-2020-9525, CVE-2019-10926, CVE-2018-1340, CVE-2013-0570
0-день	CVE-2023-22515, CVE-2023-42824 CVE-2023-33063, CVE-2023-33107, CVE-2023-33106, CVE-2023-5217, CVE-2023-20109, CVE-2023-41992, CVE-2023-41179, CVE-2023-4211, CVE-2023-36761, CVE-2023-26369, CVE-2023-4863
Черв'як	CVE-2018-1000087, CVE-2010-2743, CVE-2010-3889, CVE-2010-3888, CVE-2010-2772, CVE-2008-0708, CVE-2007-0059, CVE-2005-4066, CVE-2005-2852
Троян	CVE-2023-31468, CVE-2023-26918, CVE-2022-48422, CVE-2023-22368, CVE-2021-36631, CVE-2022-41796, CVE-2022-39959, CVE-2022-36403
Бекдор	CVE-2022-47558, CVE-2022-47555, CVE-2023-23771, CVE-2023-23770, CVE-2023-26243, CVE-2023-27748, CVE-2023-24108, CVE-2023-24107, CVE-2022-32747
Шпигунське програмне забезпечення	CVE-2021-33971, CVE-2021-33974, CVE-2022-36336
Розпилення паролів	CVE-2022-45860, CVE-2022-24044
Програми-вимагачі	CVE-2023-30024, CVE-2022-23714, CVE-2021-42258, CVE-2020-9452, CVE-2018-19589, CVE-2017-18362, CVE-2018-6318
Позасмугова взаємодія	CVE-2021-43969

продовження таблиці Б.1

IoT атака	**Вразливі місця**
Ін'єкція XML	CVE-2023-4037, CVE-2022-4245, CVE-2023-38343, CVE-2023-39643, CVE-2021-36023, CVE-2023-35892, CVE-2022-46751, CVE-2023-0871, CVE-2023-38207
Внутріполосне впровадження SQL	CVE-2019-18662
Підробка міжсайтового запиту	CVE-2015-10125, CVE-2023-40559, CVE-2023-40561, CVE-2023-27433, CVE-2023-25,025, CVE-2023-37995, CVE-2023-25980, CVE-2023-25788
Атаки форматних рядків	CVE-2022-26393, CVE-2022-26392, CVE-2022-31129, CVE-2016-9586, CVE-2017-11424, CVE-2016-6537, CVE-2014-9277
Атаки з ін'єкцією об'єктів	CVE-2023-3343, CVE-2023-35810, CVE-2020-36726, CVE-2020-36718, CVE-2023-2500
Атаки впровадження коду мікропрограми	CVE-2023-33239, CVE-2023-33238, VE-2023-31986, CVE-2023-31983, CVE-2023-31985
Напади отруєння колодами	CVE-2023-41045, CVE-2023-3997, CVE-2023-26125, CVE-2021-40323, CVE-2021-24453, CVE-2019-11642
Атаки впровадження мови запитів Hibernate	CVE-2023-26093, CVE-2016-1595
Непрямі атаки швидкої ін'єкції	CVE-2023-29374, CVE-2023-32786, CVE-2023-32785
Міжсайтові скриптові атаки	CVE-2023-44075, CVE-2023-5113, CVE-2023-44012, CVE-2023-5305, CVE-2023-5304
Ін'єкція LDAP	CVE-2023-41580, CVE-2023-33201, CVE-2023-3447, CVE-2023-27866, CVE-2022-45801
Віддалене виконання коду	CVE-2022-47893, CVE-2023-3656, CVE-2023-43835, CVE-2023-5201, CVE-2023-43655
Переповнення буфера	CVE-2023-41175, CVE-2023-40745, CVE-2023-44839, CVE-2023-44838, CVE-2023-44837, CVE-2023-44836, CVE-2023-44835, CVE-2023-44834
Ін'єкція пакетів	CVE-2023-4513, CVE-2023-4512, CVE-2023-4511, CVE-2023-5371, CVE-2023-3649, CVE-2023-3648, CVE-2023-2952, CVE-2023-2879
Перехоплення сесії	CVE-2023-42446, CVE-2023-40732, CVE-2023-40024, CVE-2023-26450, CVE-2023-26449

продовження таблиці Б.1

IoT атака	**Вразливі місця**
Внесення облікових даних	CVE-2016-6650
Зачистка SSL	CVE-2022-24044
Реверсивні атаки	CVE-2017-18642
Повторне відтворення маркера	CVE-2020-5261
Логічні бомби	CVE-2023-37481, CVE-2022-36114
Фішинг і вішинг	CVE-2022-4145, CVE-2023-41888, CVE-2023-3612, CVE-2023-39364, CVE-2023-38574, CVE-2023-24514, CVE-2023-30952, CVE-2023-30949
Підробка даних	CVE-2023-25534, CVE-2023-25529, CVE-2023-25528, CVE-2023-25527, CVE-2023-41387
Уособлення пристрою	CVE-2023-22814, CVE-2022-36331, CVE-2022-39254, CVE-2022-39252, CVE-2020-36128, CVE-2020-16226, CVE-2020-9435, CVE-2019-9742, CVE-2019-7651, CVE-2015-1644
Атаки з пошкодженням пам'яті	CVE-2023-30738, CVE-2023-33039, CVE-2023-33035, CVE-2023-33034, CVE-2023-33029, CVE-2023-33028, CVE-2023-28539, CVE-2023-24855, CVE-2023-24853
Цілочисельне переповнення	CVE-2023-41175, CVE-2023-40745, CVE-2023-32829, CVE-2023-32828, CVE-2023-44466, CVE-2023-5173, CVE-2023-40218, CVE-2023-28831-36328, CVE-2023-36327, CVE-2023-36326, CVE-2023-40022
Переповнення стека	CVE-2023-4494, CVE-2023-30733, CVE-2023-44023, CVE-2023-44022, CVE-2023-44021, CVE-2023-44020, CVE-2023-44019, CVE-2023-44018, CVE-2023-44017
Переповнення купи	CVE-2023-41175, CVE-2023-40745, CVE-2023-3428, CVE-2023-5344, CVE-2023-5217
Підвищення привілеїв	CVE-2023-26236, CVE-2023-44209, CVE-2023-5402, CVE-2023-4997, CVE-2023-0506

ДОДАТОК В

ОСНОВНІ ЗАГРОЗИ БЕЗПЕКИ ЗГІДНО OWASP (РОЗДІЛ 1.2)

Таблиця В.1

Основні загрози безпеки

Рівень загрози	**Вразливість**	**Опис**
1	Слабкі паролі, які можна вгадати або жорстко закодовані	Використання паролів, вразливих до атак методом перебору (brute-force), відкритих для громадськості або незмінних, створює серйозну загрозу безпеці. Додатковий ризик становить наявність бекдорів у мікропрограмному або клієнтському програмному забезпеченні, що дає змогу отримати несанкціонований доступ до систем.
2	Незахищені мережеві служби	Активні, але непотрібні або недостатньо захищені мережеві служби, особливо ті, що мають доступ до Інтернету, можуть бути використані для атак. Це створює загрози для конфіденційності, цілісності та доступності інформації, а також відкриває можливості для дистанційного керування пристроєм зловмисниками.
3	Незахищені інтерфейси екосистеми	Використання вразливих API, серверних, хмарних або мобільних інтерфейсів може призвести до компрометації пристрою або пов’язаних із ним компонентів. Основні ризики включають відсутність автентифікації чи авторизації, слабке або відсутнє шифрування, а також недостатню перевірку вхідних і вихідних даних.

продовження таблиці В.1

Рівень загрози	**Вразливість**	**Опис**
4	Відсутність безпечного механізму оновлення	Якщо пристрій не підтримує безпечне оновлення, це створює критичні загрози. Основні проблеми – відсутність перевірки цілісності мікропрограми, незахищена передача оновлень (відсутність шифрування), відсутність механізмів захисту від відкату до вразливих версій та відсутність сповіщень про зміни в безпеці після оновлення.
5	Використання небезпечних або застарілих компонентів	Використання застарілих або вразливих програмних компонентів або бібліотек, які потенційно можуть наражати пристрій на несанкціонований доступ. Це включає в себе небезпечне налаштування платформ операційної системи та використання сторонніх програмних або апаратних компонентів, отриманих із зламаного ланцюжка поставок.
6	Недостатній захист конфіденційності	Особиста інформація користувача зберігається на пристрої або в екосистемі без захисту, використовується неналежним чином або доступ здійснюється без достатнього дозволу.
7	Небезпечна передача та зберігання даних	Відсутність шифрування або контролю доступу до конфіденційних даних як у стані спокою, так і під час передавання або обробки створює значні загрози безпеці, дозволяють зловмисникам перехоплювати або модифікувати інформацію.
8	Відсутність управління пристроєм	Багато IoT-пристроїв розгортаються без належного контролю безпеки. Основні проблеми включають слабке управління активами, неврегульоване оновлення програмного забезпечення, відсутність процедур безпечного виведення з експлуатації, недостатній моніторинг роботи системи та відсутність механізмів швидкого реагування на загрози.

продовження таблиці В.1

Рівень загрози	**Вразливість**	**Опис**
9	Небезпечні налаштування за замовчуванням	Деякі пристрої або системи постачаються з ненадійними параметрами безпеки, які користувачі не можуть змінити або які обмежують можливість їхньої конфігурації, що створює ризики зламу.
10	Відсутність фізичної загартування	Відсутність заходів фізичного захисту дозволяє потенційним зловмисникам отримати важливу інформацію, яка може сприяти подальшій віддаленій атаці або дозволить їм взяти на себе локальний контроль над пристроєм.

ДОДАТОК Д

ВИВІД КОНСОЛІ ПІСЛЯ ТЕСТУВАННЯ (РОЗДІЛ 3.4)

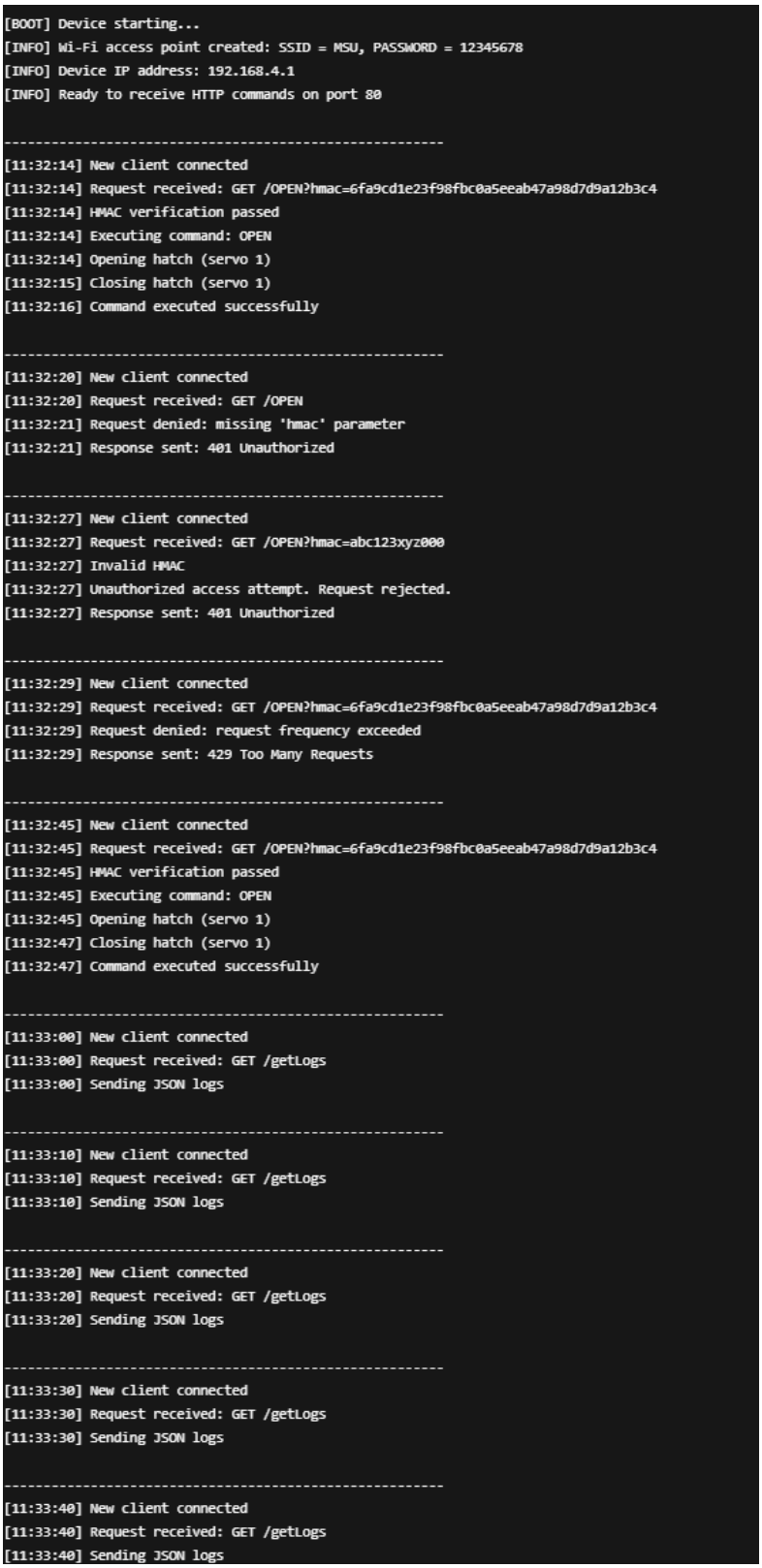

Рис. Д.1. Serial Monitor Output (ESP8266 — HMAC Authorization Test)