Міністерство освіти і науки України
Київський національний університет імені Тараса Шевченка

Факультет інформаційних технологій
Кафедра мережевих та інтернет технологій

ЗАТВЕРДЖУЮ
завідувач кафедри
мережевих та інтернет технологій
_____ Ю.В. Кравченко
«_____»_____2022 року

КВАЛІФІКАЦІЙНА РОБОТА МАГІСТРА

галузі знань 17 «Електроніка та телекомунікації»
за спеціальністю 172 «Телекомунікації та радіотехніка»
освітньо-професійна програма «Мережеві та інтернет технологій»

на тему:

МОДЕЛІ ТА МЕТОДИ ЗАБЕЗПЕЧЕННЯ ІНФОРМАЦІЙНОЇ БЕЗПЕКИ В МЕРЕЖАХ ДЕ ВИКОРИСТОВУЮТЬСЯ ІОТ ПРИСТРОЇ

Виконав: студент групи МІТ – 21(м)

Науменко Денис Ігорович _____
(прізвище ім'я по-батькові) (підпис)

Керівник: професор кафедри мережевих та інтернет технологій

д.т.н, професор Плющ О. Г. _____
(посада, прізвище ім'я по-батькові) (підпис)

Київ 2022

Міністерство освіти і науки України
Київський національний університет імені Тараса Шевченка

Факультет інформаційних технологій
Кафедра мережевих та інтернет технологій

ЗАТВЕРДЖУЮ
завідувач кафедри
мережевих та інтернет технологій

_____ Ю.В. Кравченко

«_____»_____2022 року

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Здобувачу вищої освіти Науменко Денис Ігорович
(прізвище, ім'я, по батькові)

1. Тема роботи:

«Моделі та методи забезпечення інформаційної безпеки в мережах де використовуються IOT пристрої»

затверджена на засіданні кафедри МІТ «31» серпня 2022 р. протокол № 1

2. Термін здачі закінченої роботи «05» грудня 2022р

3. Вихідні дані до проекту (роботи)

Розроблений код приватної Blockchain мережі для розгортання в хмарному середовищі GCP

Програмно визначена мережа SDN в Cisco Packet Tracer та конфігурація периметру SDP

4. Зміст пояснювальної записки (перелік питань, що їх потрібно розробити, обсяг – 70-90 стор.)

1. Дослідження та аналіз предметної області забезпечення інформаційної безпеки в в мережах де використовуються IoT пристрої.
2. Аналіз застосування рішень основаних на технологіях Blockchain та SDP поверх SDN для забезпечення інформаційної безпеки в мережах де використовуються IoT пристрої.
3. Надання рекомендацій та пропозицій щодо впровадження найкращих практик Для захисту IoT екосистем та мереж.

5. Перелік графічного матеріалу 8-12 слайдів

Методи та моделі захисту IoT екосистем.

Опис методів та моделей.

Результати власного дослідження.

Висновки.

Дата видачі завдання

Керівник роботи д.т.н., професор кафедри МІТ Плющ О.Г.
(підпис) (посада, прізвище, ім'я, по батькові)

Завдання прийняв до виконання
(підпис) (прізвище, ім'я, по батькові)

КАЛЕНДАРНИЙ ПЛАН ВИКОНАННЯ РОБОТИ

Номер	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	Підготовчий	24.10.2022	
2	Розділ 1	01.11.2022	
3	Розділ 2	15.11.2022	
4	Розділи 3, 4	01.12.2022	
5	Доповідь та слайди	05.12.2022	
6	Пояснювальна записка	05.12.2022	

Здобувач вищої освіти _____
(підпис) (прізвище, ім'я, по батькові)

Керівник _____
(підпис) (прізвище, ім'я, по батькові)

РЕФЕРАТ

Пояснювальна записка: 90с., 68 рис., 2 табл., 2 додатки, 15 джерел, 2 цитати.

Об’єкт дослідження: методи та моделі забезпечення інформаційної безпеки в мережах де використовуються IoT пристрої.

Предмет дослідження: методи захисту мереж та екосистем IoT на основі технологій Blockchain та програмно визначених мереж SDN .

Мета роботи: розробити код приватної Blockchain мережі для розгортання в хмарному середовищі GCP та програмно визначену мережу в ПЗ Cisco Packet Tracer.

Методи дослідження: системний підхід, аналіз, експеримент, опис, розробка, розгортання.

У основній частині дана характеристика сучасного стану технологій Blockchain та SDN для IoT.

У кваліфікаційній роботі було проведено аналіз існуючих технологій та підходів до захисту мереж де використовуються пристрої IoT.

Надано рекомендації та пропозиції щодо впровадження найкращих практик захисту IoT екосистем. Описано метод унеможливлення декомпіляції коду прошивок розумних пристроїв.

Надано рекомендації щодо проведення аналізу оцінки впливу конфіденційності PIA.

Розроблено код Blockchain мережі для розгортання в хмарному середовищі в якій використовується віртуальний прототип симулятор IoT пристрою. Кожна відправка даних з цього пристрою достовірна завдяки використанню даної технології.

Розроблено прототип програмно визначеної мережі в Cisco Packet Tracer на основі Cisco SDN контролеру.

Практичне значення роботи полягає у створенні коду приватної Blockchain мережі для розгортанні в хмарному середовищі GCP та створенню мережі SDN.

Результати здійснених у дипломному проекті досліджень можуть бути використані в комерційних цілях.

Галузь використання – забезпечення інформаційної безпеки мереж де використовуються IoT пристрої.

Ключові слова: ІНФОРМАЦІЙНА СИСТЕМА, МЕРЕЖІ, ЕКОСИСТЕМА IOT, BLOCKCHAIN, ПРОГРАМНО ВИЗНАЧЕНА МЕРЕЖА, НАДАНЯ РЕКОМЕНДАЦІЙ, ХМАРНЕ СЕРЕДОВИЩЕ, КОНТРОЛЕР ПРОГРАМНО ВИЗНАЧЕНОЇ МЕРЕЖІ, УНЕМОЖЛИВЛЕННЯ ДЕКОМПІЛЯЦІ КОДУ, ОЦІНКА ВПЛИВУ КОНФІДЕНЦІЙНОСТІ PIA.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

IoT – технології інтернет речей.

IC – інформаційна система.

SDN – програмно визначена мережа.

SDP – програмно визначений периметр.

DPKI – децентралізована інфраструктура приватних ключів.

Hyper Ledger Fabric – ПЗ для створення приватного блокчейну.

Blockchain – розподілена система що зберігає ланцюжок транзакцій.

DApp – децентралізований застосунок Backend.

DDB - децентралізована база даних

PIA - договір про конфіденційну інформацію.

GCP – провайдер хмарних сервісів Google.

MQTT – протокол обміну даними.

LTE – стільникова мережа 4ого покоління.

AMQP - протокол обміну даними між розумними пристроями.

ВСТУП

Хоча термін «Інтернет речі» IoT є досить новим, ідеї сьогоденного та майбутнього застосування IoT не є новими. Безсумнівно, неймовірний прогрес у цифровій обробці, бездротовому зв'язку, машинному виробництві, датчиках і сенсорах втілює в життя реалістичні уявлення як нашого нинішнього покоління, так і наших предків. Але всі надії та думки про майбутнє використання IoT мають також розглядатись з іншої точки зору, а саме разом з тим фактом, що людська свідомість і поведінка не завжди були і не завжди будуть відповідати хорошим утопічним ідеалам. Завжди буде явна і прихована злочинна діяльність; Завжди знайдуться спекулянти та шахраї, які бажають завдати шкоди іншим і отримувати вигоду з них. Іншими словами, завжди знайдуться люди, мотивовані проникнути всередину якоїсь системи та зламати її, щоб викрасти найцінніше майно або заподіяти збитків.

Втрати користувачів – це прямий прибуток та успіх кібер злочинців. Гірше того, що в сфері IoT ця мотивація може навіть поширюватися на нанесення якихось фізичних ушкоджень або навіть заподіяння невідшкодованих збитків іншим людям. Хакери сьогодні можуть навіть лишити життя людини, яка використовує медичні прилади підключені до інтернету, при не правильному налаштуванні кардіо стимулятора можна заподіяти серйозної шкоди здоров'ю іншої людини; також є вірогідність вивести з ладу IoT систему оповіщення медичного персоналу в лікарні, що призведе до людських жертв.

В цій роботі буде приділено особливу увагу методам та моделям забезпечення інформаційної безпеки в мережах де використовуються IoT пристрої, а також безпеці самих пристроїв безпосередньо.

Безпека Інтернету речей — це не традиційна кібербезпека, а поєднання кібербезпеки з іншими інженерними дисциплінами. Це набагато більше, ніж просто дані, сервери, мережева інфраструктура та інформаційна безпека. Це також включає розподілений моніторинг або контроль стану фізичних систем, підключених через Інтернет.

Таким чином, предметом інформаційної безпеки IoT є не лише застосування єдиного статичного набору правил, які застосовуються до мережевих пристроїв і хостів. Для цього потрібна унікальна програма для кожної системи у яких задіяні пристрої IoT. Будь-що фізичне сьогодні можна підключити до Інтернету за допомогою відповідних електронних інтерфейсів. Таким чином, безпека пристрою IoT є функцією використання пристрою, фізичного процесу чи стану, на який впливає або контролюється пристроєм, і чутливості систем, до яких пристрій підключається.

Актуальність теми даної роботи є прямим результатом сьогоднішнього зростання кібератак на мережі де використовуються значна кількість IoT пристроїв. Також значимість цієї теми сьогодні обумовлена в першу чергу зацікавленістю компаній та корпорацій разом з кіберспеціалістами захистити інформаційні системи та мережі від небажаного впливу зі сторони хакерів. Це допоможе зберегти їхню репутацію та зменшити збитки.

Ця тема також певною мірою пов’язана з іншими темами інформаційних технологій, а саме Big Data та ML. Дані які генерують в дуже великій кількості пристрої IoT, повинні бути збережені в озерах даних в сирому вигляді, а потім вже оброблені механізмами чи так би мовити двигунами обробки даних, після чого завантажені до структурованих баз даних, та проаналізовані алгоритмами та інструментами, які використовують спеціалісти DS. Алгоритми ML навчаються з часом на великих наборах даних. Важлива область досліджень безпеки IoT та ML пов’язана з використанням прикладів, які можуть навчити системи ідентифікувати зловмисні вхідні дані в алгоритми. Наприклад, дослідження показали, що можна дещо змінити зображення, щоб обдурити моделі ML, щоб вони подумали, що щось не те, чим воно є насправді. Впровадження інформаційної безпеки у цей процес на етапі збору даних з IoT пристроїв може допомогти підготувати алгоритми для виявлення та реагування на спроби зловживання доступом та проникнення в системи.

1.ДОСЛІДЖЕННЯ ТА АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ЗАБЕЗПЕЧЕННЯ ІНФОРМАЦІЙНОЇ БЕЗПЕКИ В МЕРЕЖАХ ДЕ ВИКОРИСТОВУЮТЬСЯ ІОТ ПРИСТРОЇ

1.1 Дослідження та аналіз еталонної моделі екосистеми IoT на предмет забезпечення інформаційної безпеки на кожному рівні

Для IoT пристроїв використовується інфраструктура, яку потрібно окремо визначити і відрізняти від звичайної мережі комп'ютерів. Схвалене членами ITU визначення IoT звучить наступним чином: «Глобальна інфраструктура для інформаційного суспільства, що забезпечує розширені послуги шляхом взаємозв'язку фізичних та віртуальних речей на основі існуючих, що розвиваються сумісними інформаційними і комунікаційними технологіями»[2].

Інше визначення дано учасниками IEEE: "Інтернет речі - це мережа, яка з'єднує унікальні ідентифіковані "речі" з Інтернетом. Ці "речі" мають можливості зондування/приведення в дію і потенційну можливість програмування. Завдяки використанню унікальної ідентифікації та зондування, інформація про "річ" може бути зібрана, а стан "речі" може бути змінений з будь-якого місця, в будь-який час, за допомогою будь-чого"[3]

Сьогодні дана технологія поширена в багатьох галузях. Сфери втілення проектів описані в наступній таблиці 1.1 показують охоплення та потенційний вплив IoT на повсякденне життя. Це не тільки споживчі іграшки, які підключені до Інтернету. Системи IoT прогресують у напрямку справжнього покращення добробуту населення та підвищення продуктивності в бізнес-середовищі.

Таблиця 1.1 – Галузі та проекти IoT їх короткий опис

Галузеві проекти IoT	Опис
Розумне землеробство та забезпечення продовольчої безпеки	Системи Інтернету речей дозволяють покращити землеробство та запроваджують нові методи забезпечення продовольчої безпеки та безпеки виготовлення харчових продуктів.
Розумний одяг	Системи Інтернету речей стають інтегрованими в наше повсякденне життя завдяки інтеграції з носієм, таким як одяг, годинники та пристрої, що кріпляться на тілі.

Розумні міста	Системи IoT надають розумні послуги для громадян, зокрема для транспорту, енергетики, охорони здоров'я, освітлення, води та відходів.
Розумне управління водою	Системи Інтернету речей забезпечують більш ефективні можливості управління водними ресурсами, зберігаючи наше водопостачання безпечним і доступним.
Розумне виробництво	Системи IoT, такі як промислова робототехніка та підключені до інтернету фабрики, підвищують продуктивність і якість на виробничих підприємствах.
Розумна енергетика	Системи IoT підтримують оптимізацію енергоспоживання. Включаючи станції з відновлюваних джерел енергії, електромережі, підстанції, диспетчерські, додатки для реагування на попит і зарядку електронних автомобілів EV

Вплив IoT на трансформацію галузевих можливостей є значним. Стає зрозуміло, що коли людство починає покладатися на ці технологічні вдосконалення, вплив відмови або підробки надання цих послуг стає суттєвим. Кожна з цих систем має бути розроблена з урахуванням безпеки та стійкості.

Для забезпечення цих цілей безпеки всесвітнім форумом IoT було розроблено еталонну модель, яка описує сім рівнів екосистеми IoT. Далі в цьому розділі буде розглянуто кілька рівнів екосистем, зокрема самі пристрої та їхнє підключення. Всі рівні еталонної моделі представлені наступним списком:

- Фізичні пристрої та контролери;
- Підключення;
- Граничні обчислення;
- Накопичення даних;
- Абстракція даних;
- Застосунок чи програма керування;
- Співпраця та аналіз.

1.1.1 Рівень фізичних пристроїв та контролерів

В IoT існує так багато різних фізичних типів речей, що стає важко визначити рекомендації щодо безпеки для розробки будь-якої конкретної системи. Однак за своєю суттю пристрої IoT базуються на апаратному забезпеченні та містять датчики та засоби зв’язку. Серед популярних плат розробки IoT є Arduino, Beagle Board, Pinocchio, Raspberry Pi та Cubieboard тощо. Ці плати розробки використовуються для створення прототипів рішень IoT. Вони включають мікроконтролери MCU, які служать мозком пристрою, забезпечують пам’ять і низку цифрових і аналогових контактів вводу/виводу загального призначення GPIO. Ці плати можна модульно складати з іншими платами, щоб забезпечити комунікаційні можливості для нових датчиків. Ці датчики часто підключаються до MCU для локальної обробки, реагування на активацію та/або передачі на інші системи.

Пристрої IoT часто використовують операційну систему реального часу ОСРЧ для керування процесами та пам’яттю, а також служби, що підтримують обмін повідомленнями та інші комунікації. Вибір кожної ОСРЧ базується на необхідних вимогах до продуктивності, безпеки та функціональності продукту. Існує багато доступних ОСРЧ, найпоширенішими серед них є:

- Contiki;
- Lite-OS;
- FreeRTOS;
- SapphireOS;
- uCLinux;
- LynxOS;
- Windows 10 IoT;
- UbuntuCore.

1.1.2 Підключення розумних пристроїв IoT-екосистеми

В свою чергу, наскрізне з'єднання між периферійними пристроями та хмарними або звичайними веб-сервісами може забезпечуватися серією фізичних і хмарних шлюзів, кожен з яких агрегує велику кількість даних. Dell, Intel та інші компанії продають шлюзи IoT. Такі компанії, як Systech, пропонують багато протокольні шлюзи, які дозволяють з'єднувати разом багато типів пристроїв IoT за допомогою кількох антен і приймачів. Існують також орієнтовані на звичайного споживача шлюзи, також звані концентраторами, доступні на комерційному ринку.

Також варто згадати й про інтеграційні платформи та рішення IoT. Ці платформи надають API, які розробники пристроїв IoT можуть використовувати для створення нових функцій і послуг. Розробники IoT дедалі частіше впроваджують ці API і демонструють легкість інтеграції в IT-середовище підприємства. Наприклад, ThingSpeak API можна використовувати для інтеграції пристроїв IoT через HTTP-зв'язок. Це дає змогу організаціям отримувати дані зі своїх датчиків, аналізувати ці дані та вживати заходів щодо цих даних. У міру розвитку IoT різні компоненти, протоколи та API продовжуватимуть об'єднуватися разом для створення потужних корпоративних систем. Ці тенденції викликають питання про те, наскільки безпечними будуть ці системи.

Для рівня підключення IoT існує багато конкуруючих стандартів зв'язку та обміну повідомленнями, які можна використовувати в мережах. Транспортні протоколи TCP і UDP мають місце в системі IoT. Наприклад, REST базується на TCP, а MQTT розроблено для роботи з TCP. Однак потреба підтримувати мережі та пристрої з тимчасовими обмеженнями та пропускною здатністю призвела до переходу від TCP до використання UDP. Наприклад, MQTT-SN — це адаптована версія MQTT, яка працює з UDP. Інші протоколи, такі як CoAP, також добре працюють з UDP.

І IPv4, і IPv6 відіграють певну роль у різних місцях багатьох систем IoT. Спеціалізовані стеки протоколів, такі як IPv6 6LoWPAN, підтримують

використання IPv6 у мережевих обмежених середовищах, у яких працюють багато пристроїв IoT. Крім того, 6LoWPan був розроблений для підтримки бездротового підключення до Інтернету з нижчою швидкістю передачі даних для пристроїв з дуже обмеженим форм-фактором. На додаток до цього, 6LoWPAN базується на специфікації 802.15.4 бездротових персональних мереж низької швидкості LRWPAN, щоб створити рівень адаптації, який підтримує використання IPv6. Рівень адаптації надає функції, які включають стиснення заголовків IPv6 і UDP і підтримку фрагментації, що дозволяє підтримувати датчики для різних цілей, включаючи автоматизацію для будівель і безпеку. Використовуючи 6LoWPAN, розробники можуть скористатися перевагами шифрування каналів, запропонованого в IEEE 802.15. і можуть застосовувати шифрування транспортного рівня, таке як DTLS. Протоколи радіочастот RF, такі як Bluetooth Low Energy BLE, ZWave та ZigBee, підтримують зв’язок між пристроями IoT або зі шлюзами, які потім використовують такі протоколи, як LTE або Ethernet, для зв’язку з хмарою.

IEEE 802.15.4 IEEE 802.15.4 відіграє важливу роль як фізичний рівень і рівень каналу передачі даних для інших протоколів IoT, включаючи ZigBee, 6LoWPAN, WirelessHART і Thread. По суті, 802.15.4 розроблений для роботи з використанням топології «точка-точка» або «зірка» і ідеально підходить для використання в середовищах з низьким енергоспоживанням або низькою швидкістю. Крім того, пристрої 802.15.4 працюють у діапазонах частот 915 МГц і 2,4 ГГц, підтримують швидкість передачі даних до 250 Кбіт/с і діапазон зв'язку приблизно 10 метрів. Фізичний рівень відповідає за керування доступом до радіочастотної мережі, тоді як рівень MAC відповідає за керування передачею та отриманням кадрів на каналі передачі даних.

Стільниковий зв'язок LTE, який часто називають стільниковим зв’язком 4G, є популярним варіантом підключення до Інтернету речей. У типовій мережі LTE таке обладнання користувача UE, як-от смартфон або пристрій IoT, містить USIM-карту, яка безпечно зберігає інформацію автентифікації. Інформація автентифікації, що зберігається в USIM, дозволяє автентифікацію за допомогою

центру автентифікації оператора AuC. Симетричний попередній спільний ключ надається як для USIM під час виготовлення, так і для AuC під час підписки, який потім використовує цей симетричний ключ для отримання об’єкта керування безпекою доступу ASME. ASME використовується для отримання додаткових ключів, які шифрують повідомлення користувачів. Сьогоденний стандарт 5G в розвинутих країнах може запропонувати додаткові варіанти розгортання систем IoT. Він має вищу пропускну здатність та підтримує набагато більше з’єднань. Це може надати розширені можливості для прямого підключення пристроїв IoT. Як один з варіантів до хмари та дозволить створювати нові децентралізовані функції контролера, які підтримують безліч територіально розподілених датчиків та пристроїв. Надійніші можливості стільникового зв’язку дозволять хмарі стати точкою агрегації для каналів даних датчиків, взаємодії веб-служб та інтерфейсів для багатьох корпоративних програм.

Загалом існує багато протоколів зв’язку, які використовуються пристроями IoT. Нижче в таблиці 1.2 наведено підсумовуючу характеристику безпеки деяких із цих протоколів.

Таблиця 1.2 – Підсумовуюча характеристика безпеки протоколів 2ого рівня еталонної моделі екосистеми IoT

Протокол зв'язку	**Характеристика безпеки**
GPRS	Усі дані та сигнали зашифровані за допомогою алгоритму шифрування GPRS GEA. SIM-карти використовуються для зберігання ідентифікаційних даних і ключів.
GSM	Стільникова технологія множинного доступу з тимчасовим розподілом TDMA. SIM-карти використовуються для зберігання ідентифікаційних даних і ключів.
UMTS	Сигнал та дані користувача шифруються за допомогою 128-бітного ключа та алгоритму KASUMI.
CDMA	Стільникова технологія множинного доступу з кодовим розділенням. SIM-карти не використовуються.
LoRaWAN	Підтримує швидкість передачі даних від 0,3 Кбіт/с до 50 Кбіт/с. Мережі LoRAWAN використовують три ключі: унікальний мережевий ключ,

	унікальний ключ програми для наскрізної безпеки та ключ для конкретного пристрою.
6LoWPAN	Малопотужна бездротова персональна мережа PAN, призначена для підтримки автоматичного приєднання пристроїв до мережі за допомогою сервера завантаження LoWPAN для надання інформації про завантаження пристроям 6LoPAN. Мережа 6LoWPAN включає сервер автентифікації, що підтримує такі механізми, як EAP. Сервер Bootstrap також можна налаштувати за допомогою чорного списку пристроїв.
ZigBee	ZigBee використовує 802.15.4 для фізичного рівня та рівня керування доступом до середовища MAC. Мережі ZigBee можуть бути налаштовані за зіркоподібною, деревоподібною та сітчастою топологіями. Сервіси безпеки ZigBee забезпечують встановлення ключів, транспортування ключів, захист кадрів і керування пристроями.
NFC	Обмежені можливості забезпечення захисту безпеки. Часто використовується разом з іншим протоколом. Підтримка лише малого радіусу дії.
802.11	Стандартні бездротові технології, які використовуються в багатьох середовищах. Використовує WPA/WPA2 захист та шифрування.

Ці протоколи забезпечують з’єднання для пристроїв IoT, але такі протоколи, як MQTT, CoAP, DDP, AMQP XMPP, працюють поверх цих протоколів зв’язку нижчого рівня та надають можливість як клієнтам, так і серверам ефективно погоджувати дані для обміну. Комунікації REST також можна дуже ефективно запускати в багатьох системах IoT. На момент написання даної кваліфікаційної роботи REST і MQTT є популярними варіантами для систем IoT.

MQTT — це модель публікації/підписки, за якою клієнти підписуються на теми та підтримують постійне TCP-з’єднання з посередником системи повідомлень. Коли нові повідомлення надсилаються брокеру, вони включають тему до повідомлення, що дозволяє брокеру визначити, яким клієнтам слід отримати повідомлення. Повідомлення надсилаються клієнтам через постійне з’єднання. Загальна схема роботи даного протоколу зображена на рисунку 1.1.

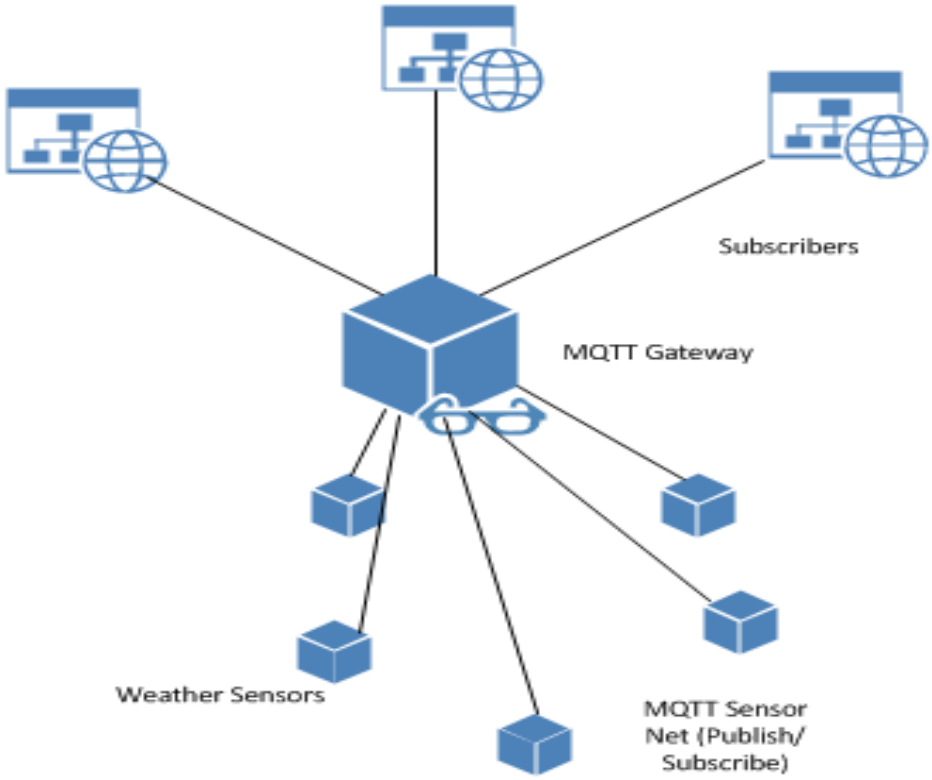


Рисунок 1.1 – Схема роботи MQTT протоколу

MQTT-SN оптимізовано для використання з пристроями, що працюють від батарейок та акумуляторів і мають обмежені ресурси обробки та зберігання. Це дозволяє датчикам і виконавчим механізмам використовувати модель публікації/підписки на додаток до специфікацій ZigBee та подібних радіочастотних протоколів.

CoAP — це ще один протокол обміну повідомленнями IoT, заснований на UDP і призначений для використання в Інтернет-пристроях з обмеженими ресурсами, наприклад бездротових сенсорах. CoAP використовує DTLS для служб безпеки. CoAP складається з набору повідомлень, які легко зіставляються з REST моделлю запитів HTTP: GET, POST, PUT і DELETE. Розумні пристрої за допомогою CoAP зв’язуються з веб-серверами за допомогою певних уніфікованих індикаторів ресурсів URI для обробки команд. Приклади реалізацій із підтримкою CoAP включають інтелектуальні вимикачі світла, у яких комутатор надсилає команду змінити поведінку, стан чи колір кожного індикатора в системі. Схема роботи даного протоколу подано на рисунку 1.2.

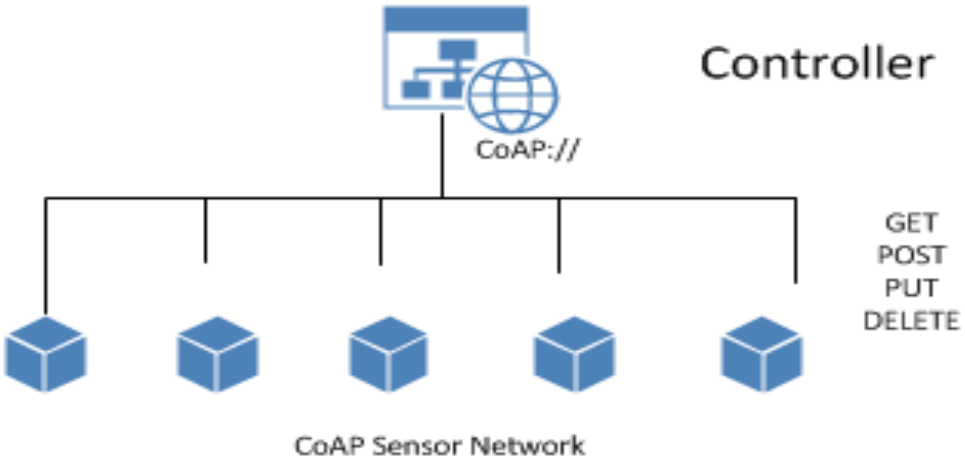


Рисунок 1.2 – Схема роботи CoAP протоколу

DDS — це шина даних DDP, яка використовується для інтеграції машин з іншими машинами. Подібно до MQTT, цей протокол використовує модель публікації/підписки, щоб читачі підписувалися на теми. DDP дозволяє спілкуватися анонімно й автономно, оскільки зв’язок між кінцевими точками не потрібен. DDS також підтримує механізми якості обслуговування QoS. DDS розроблено в основному для зв’язку між пристроями та використовується в різноманітних сценаріях розгортання, включаючи вітрові електростанції, системи медичної візуалізації та систем стеження. На рисунку 1.3 представлена схема роботи DDP протоколу.

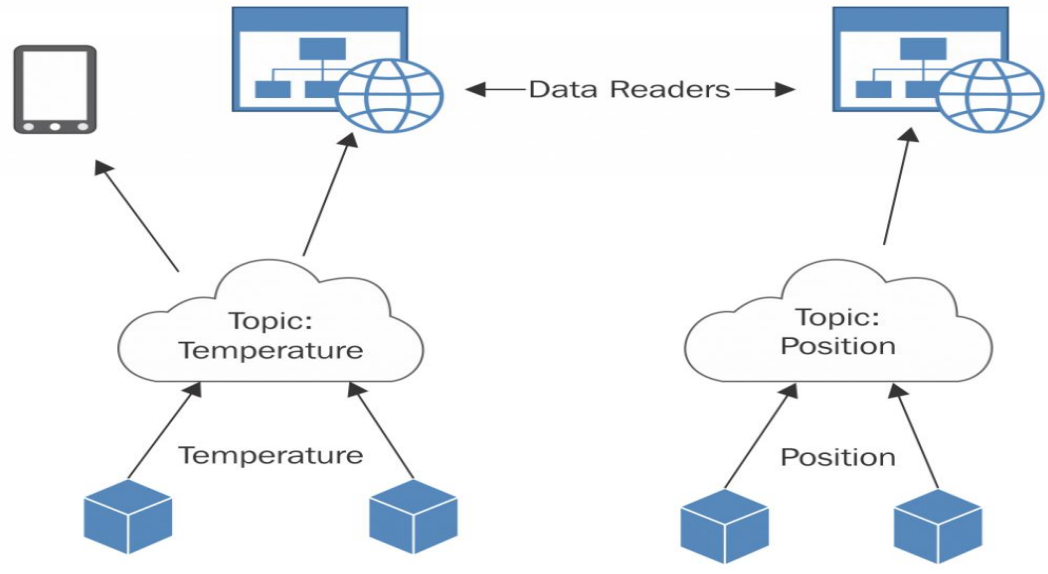


Рисунок 1.3 – Схема роботи DDP протоколу

І останній протокол який варто згадати це AMQP, що був розроблений, щоб забезпечити систему масового обслуговування для підтримки зв’язку між серверами. Застосований до Інтернету речей, він дозволяє як публікувати/підписуватися. Працює на основі топології мереж «точка-точка». Кінцеві точки AMQP IoT прослуховують повідомлення в кожній черзі. AMQP використовується в багатьох секторах, таких як транспорт, де телеметричні пристрої транспортних засобів надають дані аналітичним системам для обробки майже в реальному часі.

1.1.3 Рівень граничних обчислень та збір даних з датчиків, абстракція даних, відстеження походження даних

Рівні збору даних з пристроїв IoT та абстракції даних еталонної моделі відіграють не менш важливу роль в екосистемі розумних пристоїв. Пристрої IoT генерують гори даних, які повинні бути зібрані, агреговані та оброблені аналітичними системами. Дані, зібрані з датчиків, можуть зберігатися як необроблені дані на периферії та агрегуватися в сховищі в межах баз даних і хмари. Дані можуть існувати в різних форматах, включаючи текстові файли, електронні таблиці, файли журналів і звичайно у реляційних базах даних і базах даних NoSQL. Для віддаленого збору даних можна використовувати такі інструменти, як REST, WebSockets, XML і JSON. Розробляючи архітектуру безпеки на цьому рівні, варто подумати про те, як перевірити джерело даних, чи були зловмисні дані введені в потоки даних і чи не було даних підроблено на будь-якому етапі життєвого циклу.

Попередня обробка даних, зібраних за допомогою IoT, часто відбувається на межі, де застосовується початковий фільтр, залишаючи лише відфільтровані дані для передачі в аналітичну систему даних у хмару. Потім дані очищаються та видаляються дублікати. Потім чисті дані вводяться в моделі даних, де їх можна отримати в візуалізаціях. Ця обробка також включає класифікацію об'єктів даних. Класифікація може здійснюватися на основі типів та/або чутливості даних.

Додаються метадані, які включають теги, що представляють конфіденційність безпеки та інші атрибути даних або джерел, які зібрали дані. Наприклад, будь-які конфіденційні дані, які потребують захисту конфіденційності, мають бути позначені як такі. На цьому етапі як дані, так і метадані повинні мати цифровий підпис. На рисунку 1.4 відображено процеси збору обробки та візуалізації даних з 3 та 4 рівня еталонної моделі екосистеми IoT.

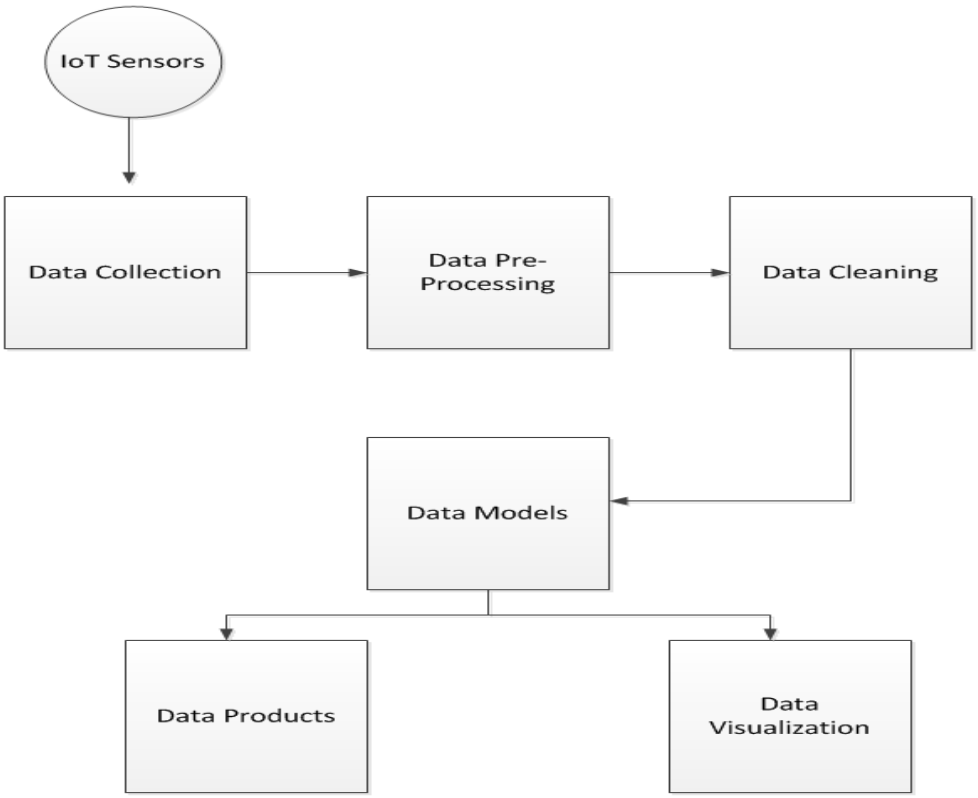


Рисунок 1.4 – Процеси 3 та 4 рівня екосистеми IoT

Ключовим моментом у життєвому циклі даних є необхідність гарантування походження даних. Походження даних відстежує перетворення та дії, застосовані до цих даних протягом певного часу. Інструменти визначення походження даних можуть візуально представляти потоки даних і переміщення в системі. На сьогоднішній день на ринку є багато інструментів для визначення походження даних. Apache Falcon — це один з інструментів із відкритим вихідним кодом, який можна застосовувати до систем Інтернету речей. Ось лише декілька його функцій:

– Встановлює зв’язок між різними даними та елементами обробки в середовищі Hadoop;

- Надає послуги керування каналами;
- Дає можливість легко підключати нові робочі процеси/конвеєри з підтримкою обробки даних із запізненням, політики повторних спроб;
- Дає можливість інтегрувати метасховище чи каталог, такі як Hive/HCatalog;
- Надає сповіщення кінцевому клієнту;
- Збирає інформацію про походження для каналів і процесів.

1.1.4 Програмні застосунки IoT, співпраця та аналіз

Два останні рівні екосистеми еталонної моделі відповідальні за програмне забезпечення, співпрацю та аналіз. Програми, розміщені в хмарі або центрах обробки даних, надають функції, звіти та аналітичні функції для систем IoT. Додатки можуть бути спрямовані на споживача, бізнес, промисловість, охорону здоров'я чи муніципальне обслуговування. Програми також можуть бути орієнтовані на керування, надаючи можливість контролювати, відстежувати та налаштовувати пристрої IoT, як у наведених нижче прикладах:

- Додатки IoT, орієнтовані на споживачів, включають розумні вимикачі та лампочки, підключені термостати, пристрої для відкривання гаражних дверей, переносні пристрої, підключені автомобілі та невеликі безпілотні літальні апарати (дрони);
- Бізнес-додатки IoT включають датчики магазинів, які збирають і аналізують поведінку покупців, щоб робити прогнози, адаптувати маркетинг і персоналізувати досвід споживачів;
- Промислові застосунки IoT включають інтелектуальні виробничі системи, промислові роботизовані системи та прогнозну аналітику для виявлення ймовірних збоїв до їх виникнення та оптимізації технічного обслуговування. Промислові програми IoT також можуть включати розумні промислові системи управління;

- Застосунки IoT для охорони здоров’я можуть включати підключені пристрої, такі як кардіостимулятори, інтелектуальні діагностичні інструменти, підключені лікарні та обладнання;
- Муніципальні додатки IoT включають розумні транспортні системи, підключені паркові системи та розумні сенсорні системи, які збирають екологічну та іншу інформацію.

Архітектура корпоративних систем IoT відносно узгоджена в різних галузях. Архітектори підприємств інтегрують рішення, які включають периферійні пристрої, шлюзи, програми, хмарні служби, протоколи, аналітику даних і сховище. Ця складність створює труднощі для підтримки безпеки IoT і забезпечення того, що окремі екземпляри IoT не можуть використовуватися як опорна точка для атак на інші корпоративні системи та програми. Для цього організації повинні використовувати послуги архітекторів корпоративної безпеки, які можуть дивитися на IoT з точки зору великої та всеосяжної картини. Архітекторам безпеки необхідно буде брати активну участь у процесі проектування, щоб визначити вимоги безпеки, які необхідно відстежувати та виконувати під час розробки та розгортання корпоративної системи IoT. Архітектори корпоративної безпеки виберуть інфраструктуру та компоненти серверної системи, які можна легко масштабувати, щоб підтримувати не лише величезні обсяги даних, створених IoT, але й мати можливість безпечно та ефективно використовувати всі ці дані.

На наступному рисунку 1.5 наведено репрезентативне уявлення про загальну корпоративну систему IoT і різноманітну архітектуру. На цій діаграмі зображено розгортання енергетичного комплексу IoT, підключеного до хмари, а також підключене придорожнє обладнання транспортних засобів, медичне обладнання та датчики моніторингу навколишнього середовища. Це не випадково — як обговорювалося раніше, одна з головних особливостей IoT полягає в тому, що будь-що можна підключити до всього і все до будь-чого. Цілком можливо, що медичний біосенсор підключається до лікарняної системи моніторингу та аналізу

даних і одночасно передає дані про енергоспоживання локальному та дистанційному обладнанню та системам моніторингу енергії.

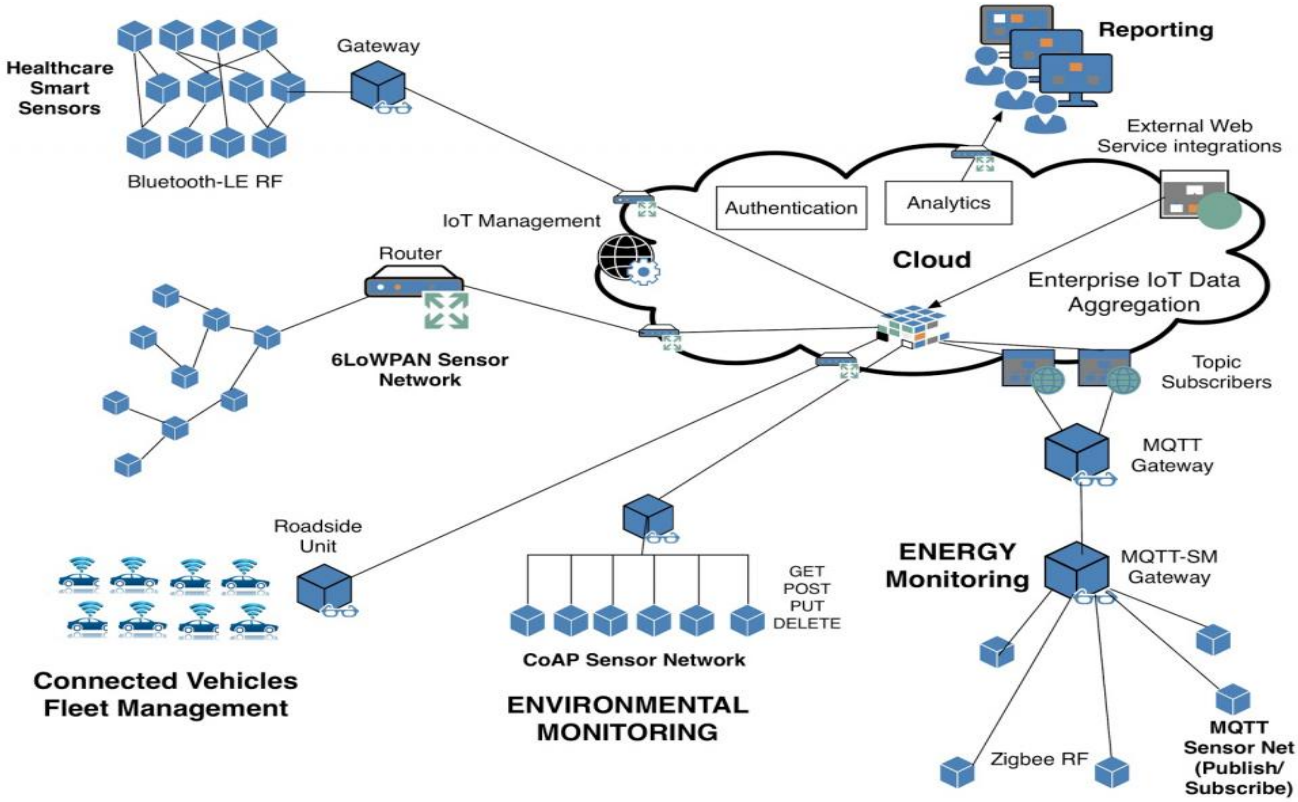


Рисунок 1.5 – Приклад різноманітної екосистеми IoT

Зростаюча кількість точок підключення між різними системами збільшує вірогідність кібер атак на підприємство. Отже, взаємозв'язки системи IoT повинні бути ретельно проаналізовані та оцінені, щоб зрозуміти загрози та необхідні засоби запобігання несанкціонованого доступу.

1.2 Аналіз існуючих підходів інформаційної безпеки IoT-екосистем та моделювання загроз

У цьому пункті даного розділу буде докладно розглядатися існуючі підходи запобігання атак на впровадження та розгортання IoT-екосистем, а саме які загрози можна представити деревом атак на кіберфізичні системи IoT. Потім буде раціоналізовано систематичну методологію для включення контрзаходів для захисту IoT. Буде проаналізовано як типові, так і унікальні вразливості, які можна побачити на різних рівнях стеку технологій IoT. Буде описано нові способи взаємодії електронних і фізичних загроз. Майже неможливо обговорювати

практичні аспекти загроз, вразливостей та ризику без визначення основних компонентів забезпечення інформації ІА, важливого піддомену безпеки Інтернету речей:

- Конфіденційність;
- Цілісність;
- Автентифікація;
- Невідмовність;
- Доступність.

Кожен з компонентів ІА відноситься до IoT, оскільки IoT поєднує інформацію з середовищем пристрою, фізичними властивостями, інформацією, джерелами даних, приймачами та мережами. Однак, окрім основ ІА, ми повинні ввести дві додаткові гарантії, які стосуються кіберфізичних аспектів IoT, а саме стійкість і безпека. Стійкість і техніка безпеки тісно пов’язані. Таким чином, загрози можуть мати різні типи, як природні, так і створені людиною. Торнадо, повені та урагани можна вважати природними загрозами; у цих випадках погода на Землі виступає в якості суб’єкта загрози. Загрози Інтернету речей включають усі загрози забезпечення інформації для даних керування, додатків, датчиків і даних керування, які надсилаються до та з пристроїв Інтернету речей.

Вразливість — це термін, який використовується для визначення слабких місць у конструкції, інтеграції чи роботі системи чи пристрою. Вразливі місця завжди присутні, і щодня виявляється незліченна кількість нових. Зараз багато онлайнових баз даних і веб-порталів надають нам автоматичні оновлення щодо нещодавно виявлених вразливостей. Можна використовувати якісні або кількісні методи оцінки ризику. Простіше кажучи, ризик - це чиясь схильність до втрат. Він відрізняється від уразливості, оскільки залежить від імовірності конкретної події, атаки чи стану та має сильний зв’язок із мотивацією зловмисника. Наприклад, операційна система може мати серйозну вразливість у своїй логіці ізоляції процесу, що дозволяє ненадійному процесу отримати доступ до віртуальної пам’яті іншої програми. Цю вразливість можна використати, і вона, безсумнівно, є слабкою стороною, але якщо система ніколи не підключається прямо чи

опосередковано до ненадійних мереж, уразливість може спричинити невеликий ризик або взагалі не створювати його. З іншого боку, якщо платформа підключена до Інтернету, рівень ризику може підскочити через те, що зловмисник знайде практичний спосіб ін’єкції ворожого коду, який використовує вразливість і дозволяє зловмисникові отримати право власності на машину певний час доки не буде виявлений.

Сфера кібербезпеки є широкою та масивною темою, яка виходить за рамки цієї роботи. Однак корисно розуміти типи атак і експлойтів на основі IoT. Оскільки топологія IoT складається з апаратного забезпечення, мереж, протоколів, сигналів, хмарних компонентів, фреймворків, операційних систем.

1.2.1 Моделювання атак та загроз на кіберфізичні IoT-екосистеми

Цілком можливо, що одна атака на пристрій або програму принесе значну цінність для зловмисника у вигляді зламаної інформації, маніпуляцій із пристроєм. На практиці, однак, атака зазвичай є частиною кампанії згрупованих і послідовних суб-атак або інших дій, кожна з яких ретельно вибирається з різноманітних методів розвідки, наприклад: соціальна інженерія, профілювання, сканування, дослідження в Інтернеті. Кожна діяльність, призначена для досягнення своєї безпосередньої мети, має певний рівень складності, вартості та ймовірності успіху. Дерева атак допомагають моделювати ці характеристики в пристроях і системах.

Створення дерева атак може здатися складним завданням, і важко зрозуміти, з чого почати. Для початку потрібен інструмент для побудови моделі та проведення аналізу з нею. Одним із прикладів є SecurITree, інструмент моделювання дерева атак на основі можливостей, розроблений канадською компанією Amenaza (http://www.amenaza.com/). Створення дерева атак, мабуть, найкраще було б описати на простому прикладі.

Припустимо, що зловмисник бажає досягти головної мети заволодінням керування безпілотними авіаційними системи UAS, тобто керування дронами. На наступному рисунку 1.6 показано приклад дерева атак для досягнення цього.

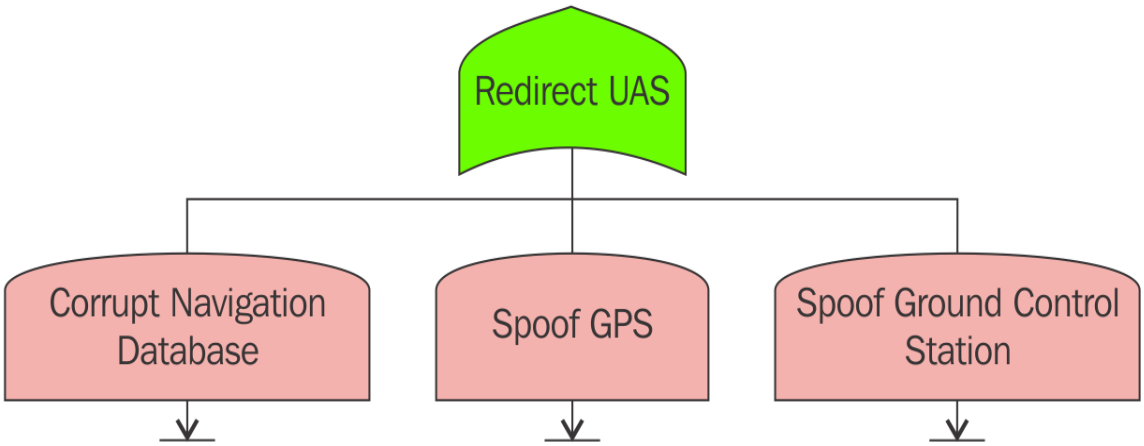


Рисунок 1.6 – Приклад моделювання атаки на root вузол за допомогою дерев атак

Кореневий вузол під назвою Redirect UAS представляє кінцеву ціль і складається з оператора АБО. Це означає, що будь-який з його дітей може задовольнити кінцеву мету. У цьому випадку зловмисник може перенаправити дрон будь-яким із таких методів.

- Пошкодження навігаційної бази даних: навігаційна база даних відображає названі місця розташування в просторі (широта, довгота та, як правило, висота над середнім рівнем моря);
- Підробка інформації GPS: у цьому випадку зловмисник може вибрати активну RF-атаку на базі GPS, під час якої він генерує та передає помилкові дані про час GPS, які дрон інтерпретує як помилкове місцезнаходження;
- Підробка наземної станції управління GCS: У цьому варіанті зловмисник може знайти спосіб підробити законного оператора дрона та спробувати надіслати зловмисні команди маршрутизації.

Тепер потрібно трохи розгорнути дерево атак. Зокрема, потрібно розгорнути цільовий вузол пошкодження навігаційної бази даних. На рисунку 1.7 представлено розширення дерева атак.

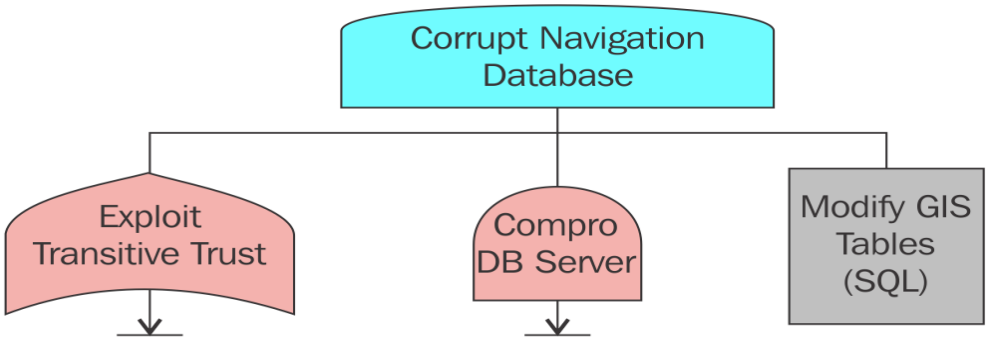


Рисунок 1.7 – Приклад розширеного моделювання атаки на child вузол

Кожен із двох вузлів, Exploit Transitive Trust і Compro DB server, має піддерева. Третій вузол, Modify GIS Tables називається листовим вузлом. Листові вузли представляють фактичні точки входу вектора атаки в модель, тобто дії зловмисника, тоді як його батьківські вузли (вузли І/АБО) представляють або конкретні стани пристрою, стани системи, або цілі, яких зловмисник може досягти завдяки своїй діяльності. Розгорнувши піддерево Exploit Transitive Trust, можна отримати наступну діаграму зображену на рисунку 1.8.

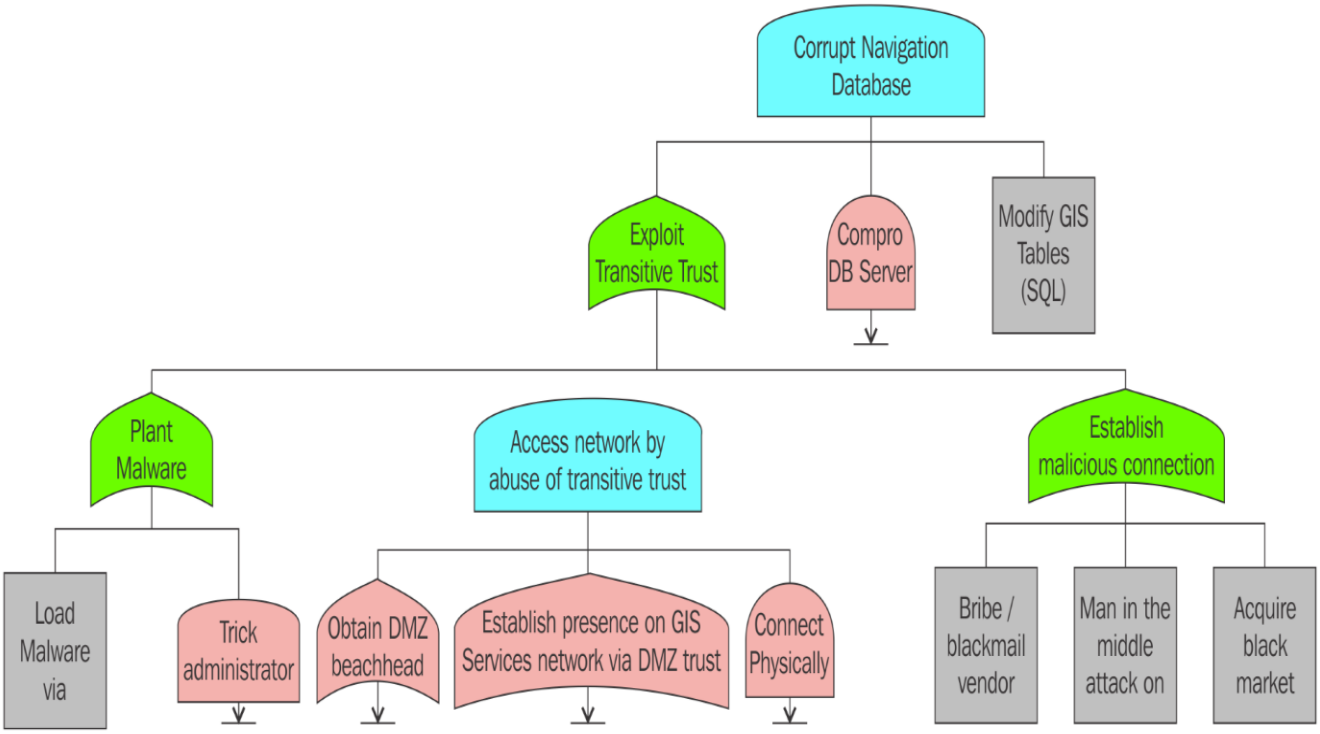


Рисунок 1.8 – Кінцевий приклад побудови моделі дерева атак на кіберфізичну систему IoT

1.2.2 Криптографічні методи захисту зв'язку, захисту вбудованого програмного забезпечення та автентифікації пристроїв IoT екосистеми

Для існуючих криптографічних методів варто навести основні визначення, які відносяться до шифрування та захисту вбудованого ПЗ пристроїв.

Загалом поширене шифрування з симетричним ключем. Тобто коли ключі шифрування та дешифрування ідентичні. RC5, DES, 3DES і AES — це всі форми шифрування з симетричним ключем.. Стандартом для шифрування є Advanced Encryption Standard (AES), який замінив старіші алгоритми DES 1970-х років. AES є частиною специфікації FIPS і стандарту ISO/IEC 18033-3, який використовується в усьому світі. Алгоритми AES використовують фіксовані блоки 128, 192 або 256 біт. Повідомлення, розмір яких перевищує розрядність, розділяють на кілька блоків. AES має чотири основні фази роботи під час шифрування. Довжина ключа AES може бути 128, 192 або 256 біт. Як правило, чим більша довжина ключа, тим кращий захист. Розмір ключа пропорційний кількості циклів ЦП, необхідних для шифрування або дешифрування блоку, тобто 128 біт потребує 10 циклів, 192 біт потребує 12 циклів, а 256 біт потребує 14 циклів. Псевдокод для загального шифрування AES буде приведено на рисунку 1.9.

```
1  state = in
2  w=keyexpansion(key)
3  addroundkey(state, w[0, nb-1])
4  for round = 1 step 1 to nr-1
5   subbytes(state)
6   shiftrows(state)
7   mixcolumns(state)
8   addroundkey(state, w[round*nb, (round+1)*nb-1])
9  end for
10 subbytes(state)
11 shiftrows(state)
12 addroundkey(state, w[nr*nb, (nr+1)*nb-1])
13 out = state
```

Рисунок 1.9 – Представлення алгоритму AES псевдокодом

AES-CCM — це важливий режим шифрування, який використовується для підпису та шифрування даних і використовується в багатьох протоколах,

розглянутих у цій роботі, включаючи Zigbee, Bluetooth Low Energy, TLS 1.2, IPSEC і 802.11 Wi-Fi WPA2. AES-CCM використовує подвійні шифри: CBC і CTR. AES-CTR. Режим лічильника використовується для загального розшифрування потоку зашифрованого тексту, що надходить. Вхідний потік містить зашифрований тег автентифікації. AES-CTR розшифрує тег, а також дані корисного навантаження. З цієї фази алгоритму формується «очікувана мітка». Фаза AES-CBC алгоритму теґує як вхідні дані розшифровані блоки з виходу AES-CTR та оригінальний заголовок кадру. Дані зашифровані; однак єдиними відповідними даними, необхідними для автентифікації, є обчислений тег. Якщо розрахований тег AES-CBC відрізняється від очікуваного тегу AES-CTR, то існує ймовірність того, що дані були підроблені під час передачі. Схему роботи даного режиму шифрування алгоритму AES подано на рисунку 1.10.

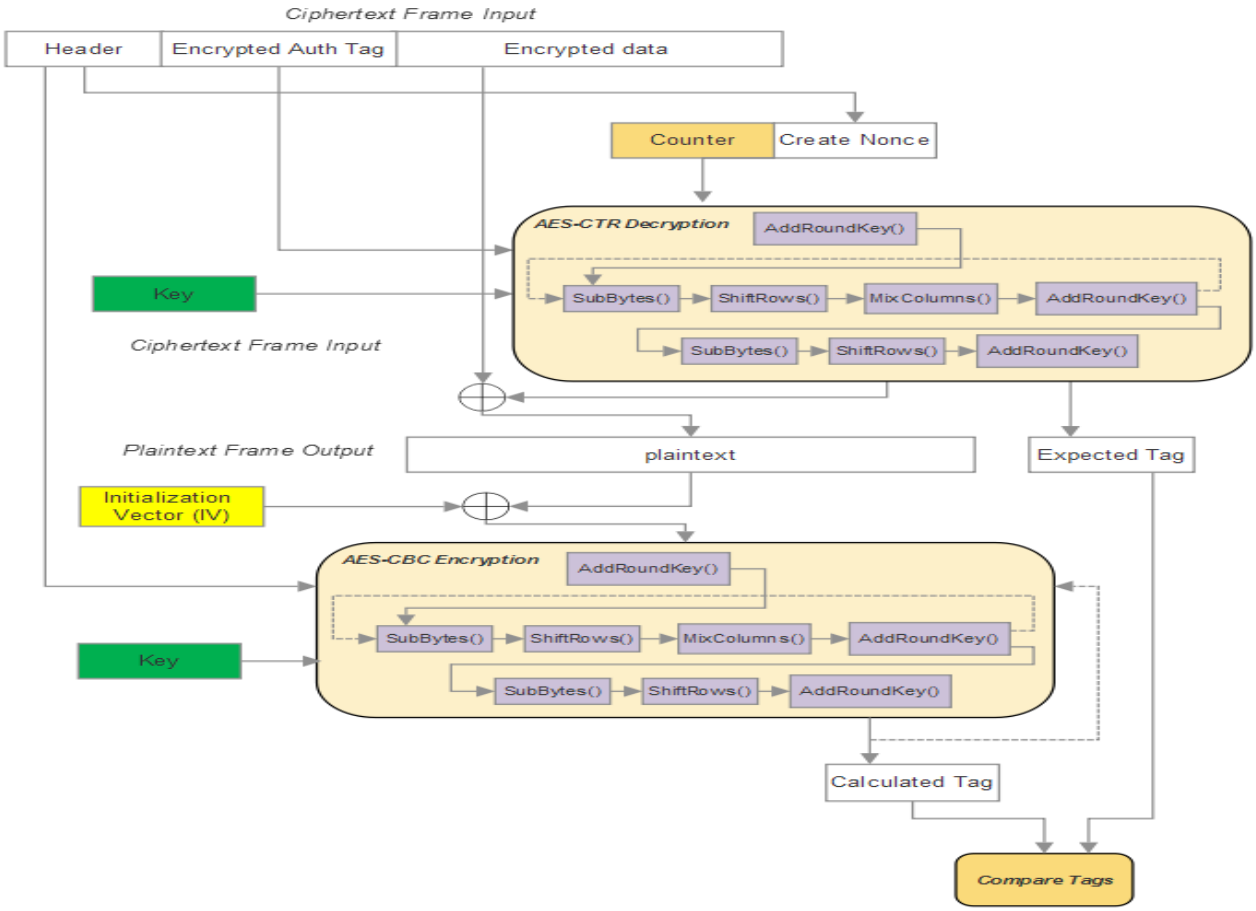


Рисунок 1.10 – Схема режиму AES-CCM

Також для розгортання IoT у топології сітка має бути необхідна кількість AES ключів. Для n вузлів у сітці, яким потрібен двонаправлений зв’язок, має бути n(n-1)/2 ключів чи O(n2).

Також існує спосіб коли шифрування відбувається відкритим ключем, а розшифрування закритим ключем. Відкритий ключ шифрування публікується для всіх, хто може використовувати та шифрувати дані. Лише одержувач має закритий ключ, який використовується для розшифровки повідомлення. Це спосіб також відомий під назвою асиметричного шифрування. Асиметрична криптографія забезпечує конфіденційність даних, аутентифікує учасників. Добре відомі протоколи шифрування в Інтернеті та протоколи повідомлень, такі як PGP, RSA, TLS і S/MIME, Діффі-Хеллмана та еліптичні криві. Потрібно зауважити, що на відміну від кількості симетричних ключів у сітці, де будь-який вузол може спілкуватися з будь-яким іншим вузлом, для методів асиметричної криптографії потрібно лише 2n ключів або O(n).

Перший описаний асиметричний метод шифрування з відкритим ключем — це алгоритм Рівест-Шаміра-Адлемана, або RSA, розроблений у 1978 році. Псевдокод для загального представлення алгоритму поданий на рисунку 1.10.

```
1  int x = 61, int y = 53;
2  int n = x * y;
3  int phi = (x-1)*(y-1);
4  int e = findcoprime(phi);
5  d = (1 mod (phi))/e;
6  public_key = (e=17, n=3233);
7  private_key = (d=2753, n=3233);
8  c = (123^17) % 3233 = 855;
9  p = (855^2753) % 3233 = 123;
```

Рисунок 1.11 – Приклад представлення алгоритму RSA псевдокодом

Ще один з асиметричних методів, який слід розглянути, є шифрування Діффі-Хеллмана. Процес починається з обміну відкритим текстом узгодженого простого числа та генератора простого числа. За допомогою незалежного закритого ключа, створеного Алісою та Бобом, відкриті ключі генеруються та надсилаються по мережі у вигляді відкритого тексту. Це використовується для

створення секретного ключа, який використовується для шифрування та дешифрування. Сильна сторона цього методу безпечного обміну ключами полягає у генеруванні справжнього випадкового числа для кожного закритого ключа. Слабка сторона цього методу полягає в тому, що навіть невелика передбачуваність генератора псевдовипадкових чисел PRNG може призвести до порушення шифрування. Однак основною проблемою є відсутність автентифікації, яка може призвести до атаки MITM. На рисунку 1.12 представлено схематично принцип роботи цього методу.

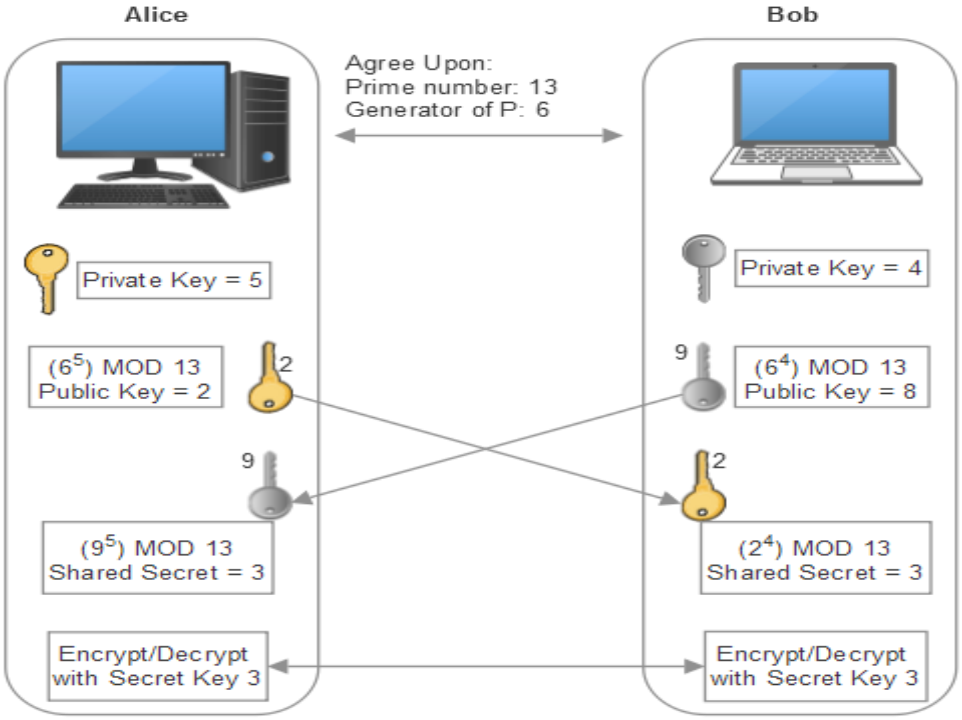


Рисунок 1.12 – Принцип роботи Діффі-Хеллмана

Іншою формою обміну ключами є еліптична крива Діффі-Хеллмана ECDH, яку запропонували Кобліц і Міллер у 1985 році. Вона базується на алгебрі еліптичних кривих над кінцевим полем. NIST схвалив ECDH, а NSA дозволяє використовувати ECDH для надсекретних матеріалів із використанням 384-бітних ключів. Криптографія еліптичної кривої ECC поділяє ці основні принципи щодо властивостей еліптичної кривої. Процес ECC починається з малювання прямої лінії від заданої точки на краю до MAX. Від точки A до B проводиться лінія. Функція точки використовується, щоб провести лінію між двома точками, а потім провести пряму лінію вгору або вниз від нового непозначеного перетину до

відповідної точки на позитивній або негативній осі Y. Цей процес повторюється n разів, де n — розмір ключа. MAX відповідає максимальному значенню на осі x, встановлюючи обмеження на те, наскільки далеко розтягнеться вершина. Якщо випадково вершина більша за MAX, алгоритм примусово встановлює значення, яке виходило б за межу MAX, і встановлює нову точку на відстані x-MAX від початку A. MAX еквівалентний розміру ключа, який використовується. Великий розмір ключа призводить до створення більшої кількості вершин і підвищення рівня секретності. По суті, це обертальна функція. На рисунку 1.13 зображено графік даної функції.

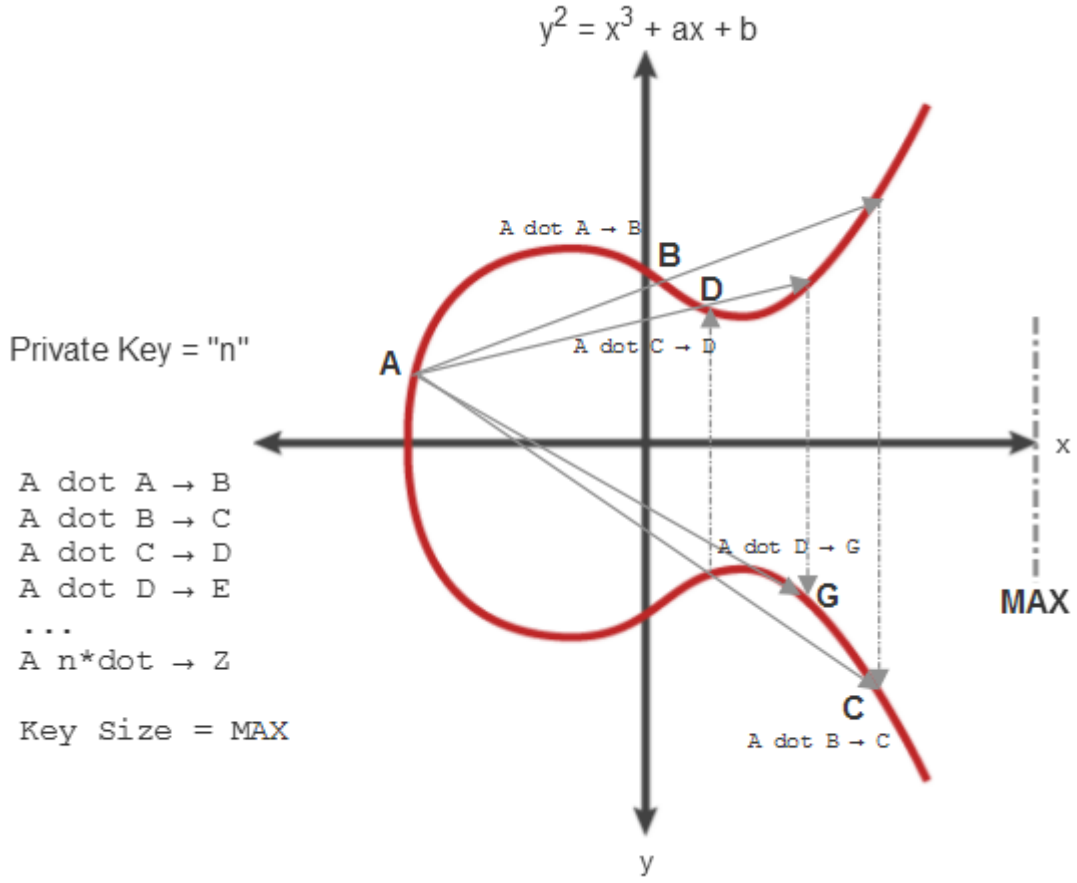


Рисунок 1.13 – Графік функції еліптичних кривих

Еліптичні криві стають переважаючими над RSA. Сучасні браузери можуть підтримувати ECDH, який є кращим методом автентифікації над SSL/TLS. ECDH також можна знайти в блокчейні, і в кількох інших протоколах. RSA тепер використовується, лише якщо сертифікат SSL має відповідний ключ RSA. Інша перевага полягає в тому, що довжина ключів може бути короткою і мати таку

саму криптографічну міцність, як і застарілий метод. Наприклад, 256-бітний ключ у ECC еквівалентний 3072-бітовому ключу в RSA. Важливість цього слід враховувати для обмежених пристроїв IoT.

Функції хешування представляють собою третій тип технології шифрування, який слід розглянути. Зазвичай вони використовуються для створення цифрового підпису. Криптографічний хеш в свою чергу зіставляє дані довільного розміру з бітовим рядком. Ця хеш-функція розроблена як "одностороння". ЇЇ неможливо запустити у зворотному порядку. MD5, SHA1, SHA2 і SHA3 — це всі форми односторонніх хешів. Зазвичай вони використовуються для кодування цифрових підписів, наприклад підписаних зображень мікропрограм, кодів автентифікації повідомлень MAC або автентифікації. Під час шифрування короткого повідомлення, наприклад пароля, вхідні дані можуть бути замалими для ефективного створення справедливого хешу; у такому випадку до пароля додається сіль або неприватний рядок, щоб збільшити ентропію. Сіль є формою ключової похідної функції KDF. На рисунку 1.14 представлено загальну схему симетричних, асиметричних методів та функцій хешування.

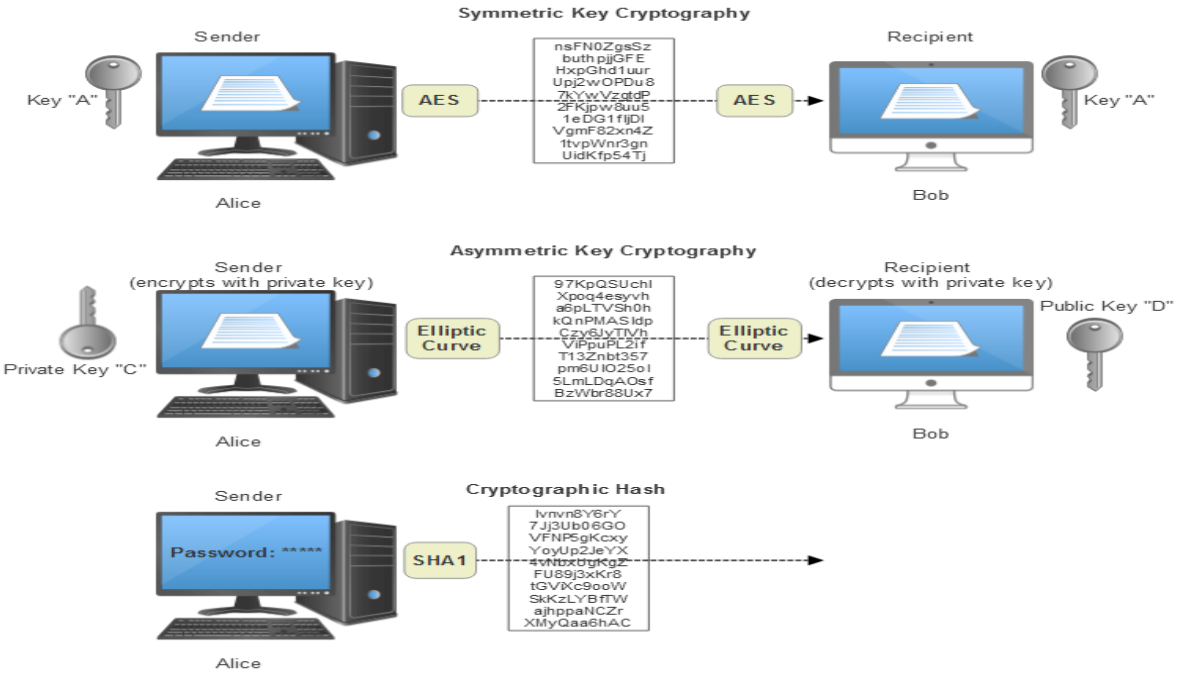


Рисунок 1.14 – Загальне представлення симетричних та симетричних методів разом з функціями хешування

1.2.3 Покращення безпеки IoT екосистем за допомогою технологій Blockchain

Використання блокчейну у сфері IoT дозволяє вирішити проблеми, пов’язані з перевіркою доступу, конфіденційністю та цілісністю даних. Можна впроваджувати рішення на основі блокчейну, щоб гарантувати не лише те, що до даних мають доступ виключно довірені пристрої та прилади, але й щоб дані, якими обмінюються пристрої IoT, не змінювалися.

Останнім часом використання медичних пристроїв, підключених через бездротові мережі, значно зросло. Серед найпопулярніших медичних пристроїв – пульсометри, тонометри, інсулінові помпи та тонометри. Дані, що передаються з цих пристроїв, можуть бути перехоплені зловмисниками шляхом підслуховування бездротових з’єднань. Таким чином, у результаті розголошення конфіденційних медичних даних під загрозою буде не тільки конфіденційність і приватність пацієнтів, але й їхня безпека.

Насправді зловмисник може змінити дані під час передачі за допомогою методів ін’єкції даних. Таким чином, зловмисник вводить шкідливі команди та неправильні значення з наміром атакувати безпеку пацієнта. Як приклад можна взяти інсулінову помпу, яка збільшує дозу інсуліну у відповідь на неправильні значення рівня глюкози в крові.

Навіть у випадку IoT рішення може полягати у впровадженні відповідного смарт-контракту, який керує взаємодією пристроїв IoT і датчиків з медичними даними. Правильне читання медичних даних, переданих пацієнтам, здійснюватиметься за допомогою смарт-контракту, до функцій якого можуть отримати доступ лише перевірені прилади та авторизовані оператори.

Будь-які зміни, внесені до медичних даних, будуть негайно перевірені авторизованими операторами та приладами. Розумний контракт сповіщатиме зацікавлені сторони (медичних операторів і навіть пацієнта) про отримані дані та лікування, яке буде дозволено. Крім того, лише ідентифікатори, пов’язані з подіями, що відбулися, можуть зберігатися в блокчейні, тоді як конфіденційна

інформація не буде зберігатися безпосередньо в блокчейні. Таким чином, конфіденційна інформація про медичні дані безпечно зберігатиметься у зовнішніх базах даних (у автономному режимі), а медичні дані будуть захищені шифруванням відповідно до правил HIPAA. Таким чином, транзакції, збережені в блокчейні, засвідчували б успішну обробку даних, а також могли б використовуватися у разі будь-яких судових спорів.

Блокчейни — це загальнодоступні, цифрові та децентралізовані реєстри або транзакції криптовалюти. Початковим блокчейном криптовалюти був біткойн, але на ринку є понад 700 нових валют, таких як Ethereum, Ripple і Dash. Сила блокчейну полягає в тому, що немає єдиного органу, який контролює стан транзакцій. Це також забезпечує резервування в системі, гарантуючи, що кожен, хто використовує блокчейн, також підтримує копію усіх записів.

Централізована публічна книга — це типовий процес, коли один контролюючий агент веде “публічні книги”. Криптовалюти використовують або децентралізовані, або розподілені книги. На рисунку 1.15 зображено три різні типи “публічних книг”.

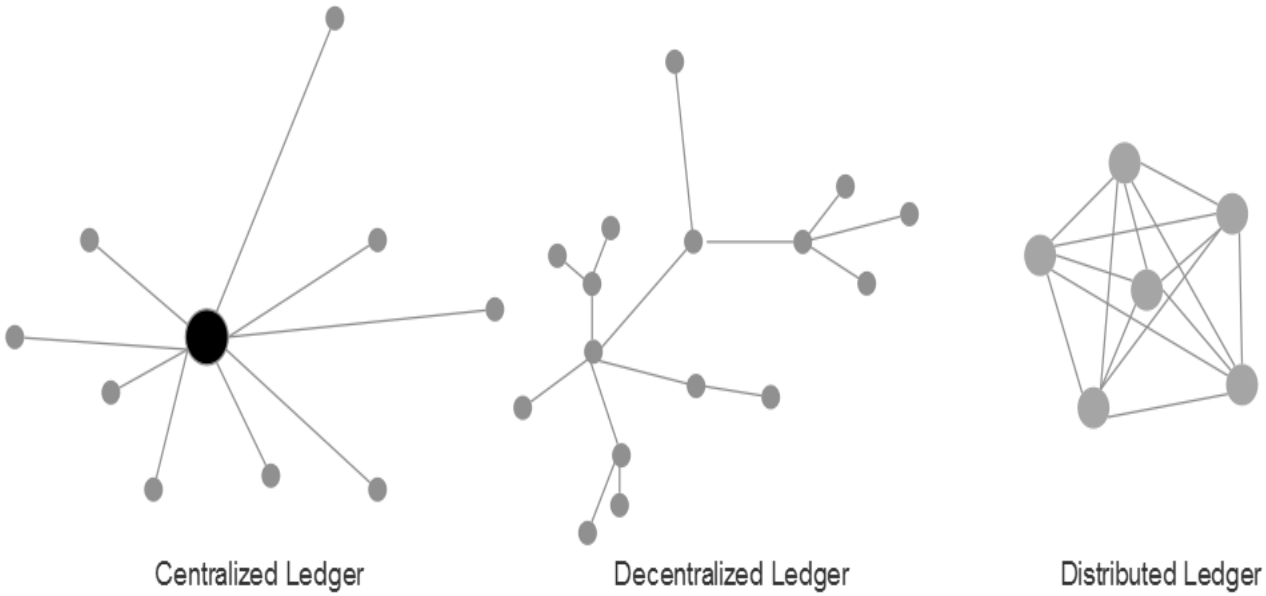


Рисунок 1.15 – Централізовані, децентралізовані та розподілені Ledgers

Завдяки децентралізованій та безпечній архітектурі блокчейн забезпечує цілісність транзакцій, що виконуються. Цілісність даних можна підтримувати шляхом створення незмінних облікових книг виконаних операцій. Блокчейни переважно використовують методи хешування для захисту своїх транзакцій.

У типовій інфраструктурі блокчейну кожна транзакція використовує блок, який хешується для створення хеш-значення. Таким чином дані пов'язуються дуже міцно криптографічними методами, які не можуть бути підроблені або відстеженими несанкціонованими особами. Після того, як ці транзакції будуть підтверджені та перевірені консенсусом, дані, присутні в блоці, стають несприйнятливими до змін.

Нова цікава криптовалюта була розроблена виключно для IoT під назвою IOTA. У цьому випадку самі пристрої IOT є основою довірчої мережі, а архітектура базується на спрямованому ациклічному графі DAG. Біткойн має відповідну комісію за транзакцію, а IOTA не має комісій. Це дуже важливо у світі IoT, який може обслуговувати мікротранзакції. Наприклад, датчик може надавати послугу звітування багатьом абонентам MQTT. Послуга має цінність у сукупному вираженні, але її вимірювано дуже мало на одну транзакцію, настільки малу, що комісія в біткойнах за надання цієї інформації може бути більшою, ніж вартість даних. Архітектура IOTA має наступні аспекти:

- Відсутність централізації;
- Відсутність дорогого обладнання;
- Мікро та нано-транзакції на рівні даних IoT;
- Дані повністю автентифіковані та захищені від підробки;
- Будь-що з невеликим SOC може стати послугою.

Транзакції здійснюються вузлами (пристроями IoT), і вони складають набір у графі DAG. Якщо між транзакцією A та транзакцією B немає спрямованого ребра, але є шлях принаймні довжиною два від A до B, можна сказати, що A опосередковано схвалює B. Коли надходить нова транзакція, вона повинна схвалити або відхилити дві попередні транзакції; це називається прямим схваленням. Це утворює пряме ребро на графіку. Все, що виконує транзакцію,

потрібне для створення «робочого» продукту від імені графа DAG. Ця робота передбачає пошук одноразового номера, який працює з хешем частини схваленої транзакції. Таким чином, за допомогою IOTA мережа стає більш розподіленою та безпечнішою. Транзакція може мати багато схвалених. Чим більше транзакцій відбувається, тим більше впевненість у тому, що транзакція законна.

1.2.4 Програмно визначений периметр SDP поверх програмованих мереж SDN IoT

Програмно-визначений периметр SDP – це підхід до безпеки мережі та зв'язку, де не існує моделі довіри. Він заснований на хмарі Агентства оборонних інформаційних систем DISA. SDP може знешкоджувати такі атаки, як DDOS, MITM, експлойти нульового дня та розвідувальна атака. Разом із забезпеченням накладання та мікросегментації для кожного підключеного пристрою, периметр створює периметр безпеки лише на основі ідентичності навколо користувачів, клієнтів і пристроїв IoT. SDP можна використовувати для створення накладеної мережі, яка є мережею, побудованою поверх іншої мережі SDN. У цьому гібридному мережевому підході площина розподіленого керування залишається незмінною.

Граничні маршрутизатори та віртуальні комутатори керують даними на основі правил площини керування. На одній інфраструктурі можна побудувати кілька накладених мереж. Оскільки SDN залишається постійним майже так само, як дротова мережа, він ідеально підходить для додатків у реальному часі, віддаленого моніторингу та складної обробки подій. Можливість створювати кілька накладених мереж, використовуючи однакові крайові компоненти, дозволяє мікро сегментувати різні ресурси, які мають прямий зв'язок з різними споживачами даних. Кожна пара ресурс-споживач є окремою незмінною мережею, яка може бачити лише за межами свого віртуального накладення, яку вибере адміністратор. Можливість створювати кілька накладених мереж за допомогою тих самих периферійних компонентів дозволяє мікро сегментувати. Теоретично кожен датчик може бути ізольований від іншого. Це потужний

інструмент для забезпечення корпоративного підключення для розгортання IoT, оскільки служби та пристрої можуть бути ізольовані та захищені один від одного. На наступному рисунку 1.16 зображено приклад накладання SDP на SDN.

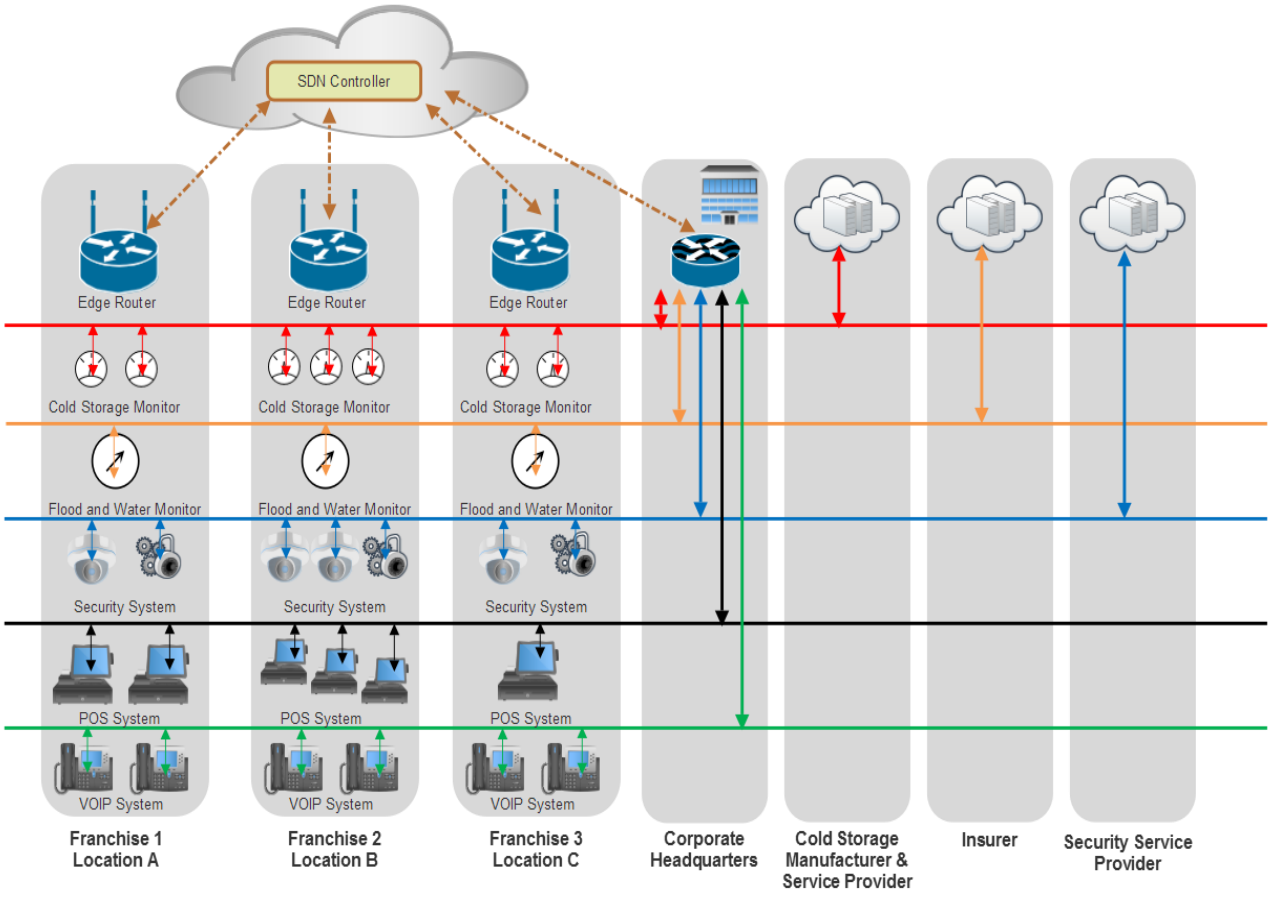


Рисунок 1.16 – Створення програмно визначеного периметру SDP поверх програмованої мережі SDN

Тут у корпорації є три віддалені франшизи з кількома різними пристроями IoT і периферійними пристроями в кожному магазині. Мережа базується на накладеній мережі SDN із мікросегментами, що ізолюють системи POS та VOIP, якими корпоративно керують різноманітні датчики для моніторингу безпеки. Сторонні постачальники послуг можуть керувати різними віддаленими датчиками за допомогою ізольованої та безпечної віртуальної накладної мережі, тобто лише пристроями, які дозволені.

SDP може додатково розширити безпеку, розробивши систему запрошень, яка змушує пару пристроїв спочатку пройти автентифікацію, а потім з’єднатися. До мережі можна додати лише попередньо авторизованого користувача або

клієнта. Якщо користувач приймає запрошення, клієнтські сертифікати та облікові дані поширюються лише на цю систему. Ресурс, який надсилає запрошення, веде облік розширених сертифікатів і забезпечує накладене з'єднання лише тоді, коли обидві сторони підтверджують.

1.3 Постановка завдання

Після розуміння загальних загроз, які становлять небезпеку для екосистем IoT, стає ясно, що для запобігання кібер атак є досить багато засобів та методів захисту. Кібер-злочинці змінюють свою зброю та тактику з огляду на зміни в технологіях і бізнес-процесах, і в той же час спеціалісти кібер безпеки досліджують різноманітні інструменти. Також зрозуміло, що зловмисникам не потрібно бути досвідченим, щоб здійснити злом системи. Завдяки інструментам із відкритим кодом і безкоштовним онлайн-платформам розвідки зловмисники можуть реалізовувати свої плани.

Настав час змінити погляд на забезпечення безпеки кіберфізичних систем IoT. Нескінченна різноманітність зловмисного програмного забезпечення має здатність бути стійкою, прихованою, саморуйнівною та може обходити традиційні системи безпеки, поводячись як законний користувач, що змушує кібер захисників застосовувати новий підхід до забезпечення безпеки. У наступному розділі буде продемонстровано використання різних підходів, які використовують захисники для боротьби з прогресивними загрозами. На конкретних прикладах буде реалізовано методи забезпечення кібер безпеки. Також буде проведена оцінка повноти вирішення поставлених задач. Зокрема, окрема увага буде приділятися рішенням з використання блокчейну та програмно визначеного периметру SDP. Таким чином буде спроба успішно розгорнути PKI, 2FA, для IoT з блокчейном. Буде описали повний процес та розробилено модель, що описує мікро транзакції, і розроблено метод забезпечення відповідальності центрів сертифікації, які поводяться неналежним чином. Також буде спроба

реалізації розгортання DNSChain та DDOS захисту основаних на сучасному блокчейні в мережі де використовуються IoT пристрої.

Висновки до розділу 1

У цьому розділі було розглянуто різні методи та моделі захисту мереж де застосовуються IoT пристрої. На додаток до цього було проаналізовано сучасні тенденції забезпечення безпеки для кожного рівня еталонної моделі екосистем інтернет речей. Було описано всі рівні та методи безпеки, які поширені в сучасних мережах. Також, у цьому розділі було представлено тематичні дослідження, щоб висвітлити кілька основних концепцій безпеки IoT у контексті реального світу. Було надано багато визначень термінам, які пов’язані з даною роботою. Головним завданням цього розділу було визначено ціль даної роботи – запровадження сучасних методів захисту в даному під домені інформаційних технологій. Була проаналізована модель побудови дерева атак на простому прикладі.

Отже, основними методами забезпечення безпеки для екосистем IoT в подальших розділах даної роботи стане використання Blockchain-based методів захисту мереж та мікро транзакцій між розумними пристроями, а також використання програмно визначеного периметру SDP поверх програмно визначених мереж SDN.

2. АНАЛІЗ ЗАСТОСУВАННЯ РІШЕНЬ ОСНОВАНИХ НА ТЕХНОЛОГІЯХ BLOCKCHAIN ТА SDP ПОВЕРХ SDN ДЛЯ ЗАБЕЗПЕЧЕННЯ ІНФОРМАЦІЙНОЇ БЕЗПЕКИ В МЕРЕЖАХ ДЕ ВИКОРИСТОВУЮТСЯ IOT ПРИСТРОЇ

Інтернет – це технологія, якій понад 40 років, і призначена для обміну інформацією через протокол TCP/IP і стек моделей взаємодії відкритих систем OSI. З моменту народження Інтернету кожна нова технологія замінювала існуючу. Інтернет є однією з найпотужніших технологій і достатньо потужний, щоб поширювати ідеї та підключати мільярди IoT девайсів.

TCP/IP був першим пакетом Інтернет-протоколів, створеним для стандартизації зв’язку між мережами; однак модель OSI була розроблена міжнародною організацією зі стандартизації ISO для забезпечення основи для стандартизації зв’язку між системами, незалежно від постачальників, моделей і технологій. Важливо було краще контролювати, які дані використовують клієнти та як вони їх використовують. У моделі клієнт/сервер клієнт керує власними локальними ресурсами, такими як апаратні та програмні компоненти робочої станції чи будь-якого пристрою, тоді як сервер є потужною системою, яка керує спільними ресурсами, такими як апаратне забезпечення, канали мережевого зв’язку та бази даних. На противагу цій моделі, однорангові мережі не мають центрального органу для моніторингу, контролю та примусового виконання команд. Хоча малий бізнес раніше віддавав перевагу цьому типу мереж для своїх внутрішніх потреб та мереж IoT, великі організації завжди уникали використання однорангових мереж через великий ризик втрати контролю над своїми бізнес-операціями та управлінням.

Однак у цій подорожі з’єднання світів є кілька моментів, які переосмислили інновації та полегшили середовище для потреб IoT екосистем кожного бізнесу. Це був блокчейн, який представляв собою однорангову мережу незалежних вузлів для обміну будь-якими даними без участі третіх сторін.

2.1 Застосування рішень основаних на Blockchain для покращення безпеки мереж

Все, що користувачі та пристрої роблять в Інтернеті, проходить в формі IP-пакетів моделі TCP/IP. IP-пакет — це найменша одиниця даних, яку можна надіслати через Інтернет. IP-пакет складається з двох компонентів: IP-заголовка та корисного навантаження. Подібно до IP пакету блок також має два компоненти: заголовок блоку та тіло блоку. Щоб надіслати будь-який тип даних або транзакції, він додає власний цифровий підпис як ідентифікатор джерела та відкритий ключ, який нагадує ідентифікатор адресата в одноранговій мережі.

Таким же чином dApp — це програма, яка працює в одноранговій мережі комп'ютерів та використовує переваги технології Blockchain. Традиційна веб-програма використовує CSS, HTML і JavaScript для відтворення зовнішньої сторінки. Вона отримує дані з бази даних через виклик API. Інтерфейс dApp використовує ту саму техніку для відтворення сторінки, але замість виклику API dApp використовує смарт-контракт, який підключається до блокчейну. Щоб зрозуміти систему в її загальному вигляді, важливо розуміти стани блокчейну:

- Підготовка транзакції: на цьому етапі сторона А створює транзакцію, яка містить інформацію, зокрема публічну адресу одержувача, вихідний цифровий підпис і повідомлення транзакції. Тепер ця транзакція стала доступною для всіх вузлів у блокчейні.
- Перевірка транзакції: вузли блокчейну працюють у моделі без довіри, де кожен вузол отримує цю транзакцію та перевіряє цифровий підпис за допомогою відкритого ключа сторони А. Після успішної перевірки ця автентифікована транзакція зберігається в черзі реєстру та чекає, поки всі вузли успішно перевірять ту саму транзакцію.
- Генерація блоків: транзакції в черзі впорядковуються разом, і один із вузлів мережі створює блок. У блокчейні біткойнів біткойни отримують винагороду, коли біткойн-вузол, також відомий як майнер, створює блок, вирішуючи якусь математично складну задачу.

– Перевірка блоку: після успішного блокування генерації, вузли в мережі обробляються для ітеративного процесу перевірки, де більшість вузлів повинні отримати консенсус. Є чотири популярні способи досягнення консенсусу: Доказ роботи PoW, Доказ стека PoS, Делегований доказ стека DPoS і Практична візантійська відмовостійкість PBFT.
– З'єднання блоків: після успішного механізму консенсусу блоки перевіряються та додаються до блокчейну.

Технологія блокчейн побудована на основі групи існуючих технологій, які в майбутньому будуть широко використовуватися в даній галузі, зокрема в екосистемах IoT. Мережа блокчейну складається з мережі кількох незалежних машин, які називаються вузлами. На відміну від традиційних баз даних, які зберігають всю інформацію на централізованому сервері баз даних, вузли Blockchain зберігають копію всієї бази даних з адміністративною роллю. Навіть якщо один вузол вийде з ладу, інформація залишиться доступною для інших вузлів. У той момент, коли вузол приєднується до мережі блокчейн, він завантажує оновлену книгу реєстр блокчейну. Кожен вузол відповідає за керування та оновлення своєї книги перевіреними блоками. Вузол підтримує реєстр і організовує його у вигляді блоків, пов'язаних з алгоритмом хешування. На рисунку 2.1 схематично зображено дані концепції.

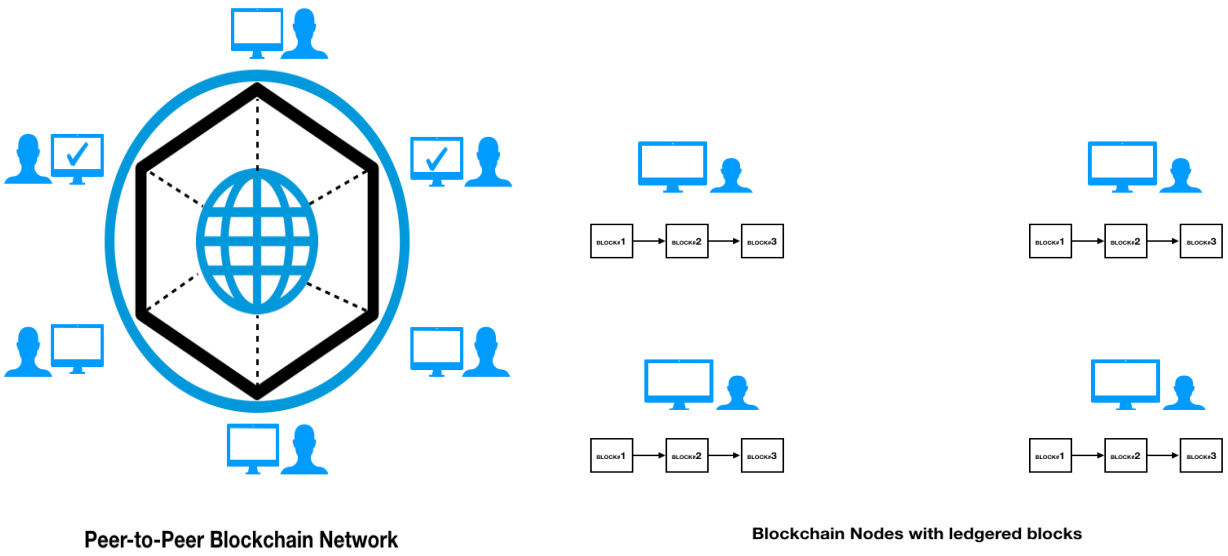


Рисунок 2.1 – Зберігання копії реєстру блокчейну та сруктура книги записів

Кілька транзакцій об’єднуються разом, щоб утворити блок, і в найпростішому вигляді це структура даних. Кожна криптовалюта має власний блокчейн зі своїми налаштованими властивостями. Наприклад, блок у блокчейні Bitcoin генерується кожні 10 хвилин і розмір кожного блоку становить 1 МБ, тоді як блок у блокчейні Ethereum генерується кожні 12-14 секунд, а розмір кожного блоку становить 2 КБ. На рисунку 2.2 представлено структуру блоку.

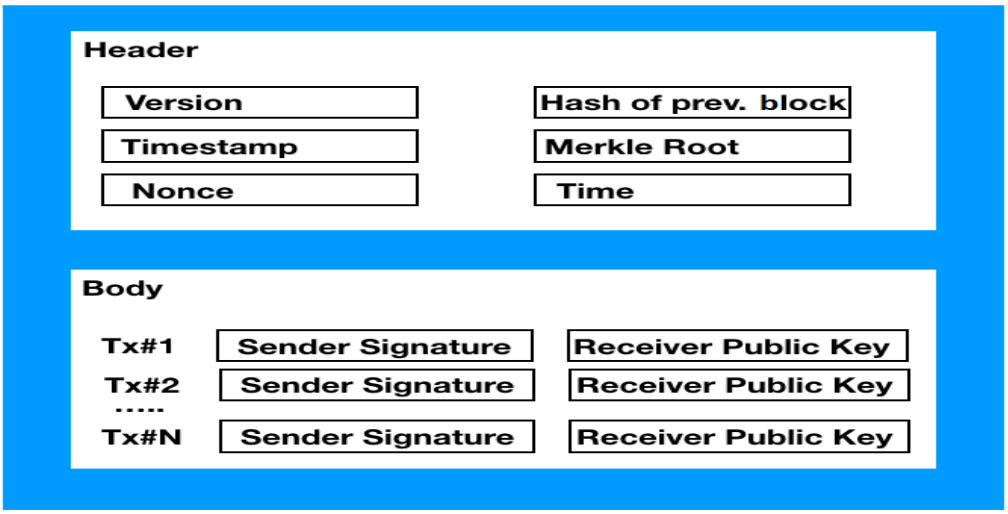


Рисунок 2.2 – Структура блоку

Оиже, блок складається з заголовку та тіла. Заголовок блоку допомагає нам ідентифікувати конкретний блок у ланцюжку блоків. Він містить набір метаданих:

– Версія: це 4-байтове поле, яке використовується для відстеження оцінок програмного забезпечення або протоколу.
– Мітка часу: це 4-байтове поле, яке вказує час створення блоку в секундах. Хеш попереднього блоку: це 32-байтове поле, яке вказує хеш попереднього блоку в ланцюжку.
– Nonce: це 4-байтове поле, яке використовується для відстеження лічильника алгоритму PoW.
– Хеш кореня Merkle: це 32-байтове поле, яке є хешем кореня дерева Merkle блокової транзакції.

Тіло блоку: ця частина блоку складається зі списку транзакцій. У світі біткойнів один блок в середньому складається з понад 500 транзакцій. Кожна

транзакція повинна мати цифровий підпис; інакше він вважається недійсним. Для цього використовується функція хешування, щоб застосувати алгоритм до фактичної транзакції з закритим ключем/секретним ключем.

У світі блокчейну хешування є основою його характеристики незмінності. Процес хешування гарантує, що жоден із блоків у книзі не буде змінено чи підроблено. Замість того, щоб відстежувати деталі кожної транзакції, блокчейн і вузли просто повинні пам’ятати та відстежувати відповідний хеш.

Криптографічне хешування — це спосіб генерувати вихідні дані фіксованої довжини на основі будь-якої заданої довжини вхідного рядка. Вихідні дані називаються хешем повідомлень і призначені для захисту цілісності будь-яких даних, таких як файл, медіа чи текст. Для захисту певного введення або конфіденційної інформації призначено лише один дайджест повідомлення. Невелика зміна вхідних даних призводить до різкої різниці в результатах, що робить майже неможливим передбачити дані в русі чи стані спокою. Існують різні способи створення хешу повідомлення. У світі криптовалют, популярним прикладом якого є біткойн, алгоритм SHA-265 використовується для створення 256-бітного хешу фіксованої довжини повідомлення для кожного блоку.

У блокчейні вузол упорядковує весь реэстр у формі хронологічно пов’язаних блоків. Щоб гарантувати, що реєстр залишається захищеною від втручання, кожен блок робиться залежним від попереднього блоку. Іншими словами, новий блок не може бути створений без хешу попереднього блоку. Перш ніж додавати новий блок у реєстр, це має бути схвалено та перевірено кожним вузлом у блокчейні. Це не дозволяє будь-кому підробляти або змінювати облікову книгу, за винятком хакера, який здатний заразити та скомпрометувати всі мільйони вузлів у блокчейні одночасно. Лише перший блок, який називається блоком генезису, створюється сам і вказує на себе. На наступному рисунку 2.3 показано спрощену структуру блокчейна біткоїну:

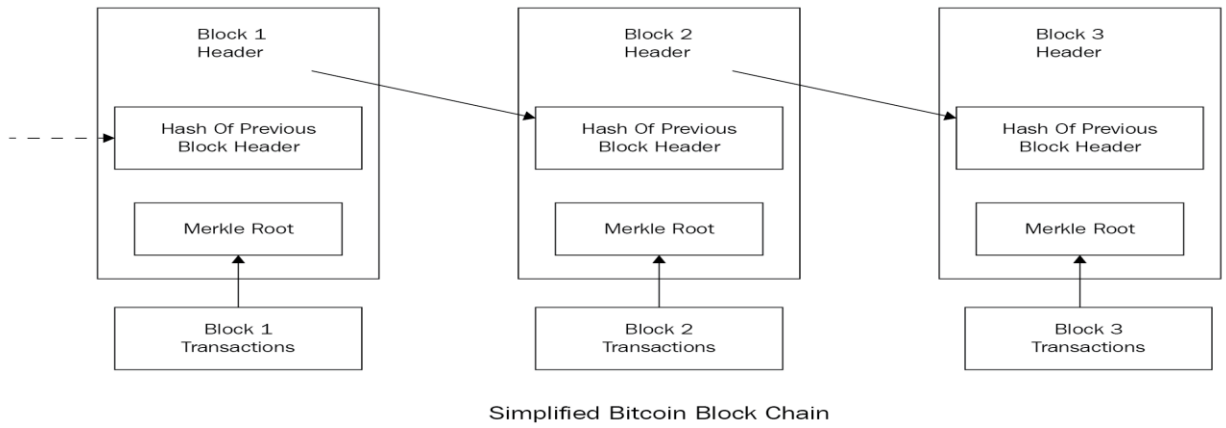


Рисунок 2.3 – Структура блокчейну

Кожен блок вказує на хеш попереднього хеш-блоку, і це стає основою незмінної системи блокчейну. Тепер, навіть якщо блок між ними буде змінено або порушено будь-яким способом, хакер ніколи не зможе взломати той самий блокч, оскільки невелика зміна в блоці може призвести до різкої зміни кінцевого хешу. З тисячами і тисячами транзакцій у кожному блоці стає надзвичайно важко знайти одну транзакцію, яка не займатиме багато часу та не буде залежати від процесу. Щоб уникнути цієї складної роботи, було розроблено комплексне хеш-дерево, яке називається деревом Меркла.

Отже, кіберзлочинці стають усе більш досвідченими, і організації змагаються з ними, щоб захистити важливі активи, такі як комерційні таємниці, інтелектуальна власність та інформація про клієнтів. Кожна організація використовує ті чи інші рішення з кібербезпеки, але щороку втрачаються мільярди доларів. Блокчейн розроблений як децентралізований, незмінний і відстежуваний, і він вирішує більшість проблем безпеки в основі. Ось деякі з його випадків використання в контексті забезпечення безпеки мереж та пристроїв IoT:

- Управління ідентифікацією та доступом
- Захист від DDoS
- Децентралізоване зберігання облікових даних
- Захист від атак "людина посередині" (MITM).
- Приватна інфраструктура ключів PKI з блокчейном
- Аутентифікація за допомогою Blockchain
- Платформа безпеки DNS на основі блокчейну

2.2 Реалізація та розгортання приватного блокчейну на основі рішення Hyperledger Fabric для IoT

Екосистеми IoT створюють низку нових розподілених технологічних моделей і додатків. Потенціал поєднання IoT і блокчейну Hyperledger Fabric все більше й більше починає використовуватись. Сьогодні поєднання цих технологій застосовується в різних сферах та галузях.

Нещодавно запущена норвезька мережа відстеження морепродуктів використовує IBM Blockchain, який побудований на Hyperledger Fabric, для обміну даними ланцюга поставок у всій індустрії морепродуктів Норвегії. Мета полягає в тому, щоб забезпечити споживачів у всьому світі безпечнішими та якіснішими морепродуктами. Мережа використовує датчики IoT, щоб повідомляти про ключові параметри середовища. Починаючи з аквасистеми, датчики відстежуватимуть і повідомлятимуть про умови, необхідні для виробництва високоякісної риби, такі як температура води, рівень кисню та якість води, а коли рибу транспортують, датчики IoT відстежуватимуть і повідомлятимуть про такі фактори, як довжина подорожі, температури та руху в дорозі.

My Sensor – це IoT-пристрій, розроблений Movistar, який виявляє мінімальні концентрації радону в оточенні. Радон — це природний газ, який може накопичуватися у вашій власності в закритих приміщеннях, особливо в підвалі та а нижніх поверхах. Якщо рівень радону перевищує певні порогові значення, це може вплинути на здоров’я людей. Використовуючи модуль TrustOS MQTT Telefonica, My Sensor відстежує та сертифікує історичні дані концентрації газу радону. Користувач може отримати сертифікат із повним доказом, який гарантує, що концентрація не перевищувала порогових значень за певний період або робила це під час n вимірювань поспіль. Датчик надсилає вимірювання через TrustOS до мережі Hyperledger Fabric, де деякі ланцюгові коди відстежують і засвідчують, що показники знаходяться в очікуваному діапазоні або виходять за межі діапазону. Сама мережа засвідчує серію вимірювань, і будь-хто може перевірити це без будь-яких сумнівів, уникаючи участі перевіреного експерта.

В цьому розділі буде продемонстровано наглядно реалізацію та розгортання рішення на основі технології Hyperledger Fabric для забезпечення безпеки мережі в якій використовуються IoT пристрої. Буде розглянуто простий випадок використання та поєднання технологій. Архітектурна діаграма даного рішення представлена на рисунку 2.4.

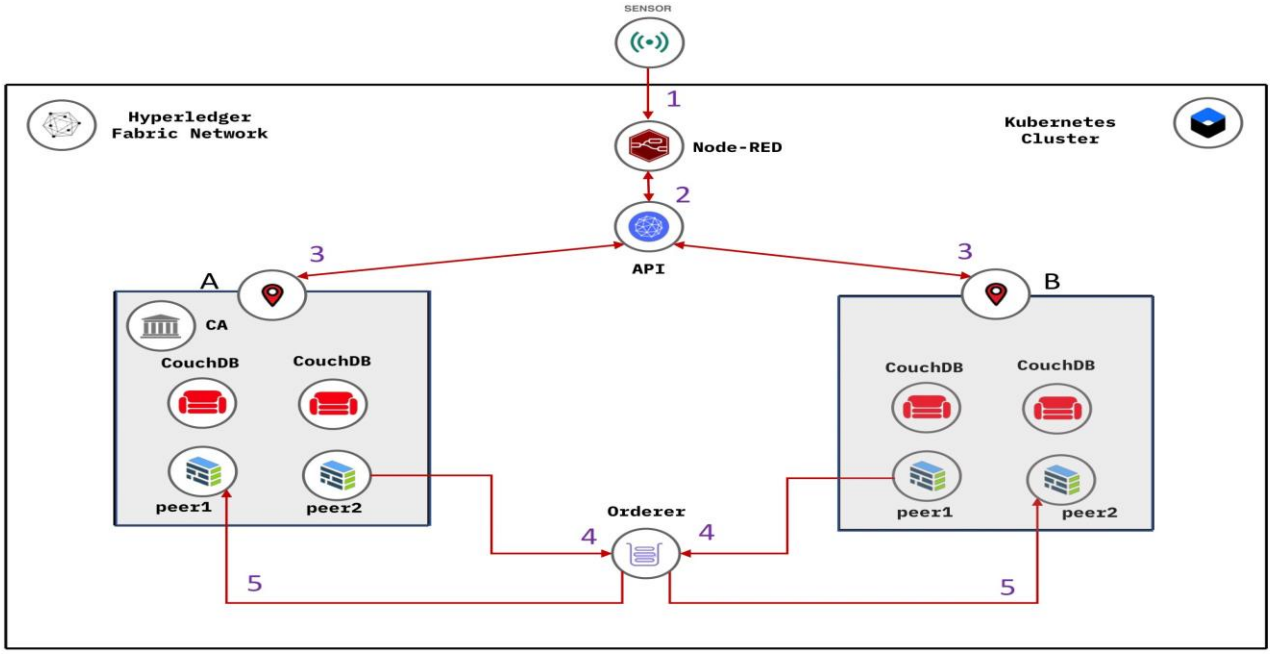


Рисунок 2.4 – Рішення реалізації та розгортання блокчейну Hyperledger Fabric на Kubernetes разом з готовим застосунком Node-RED симуляції даних з IoT пристроїв

Отже, робочий процес рішення зображеного на даній діаграмі може бути описаний кількома кроками:

1. Node-RED отримує вхідні дані MQTT від датчика IoT (або генерує змодельовані дані).
2. Щоб викликати реєстр блокчейну для запису та читання даних, вузли всередині Node-RED виконують HTTP-запити та отримують відповідь від API.
3. API, створений на основі Hyperledger Fabric Client SDK для Node.js, взаємодіє із мережевим кодом у мережі Hyperledger Fabric і оновлює та зчитує реєстр.

4. Peers виконують функції, визначені в ланцюжковому коді відповідно до запиту, і надсилають його замовнику.
5. Замовник створює блоки та надсилає їх назад до Anchor Peers, які транслюватимуть блоки цих Peers.

CouchDB використовуються як розподілена база даних стану транзакцій. CouchDB є абревіатурою від Cluster Of Unreliable Commodity Hardware. CouchDB можна визначити як:

- Розподілений сервер бази даних документів, доступний через RESTful JSON API;
- Ad-hoc і без схеми з плоским адресним простором;
- з надійною поетапною реплікацією з двонаправленим виявленням конфліктів і керуванням ними;
- з можливістю запитів та індексування, оснащений таблично-орієнтованим механізмом звітності, який використовує JavaScript як мову запитів.

Kubernetes — це платформа оркестровки контейнерів, яка доступна в основних хмарних провайдерах, таких як Amazon Web Services, Google Cloud Platform, IBM і Azure. Марсело Фейтоза Один із геніальних хмарних архітекторів IBM, створив і опублікував посібник на GitHub про те, як налаштувати робоче середовище Hyperledger Fabric на Kubernetes.

Node-RED є одним із інструментів FBP, який розроблений командою IBM Emerging Technology Services. Node-RED — це віртуальне середовище, яке революційним чином поєднує апаратні пристрої, API та онлайн-сервіси в браузері. Він надає такі функції:

- Інтерфейс на основі браузера;
- Працює з Node.js і є легким;
- Інкапсулює функцію та може використовуватися як вузол;
- Можна створювати та додавати власні вузли;
- Легкий доступ до хмарних сервісів IBM.

На середовище Kubernetes хмарного провайдера GCP буде встановлено Hyperledger Fabric версії 2.3, а також API сервер та CouchDB. А в наступному

розділі даної магістерської роботи буде визначено та запроваджено покроковий посібник для ведення власної бізнес-мережі на основі даних технологічних рішень.

На рисунку 2.5 показано процес створення NFS сервера та GKE кластера для майбутнього розгортання рішення блокчейн мережі.

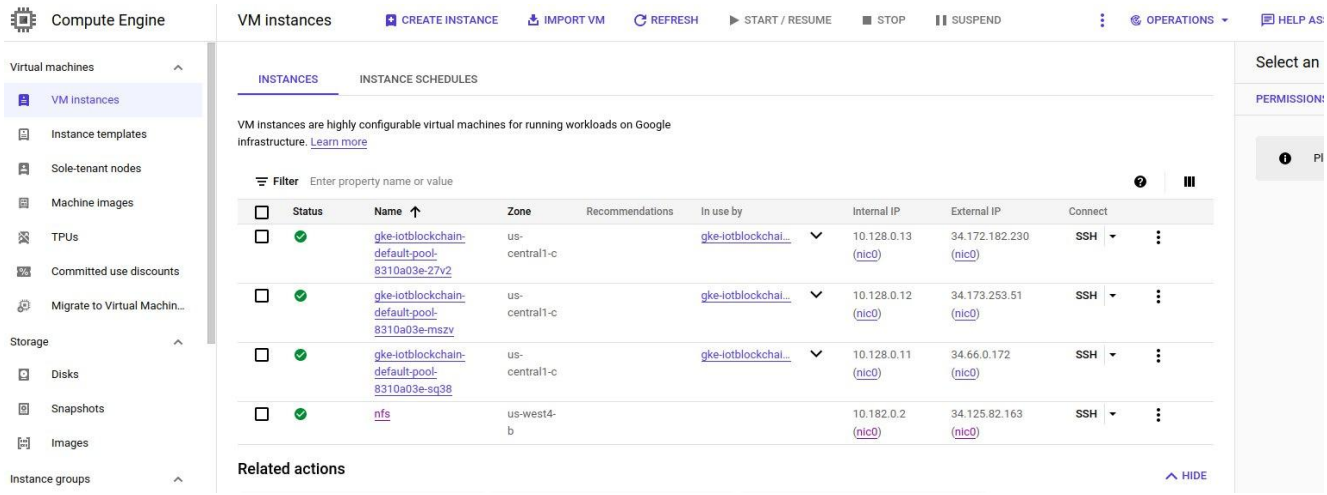


Рисунок 2.5 – Створення NFS сервера та кластеру GKE для розгортання Hyperledger Fabric

Для початку роботи потрібно створити кластер GKE та NFS сервер для того щоб всі поди могли зберігати сертифікати, конфігураційні файли та chaincode ще в одному місці. Таким чином, на випадок відключення кластеру Kubernetes всі дані буде збережено і навіть якщо всі учасники розподіленої мережі будуть відключені. Після успішного створення частини інфраструктури потрібно приступати до створення іншої частини. А саме до наступних кроків:

– Створення доступу до GKE Kubernetes Cluster;
– Розгортання Hyperledger Fabric;
– Розгортання Hyperledger Fabric SDK для Node.js;
– Розгортання Node-RED.

Спочатку потрібно перевірити щоб порти UDP та TCP 111 та 2049 були доступні для NFS з кластеру Kubernetes. Це можна зробити в Firewall rules VPC хмарної платформи GCP. На рисунку 2.6 показано процес налаштування правил мережевого екрану хмарної платформи.

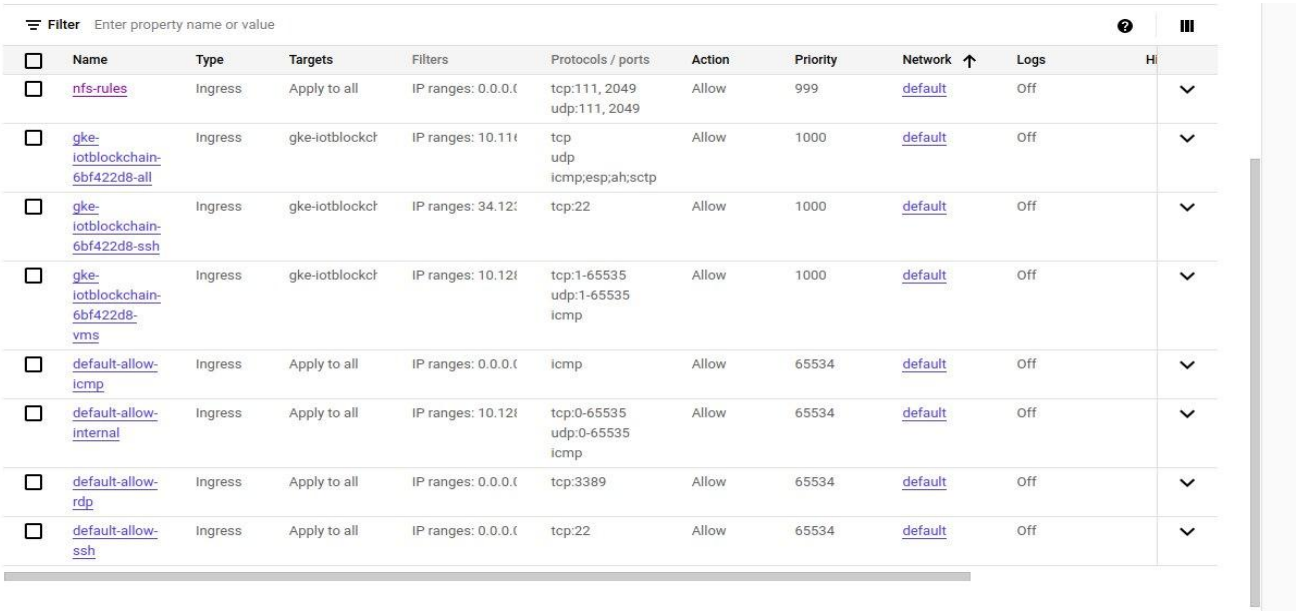


Рисунок 2.6 – Налаштування мережевих правил платформи GCP для доступу кластера Kubernetes до NFS серверу

Після створення усіх нодів кластеру за допомоги команди kubectl get pods можна побачити вузли кластеру, як на малюнку 2.7. Потім потрібно створити PV та PVC для надання можливості Pods використовувати NFS сервер. PV — це частина пам’яті в кластері, яку надав адміністратор. PVC — це запит користувача на зберігання. Це сховище буде використане для зберігання файлів конфігурації та ланцюжкового коду. На рисунку 2.8 продемонстровано файли .yaml для створення PV та PVC.

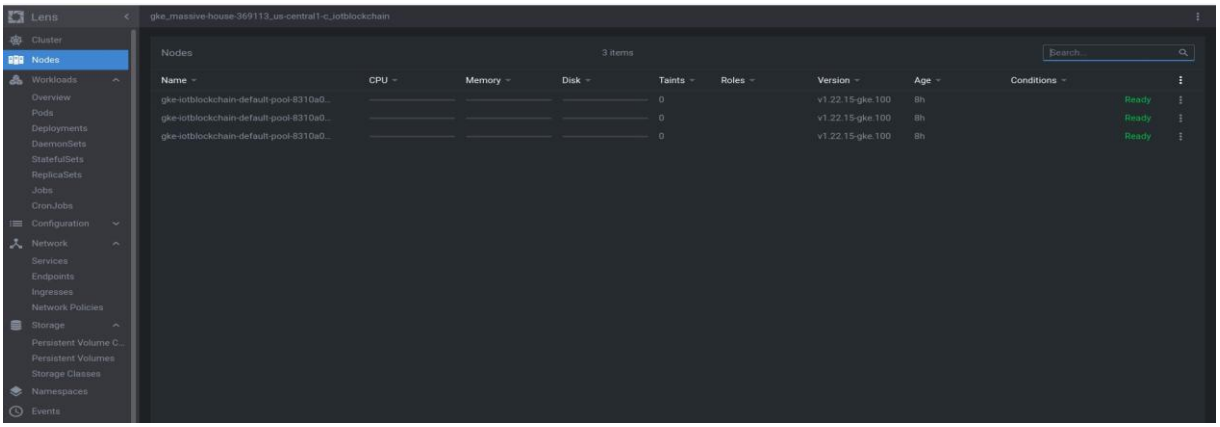


Рисунок 2.7 – kubectl get pods для щойно створеного кластера

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: mypvc
spec:
  storageClassName: standard
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 4050Mi
```

pv.yaml: Блокнот

Файл Редагування Формат Вигляд Довідка

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: mypv
spec:
  storageClassName: standard
  capacity:
    storage: 5Gi
  volumeMode: Filesystem
  accessModes:
    - ReadWriteOnce
  nfs:
    path: /mnt/nfs_share
    server: 10.182.0.2
```

Рисунок 2.8 – Створення PV та PVC за допомогою команди kubectl apply -f .

Після створення PV і PVC наступним кроком буде скопіювати локальні файли коду даного проекту на сервер NFS. Структура файлів даного проекту подана на наступному рисунку 2.9, яка зберігається на NFS сервері.

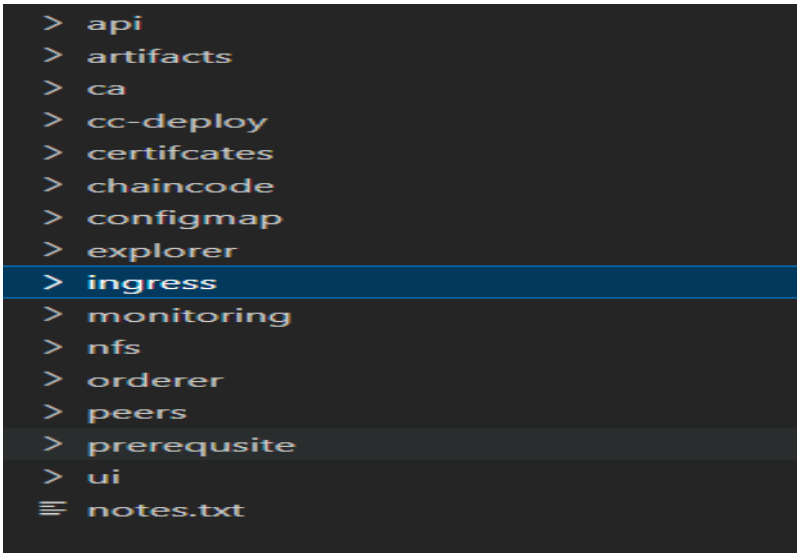


Рисунок 2.9 – Файлова структура проекту

Наступним кроком після виконання всіх попередньо зазначених дій буде створення ключових матеріалів Hyperledger Fabric і пов’язаних з конфігурацією каналів артефактів. Cryptogen — це утиліта для генерації матеріалу ключа Hyperledger Fabric. Він надається як засіб попереднього налаштування мережі з метою тестування. Команда configtxgen дозволяє користувачам створювати та перевіряти пов’язані з конфігурацією каналу артефакти. Для генерації усіх артефактів буде використано ресурс jobs в Kubernetes. В директорії jobs подано

маніфести .yaml для створення та запуску цих ресурсів. На рисунку 2.10 показано структуру даної директорії.

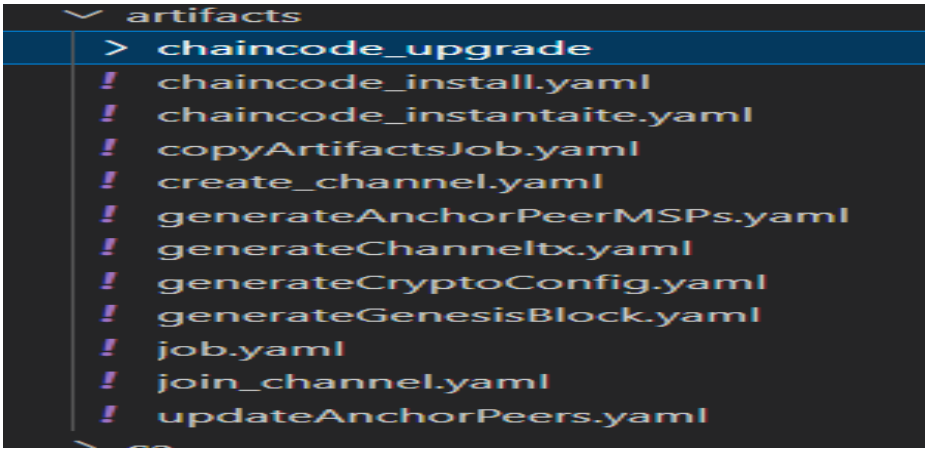


Рисунок 2.10 – Структура директорії artifacts

Потрібно запустити усі ці файли за допомогою команд які подані на наступному рисунку 2.11.

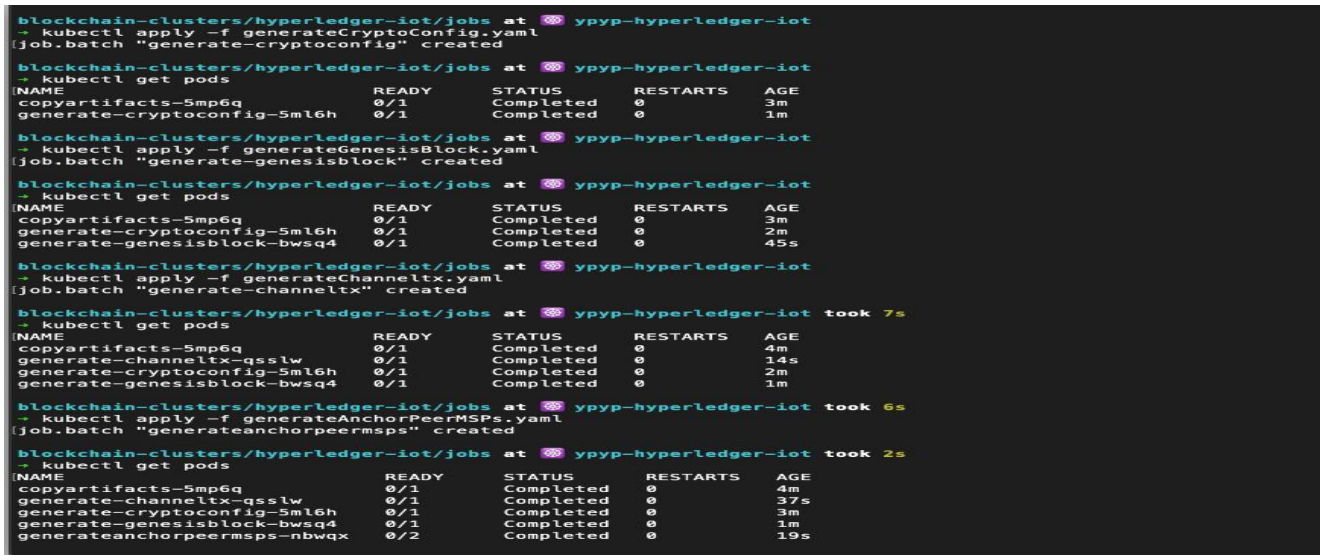


Рисунок 2.11 – Запуск джоб в Kubernetes для створення усіх потрібних артефактів

Далі буде приведено опис усіх цих команд:

– На рисунку 2.12 подана команда яка створює MSP (постачальники послуг членства).

```
$ kubectl apply -f generateCryptoConfig.yaml
job.batch/generate-cryptoconfig created
```

Рисунок 2.12 – Запуск джоби в Kubernetes для створення CryptoConfig

– Наступна команда, яка представлена рисунком 2.13, створить «genesis.block», який використовуватиметься як перший блок блокчейну.

```
$ kubectl apply -f generateGenesisBlock.yaml
job.batch/generate-genesisblock created
```

Рисунок 2.13 – Запуск джоби в Kubernetes для створення першого генезис блоку

– Наступна команда, яка представлена рисунком 2.14, створить конфігураційний файл 'channel1.tx', який використовуватиметься для створення каналу.

```
$ kubectl apply -f generateChanneltx.yaml
job.batch/generate-channeltx created
```

Рисунок 2.14 – Запуск джоби в Kubernetes для створення першого каналу

– Наступна команда, яка представлена рисунком 2.15, згенерує файли «Org1MSPanchors.tx» і «Org2MSPanchors.tx», які використовуватимуться для встановлення однорангових вузлів прив’язки в мережі.

```
$ kubectl apply -f generateAnchorPeerMSPs.yaml
job.batch/generateanchorpeermsps created
```

Рисунок 2.15 – Запуск джоби в Kubernetes для генерації «Org1MSPanchors.tx» і «Org2MSPanchors.tx» для встановлення однорангових вузлів прив’язки в мережі.

Одноранговий вузол на каналі, це вузол який усі інші однорангові вузли можуть виявити та спілкуватися з ним. У кожного учасника каналу є вузол прив’язки або кілька вузлів прив’язки, щоб запобігти єдиній точці збою, що дозволяє одноранговим вузлам, які належать до різних учасників, виявляти всіх існуючих однорангових пристроїв на каналі.

Отже, зараз були виконані необхідні кроки для розгортання мережі. Тепер потрібно розгорнути компоненти Hyperledger Fabric, центр сертифікації, замовника та однорангові вузли у цьому кластері. На рисунку 2.16 усі команди для створення цих ресурсів об’єднані в один скріпт. А на рисунку 2.17 продемонстровано їх виконання та перевірка готовності усіх необхідних компонентів блокчейн мережі.

```
$ cd ../network-deployment
$ sh deployAll.sh
service/orderer created
deployment.apps/orderer created
service/caorg1 created
deployment.apps/caorg1 created
service/org1peer1 created
deployment.apps/org1peer1 created
service/org1peer2 created
deployment.apps/org1peer2 created
service/org2peer1 created
deployment.apps/org2peer1 created
service/org2peer2 created
deployment.apps/org2peer2 created
```

Рисунок 2.16 – Створення компонентів Hyperledger Fabric, центру сертифікації, замовника та однорангових вузлів

```
blockchain-clusters/hyperledger-iot/jobs at ypyp-hyperledger-iot
[→ cd ../network-deployment

blockchain-clusters/hyperledger-iot/network-deployment at ypyp-hyperledger-iot
[→ sh deployAll.sh
service "orderer" created
deployment.apps "orderer" created
service "caorg1" created
deployment.apps "caorg1" created
service "org1peer1" created
deployment.apps "org1peer1" created
service "org1peer2" created
deployment.apps "org1peer2" created
service "org2peer1" created
deployment.apps "org2peer1" created
service "org2peer2" created
deployment.apps "org2peer2" created

blockchain-clusters/hyperledger-iot/network-deployment at ypyp-hyperledger-iot took 18s
[→ kubectl get pods
NAME                              READY     STATUS      RESTARTS   AGE
caorg1-5f44c8d8f6-l4w5z           1/1       Running     0          1m
copyartifacts-5mp6q               0/1       Completed   0          7m
generate-channeltx-qsslw          0/1       Completed   0          3m
generate-cryptoconfig-5ml6h       0/1       Completed   0          5m
generate-genesisblock-bwsq4       0/1       Completed   0          4m
generateanchorpeermsps-nbwqx      0/2       Completed   0          2m
orderer-6c559764f7-6plhl          1/1       Running     0          1m
org1peer1-759d844dc5-vn8wn        3/3       Running     0          1m
org1peer2-76c457b87f-4qgd9        3/3       Running     0          1m
org2peer1-59865f996f-d96rv        3/3       Running     0          1m
org2peer2-75d49665b9-hlbkt        3/3       Running     0          1m
```

Рисунок 2.17 – Створення та перевірка усіх компонентів Hyperledger Fabric, центру сертифікації, замовника та однорангових вузлів

У наступних кроках буде налаштовано ці компоненти відповідно до даного сценарію використання. Для створення каналу на основі раніше згенерованого конфігураційний файл 'channel1.tx' та приєднання усіх вузлів, будуть використовуватись наступні маніфести Kubernetes .yaml. Запуск цих маніфестів представлено на малюнку 2.18.

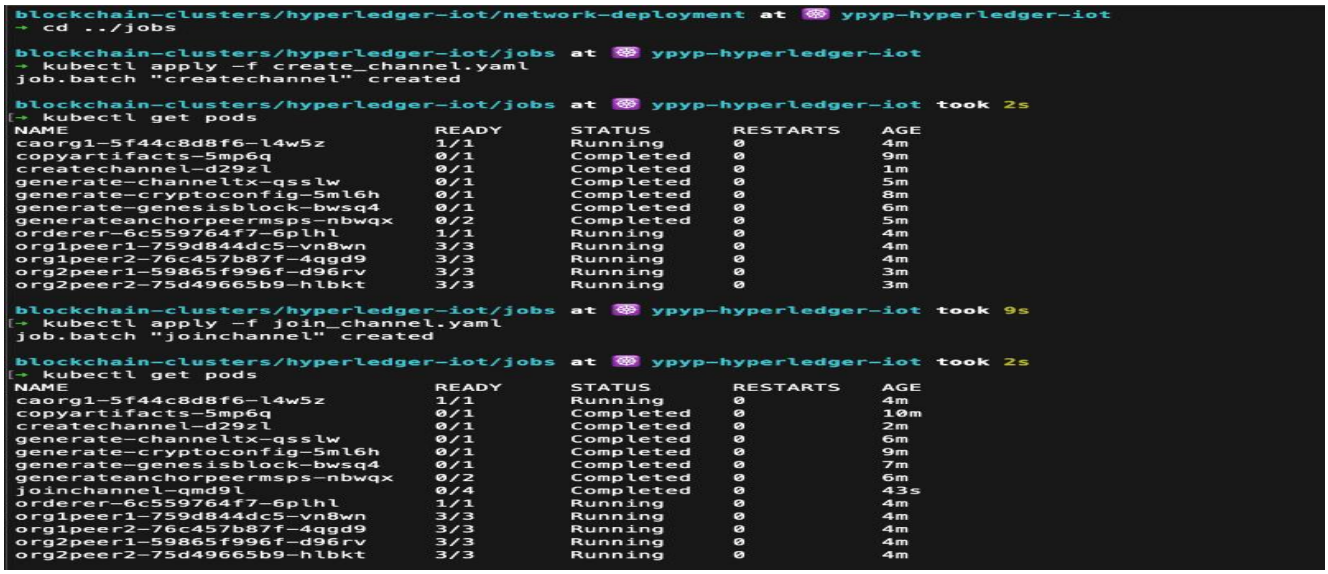


Рисунок 2.18 – Створення каналу та під'єднання усіх однорангових пірів Hyperledger Fabric

І врешті-решт, але все ж таки одним з передостаннім кроків, буде встановлення ланцюгового коду chaincode для однорангових вузлів (org1peer2, org2peer2) і створення екземпляру встановленого ланцюжкового коду в каналі. Крім того, потрібно встановити політику схвалення для запиту 1-ого підпису від кожної з двох організацій. На рисунку 2.19 представлено усі команди для виконання маніфестів. Остання команда оновить канал і перевстановить вузли (org1peer1, org2peer1) знову як вузли прив'язки.

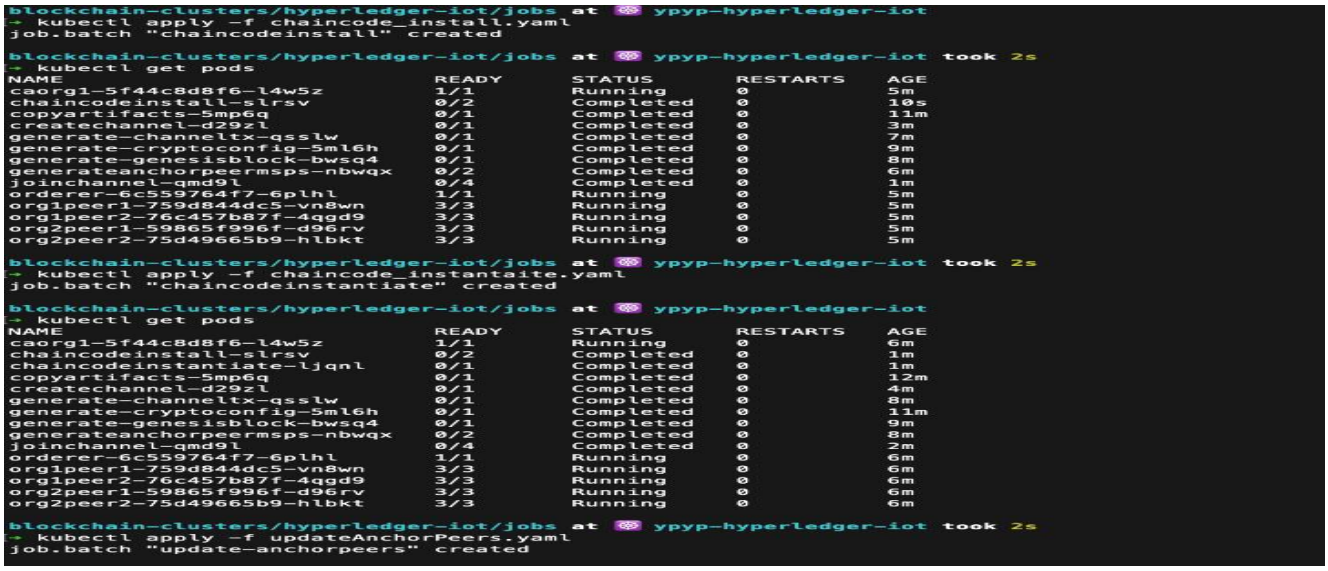


Рисунок 2.19 – Виконання джоб в Kubernetes для встановлення ланцюгового коду chaincode для однорангових вузлів) і створення екземпляру встановленого ланцюжкового коду в каналі

Другим передостаннім кроком є розгортання REST-API та створення служби Kubernetes, яка дає можливість підключатися до цього застосунку за межами кластеру. На рисунках 2.20 та 2.21 представлено успішне розгортання Hyperledger Fabric SDK для Node.js.

```
kubectl create deployment rest-api --image=dnaumenko/rest-api
deployment.apps/rest-api created
kubectl expose deployment rest-api --port=3000 --target-port=3000
service/rest-api exposed
```

Рисунок 2.20 – Розгортання Hyperledger Fabric SDK для Node.js

```
→ kubectl get pods
NAME                                READY   STATUS      RESTARTS   AGE
caorg1-756c665854-v5hdf             1/1     Running     0          13m
chaincodeinstall-8s9sl              0/2     Completed   0          10m
chaincodeinstantiate-2ntb9          0/1     Completed   0          10m
copyartifacts-fnbhl                 0/1     Completed   0          18m
createchannel-ddgjb                 0/1     Completed   0          11m
generate-channeltx-8plvl            0/1     Completed   0          16m
generate-cryptoconfig-8w5cm         0/1     Completed   0          16m
generate-genesisblock-f265g         0/1     Completed   0          16m
generateanchorpeermsps-8fp9h        0/2     Completed   0          16m
joinchannel-h28c4                   0/4     Completed   0          11m
orderer-6b4d69d585-dmmpt            1/1     Running     0          13m
org1peer1-b7b9fcfdb-m7kjn           3/3     Running     0          13m
org1peer2-7f5879b44-d4kw5           3/3     Running     0          13m
org2peer1-7f7bdbcd86-njspg          3/3     Running     0          13m
org2peer2-559ccc95dd-snhbn          3/3     Running     0          13m
rest-api-6d986947cb-8cxl7           1/1     Running     0          23s
update-anchorpeers-k6g9h            0/2     Completed   0          9m40s
```

Рисунок 2.21 – Перевірка розгортання REST-API серверу

Окрім API, в кластері добре було б мати розгорнутий готовий Frontend застосунок NodeRED. На цій інформаційній панелі можна буде переглядати вхідні дані датчиків і історію реєстру блокчейна. Крім того, через цей інструмент будуть виконуватися HTTP-запити. Для цього потрібно лише виконати команди, які представлені на рисунку 2.22.

```
$ cd node-red
$ make build
$ make push
$ kubectl create deployment node-red --image=<your_account_name>/hyperledger-iot-nodered:1.0.2
deployment.apps/node-red created
```

Рисунок 2.22 – Компіляція застосунку NodeRED та створення розгортання в кластері Kubernetes

Замість того, щоб розробляти програму з повними можливостями, цей продукт є набагато більш зрозумілим і повчальним. Основними причинами вибору даного застосунку для симуляції IoT пристрою є :

– Спрощення програмування через особливий підхід FBP;
– Поширення;
– Висока якість;
– Open source.

Можна з впевненістю сказати, що цей інструмент підходить для створення послуг, пов’язаних з Інтернетом речей, таких як контроль даних на пристроях і зв’язування периферійних пристроїв і хмарних сервісів. Спочатку концепція розробки Node-RED була спрямована на IoT, тому це має сенс.

FBP — це парадигма програмування, яка визначає програми як мережі процесів «чорних скриньок», які обмінюються даними через попередньо визначені з’єднання за допомогою передачі повідомлень, де з’єднання вказуються поза процесами. Ці процеси чорного ящика можна безкінечно перепідключати для створення різних додатків без необхідності змінювати їх внутрішньо. Таким чином, FBP природно орієнтований на компоненти.

Після розгортання NodeRED застосунку потрібно отримати до нього доступ в кластері. Для виконання зазначих дій потрібно запустити команди представлені на рисунку 2.23, які потірбно виконати через client kubectl. Та відкрити браузер і перейти до «Your_external_IP»:30002, у результаті чого відкриється служба Node-RED. Наприклад, 52.116.26.52:30002

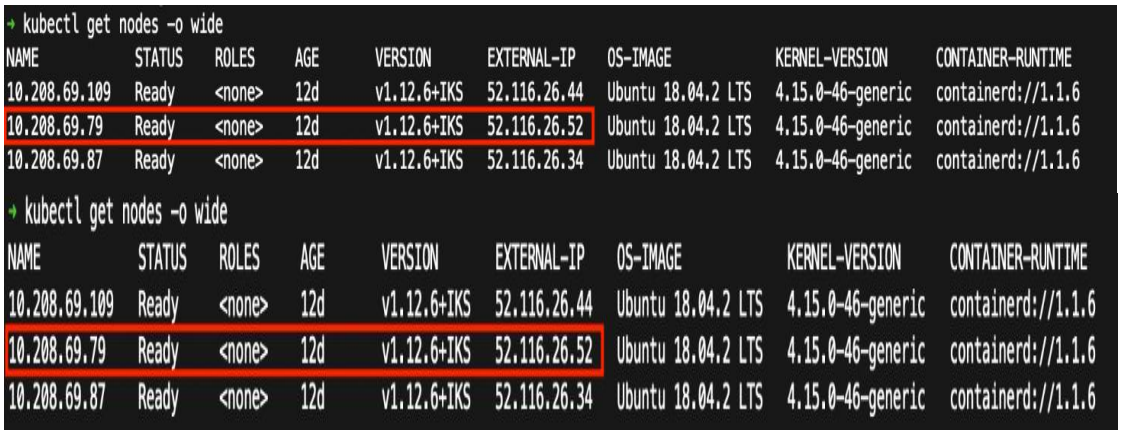


Рисунок 2.23 – Визначення IP адреси NodeRED застосунку в кластері

Наступною дією потрібно виконати три запити HTTP Post відповідно:

– Перший POST запит зареєструє адміністратора з іменем «admin» до центру сертифікації організації 1.
– Другий POST зареєструє реєстр і зареєструє користувача з іменем «user1» до центру сертифікації організації 2.
– Третій POST Зареєструє новий датчик, який використовуватиметься для збору даних.

В результаті цих дій буде отримано вигляд інтерфейсу UI, як показано нижче. Отримані результати можна побачити на рисунку 2.24.

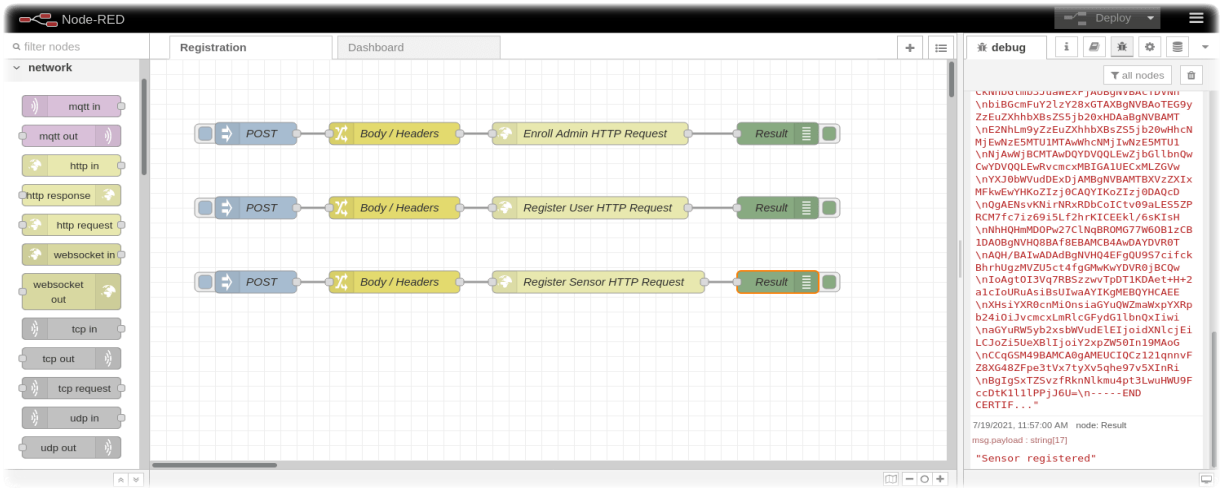


Рисунок 2.24 – Вигляд UI інтерфейсу NodeRED та виконання 3ьох дій

Саме після отримання даного вікна настав час створити інтерфейс користувача, щоб програма виглядала привабливо.

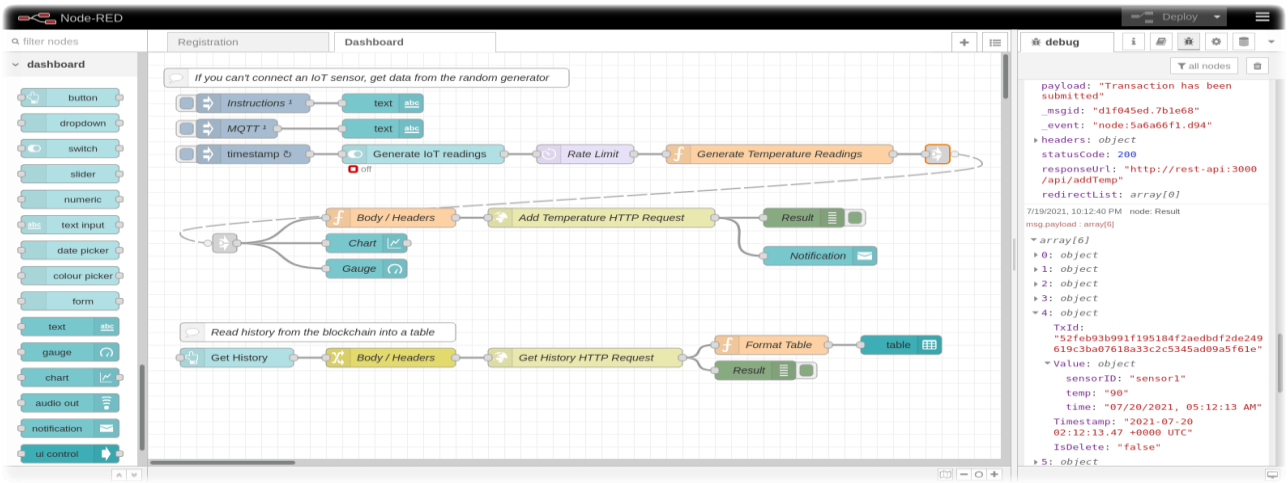


Рисунок 2.25 – Створення програми для генерації даних з віртуального IoT пристрою

Для виконання даної кваліфікаційної роботи міг би знадобитись реальний фізичний IoT пристрій, але щоб обійтись віртуальним потрібно просто перемкнути перемикач Generate IoT readings, щоб створити імітацію даних датчика IoT. Після цього дані датчика можливо буде побачити в прямому ефірі на манометрі та діаграмі. Щоб запитати історію даних датчика, що зберігається в блокчейні, потрібно перейти на вкладку «Історія датчика» в меню. Історія даних надходить із приватній книги Ledger, де дані незмінно зберігаються в блокчейні. На рисунку 2.26 представлено скріншоти остаточного вигляду програми.

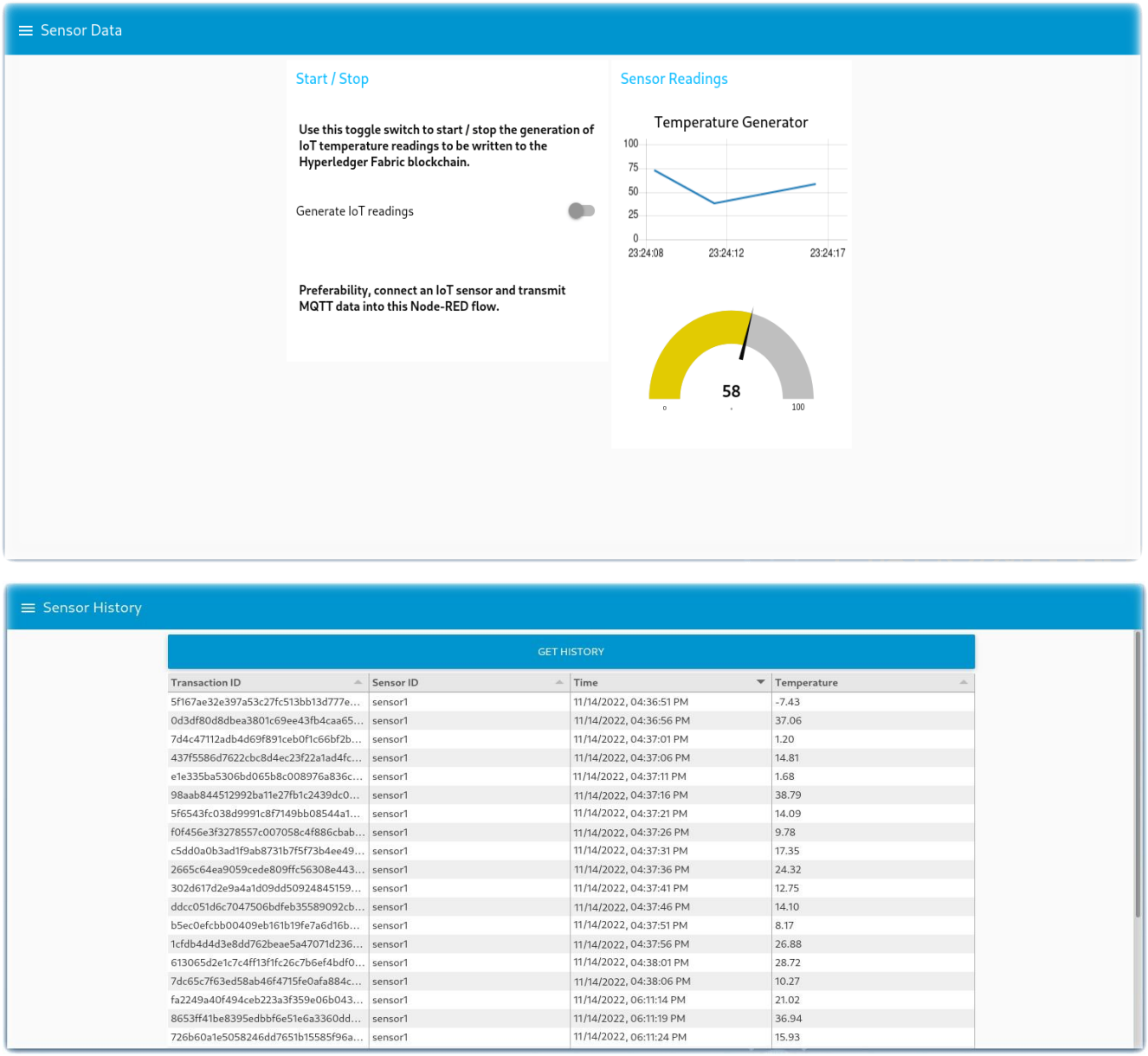


Рисунок 2.26 – Відображення даних з віртуального IoT пристрою на манометрі та діаграмі, а також історія походження даних та підтвердження достовірності даних

2.3 Аналіз реалізації рішення приватної інфраструктури ключів PKI для IoT на основі блокчейну

Щоб встановити довіру через загальнодоступну мережу, потрібна незалежна довірена сторона. Інфраструктура відкритих ключів PKI — це відкритий фреймворк, створений для створення факторів довіри між підключеними до Інтернету користувачами та IoT пристроями, які генерують та обмінюються даними. PKI забезпечує безпечний засіб автентифікації сторін. Підприємства можуть зменшити кількість проблем із розгортанням і керуванням, з якими стикаються додатки, використовуючи PKI. Оскільки компанії все більше переходять на хмарні додатки, надзвичайно важливо захищати чутливі до небезпеки застосунки від нових загроз. Існує багато загроз безпеки під час обміну даними в Інтернеті, такі як крадіжка даних, атаки типу "людина посередині" MITM і витік даних, які можуть скомпрометувати IoT пристрої.

Інтернет дозволяє будь-кому підключатися до будь-кого, і, на відміну від реального світу, тут не існує географічних/фізичних бар’єрів. Це ускладнює ідентифікацію людини в Інтернеті та встановлення довіри для подальшого спілкування. PKI вирішує цю проблему, додаючи довірену третю сторону TTP між першою та другою стороною.

PKI надає ієрархічний стандарт для керування цифровими активами суб’єкта для встановлення безпечного каналу зв’язку. Це не лише користувачі; він також використовується кількома різними системами, такими як електронна пошта, веб-додатки, смарт-картки тощо. PKI використовується для контролю доступу до маршрутизаторів і комутаторів з автентифікацією 802.1X. Програми повинні отримати підписаний сертифікат від ЦС для роботи в ОС. Маршрутизатори та брандмауери використовують сертифікати для автентифікації інших кінцевих точок через Інтернет.

Проблеми існуючої моделі PKI:

- Проблема 1: згідно зі звітом дослідження Ponemon Institute у 2016 році, 62% компаній розгорнули хмарні програми за допомогою PKI, у 2015 році цей показник був 50%. Якщо центральне сховище сертифікатів буде

зламано, це призведе до масового витоку даних і викрадення облікових записів;

– Проблема 2: центральний орган (кореневий орган) відповідає за керування запитами та відповідями DNS (кореневий орган), сертифікатами X.509 тощо.

Організації, як правило, використовують додатковий рівень безпеки, такий як апаратні модулі безпеки (HSM), щоб захистити свої PKI. HSM розгортаються для захисту PKI для найбільш критичного кореня та для видачі закритих ключів ЦС. Організації обирають багатофакторну автентифікацію для адміністраторів і використання HSM. А керування всіма підключеними до Інтернету пристроями та системами зазвичай виконується третьою стороною відкритих ключів та ідентифікаторів. Крім того, ці надійні треті сторони здатні перехоплювати повідомлення та скомпроментовувати цілісність і безпеку користувачів у всьому світі. Було кілька випадків, коли ці надійні треті сторони передавали інформацію про своїх клієнтів службам безпеки та іншим органам. Вони можуть робити це просто для фінансової вигоди.

Отже. PKI має велику вразливість через свою централізовану систему керування. Проте блокчейн принципово децентралізований і дозволяє спілкуватися між декількома сторонами без сторонньої участі. Підхід до децентралізації може бути зміною парадигми PKI, тому для його розгортання потрібен системний підхід.

Як відомо, блокчейн побудований на основі підходу без довіри, який дозволяє як довіреним, так і ненадійним сторонам спілкуватися одна з одною. Однак довіра зазвичай встановлюється між географічно та політично розрізненими учасниками з кількома консенсусними моделями стану бухгалтерської книги. За визначенням, блокчейн дозволяє зберігати будь-які значення на кількох вузлах у мережі.

Проблема принципал-агент виникає, коли існує конфлікт інтересів між агентом і принципалом, який зазвичай виникає, коли агент діє виключно у своїх власних інтересах. У відносинах принципал-агент принципал — це сторона, яка

законно призначає агента приймати рішення та вчиняти дії від його імені. Принципалу можна надати прямий контроль над глобальними ідентифікаторами, такими як домен веб-сайту, шляхом реєстрації ідентифікатора в блокчейні.

У базі даних ключ-значення принципал використовує ідентифікатор як ключ пошуку. Blockchain може дозволити призначення конфіденційних активів, таких як відкриті ключі та інші атрибути, і зробити ці значення доступними для глобального читання безпечним способом, який не може бути скомпрометований жодним MITM, що можливо в PKIX. Це досягається шляхом дозволу найбільш правильному відкритому ключу зв'язатися зі значенням ідентифікатора, а автентифікація виконується шляхом пошуку ідентифікатора останнього відкритого ключа. У цій конструкції DPKI система залишається децентралізованою, а контроль над ідентифікатором залишається за принципалом.

Ethereum є одним із найбільш підходящих і надійних блокчейнів для DPKI. Це програмований блокчейн, який підходить для деталізованої DPKI на основі політик. PKI реалізовано як функцію в смарт-контракті в блокчейні Ethereum. Кожна сутність може мати кілька атрибутів для автентифікації права власності. Ця сутність може бути відкритим ключем або адресою Ethereum. Кожна транзакція ідентифікується за допомогою відкритого ключа, а потім представляється відповідним ідентифікатором об'єкта та PKI. Розумний контракт використовується для програмування подій і функцій різних операцій в PKI. Розумний контракт також можна налаштувати для виклику певних операцій PKI, таких як створення, отримання, видалення, знищення та багато інших. Наступні набори операцій PKI стають доступними за допомогою програмування смарт-контракту:

- Реєстрація сутності;
- Підпис атрибутів;
- Отримання атрибутів;
- Відкликання підпису.

Користувачі або системи додаються до системи PKI шляхом виклику події реєстрації смарт-контракту. Сутність може бути такою ж простою, як адреса

Ethereum, відкритий ключ, ідентифікатор атрибута, дані та хеші даних. Налаштована подія в смарт-контракті збирає сутність і пересилає її як транзакцію в Ethereum. Транзакції в черзі майнятся, і створюється блок, який пізніше буде додано до блокчейну.

Кожен атрибут сутності може бути підписаний системою PKI за допомогою смарт-контракту, після чого буде оформлено транзакцію. Пізніше ця підписана сутність стане доступною для інших сутностей або користувачів.

Атрибути сутностей можна знайти шляхом застосування фільтра до блокчейну за допомогою відповідних ідентифікаторів подій, налаштованих у смарт-контракті.

Розумні контракти можна налаштувати для виклику події відкликання та відкликання підпису на певній сутності.

У розгортанні DPKI реєстратор все ще має певну роль в інфраструктурі, але він обмежений таким чином, щоб гарантувати, що ідентифікаційні дані сутностей представлені в мережі. Приватні ключі мають бути згенеровані децентралізованим способом, щоб гарантувати, що вони залишаються під контролем принципала. Генерація пари ключів від імені принципала має бути суворо заборонена. Не повинно бути однієї сутності, яка може змінювати інші сутності без згоди принципала. Коли простір імен створюється в блокчейні за допомогою смарт-контракту Ethereum, його не можна знищити. Реєстрація та відновлення ідентифікаторів мають бути прозорими. За замовчуванням програмне забезпечення, яке керує ідентифікаторами, має гарантувати, що всі дії, такі як створення, оновлення, оновлення або видалення ідентифікаторів, пересилаються через децентралізований механізм.

Механізм генерації відкритого закритого ключа PKI працює на основі криптосистеми з еліптичною кривою. У розділі 1 було розглянуто, що в криптографії еліптичної кривої відкритий ключ походить від закритого ключа, і цей процес є одним із способів, який означає, що за наявності відкритого ключа неможливо отримати закритий ключ через жорсткість дискретного логарифму еліптичної кривої. У мережі Ethereum усі вузли повинні використовувати

однакову криву ECC для генерації пари відкритого та закритого ключів. Математичний процес та алгоритм генерації відкритого та закритого ключів DPKI складається з наступних дій:

1.Користувач генерує 256-бітний випадковий ключ. Це private_key, приклад якого зображено на рисунку 2.27.

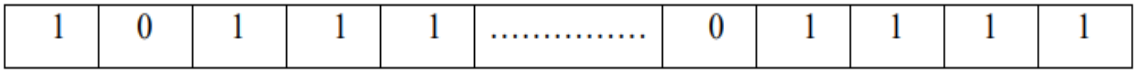


Рисунок 2.27 – 256-бітний закритий ключ випадкового числа

2. ECC генерує 512-бітний відкритий ключ, пов'язаний із закритим ключем відповідно до public_key= $n \times G$ таким чином, що G є опором точки генератора (x, y) дискретному логарифму на еліптичній кривій, а n є private_key, приклад якого зображено на рисунку 2.28.

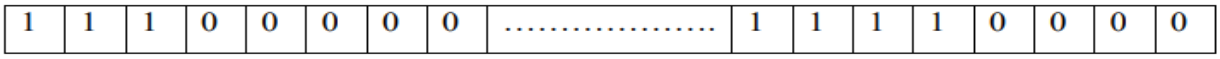


Рисунок 2.28 – 512-бітний відкритий ключ

3. Відомо, що на еліптичній кривій порядок точки G такий, що:

$$\infty = G \times N = G + G + G + \ldots + G \qquad (1)$$

4. Відкритий ключ формально є формою ($n \times x$, $n \times y$), тоді можна визначати:

$$\text{public\_key} = (n \times x) \,||\, (n \times y) \qquad (2)$$

5. Обчислення хешу Keccak-256 для public_key:

$$\text{Hash=Keccak-256 } ((n \times x) \,||\, (n \times y)) \qquad (3)$$

6. Обчислення адреси Ethereum:

$$\text{Ethereum\_Adress = останні 160 біт Hash} \qquad (4)$$

Це називається хешуванням відкритого ключа P2PKH. Хеш публічного ключа є важливою концепцією, яка просто описує, як право власності передається в блокчейні. Крім того, P2PKH відіграє важливу роль у розумінні основ побудови транзакцій для Etherum. Це спрощує процес передачі крипто активів в блокчейні. Крім того, існує багато інших варіантів, таких як P2PK, P2SH, P2MS тощо, але всі вони працюють на одній базовій концепції.

2.4 Аналіз застосування програмно визначеного периметру SDP поверх програмованих мереж SDN-IoT

IoT і SDN це дві новітні технології. Метою IoT є з’єднання об’єктів реального світу через Інтернет, а SDN забезпечує оркестровку для мережі управління шляхом роз’єднання площини керування(control plane) та площини даних(data plane). Кількість підключених об'єктів обчислюється мільярдами, а управління ними і контроль є складним завданням для великої розподіленої мережі. SDN забезпечує гнучкість і можливість програмування в мережі IoT не порушуючи базову архітектуру існуючої реалізації. В цьому пункті другого розділу буде проведено порівняльний аналіз існуючих рішень IoT на основі SDN також буде викладено стислий погляд на нові тенденції. Інтеграція SDN та IoT може виправдати очікування по контролю та управлінню в різних сценаріях.

У SDN площина керування відокремлена від площини пересилання та зв'язок між двома площинами здійснюється через API, наприклад через OpenFlow API. SDN в основному складається з багаторівневої три шарової архітектури:

- Площина даних;
- Площина керування/контролер;
- Прикладний рівень.

Площина даних складається з пристроїв пересилання тобто маршрутизаторів і комутаторів, які пересилають дані лише на контролер. Контролер діє як мозок і керує мережею. Контролер керує всією мережею та має глобальне уявлення про підключені мережі. Усі додатки/програми працюють з контролером. Рівні SDN спілкуються один з одним через відкриті API північного інтерфейсу (NI) та API південного інтерфейсу (SI). Контролер SDN забезпечує програмованість і гнучкість керування станом пересилання потоку в площині даних. SDN може полегшити передачу даних між пристроями IoT для задоволення зростаючих потреб та нових вимог клієнтів. В цьому пункті даного розділу буде показано, як можна створити симуляцію SDN мережі з підключенням пристроїв IoT. Для симуляції даної мережі буде використано ПЗ Cisco Packet Tracer версії 8.0.1. Спочатку мережа буде налаштована таким чином:

– Маршрутизатори будуть працюювати з налаштованим протоколом OSPFv2;
– SSH буде увімкнено на всіх пристроях із користувачем cisco та паролем cisco123!;
– В мережі R1 не буде хостів, а тільки сервер з статично прописаною конфігурацією мережевого адаптеру;
– R2 LAN буде налаштовано статично;
– R3 — це сервер DHCPv4 для LAN1 і LAN2;
– Комутатори рівня 2 будуть налаштовані без VLAN;
– SDN контролер підключений до мережі 192.168.101.2 до комутатора 3 рівня через порт GigabitEthernet прямим мідним кабелем.

На рисунку 2.29 зображено початково налаштовану мережу. Кожен хост може успішно виконувати мережеву команду ping і відправляти пакети на всі мережеві пристрої в даній мережі. SDN контролер поки що не був налаштованим.

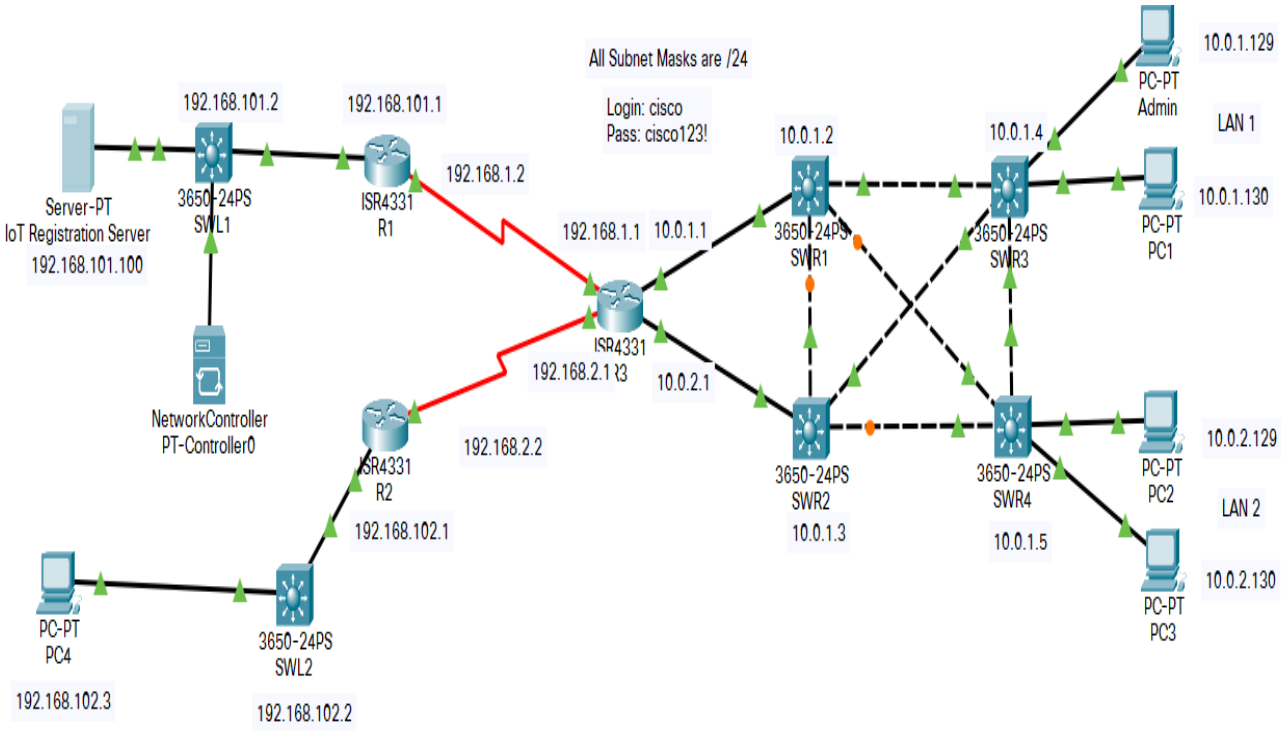


Рисунок 2.29 – Початково налаштована мережа без налаштованого SDN контролера

Packet Tracer надає простий PT-контролер для імітації контролера SDN. У цій частині даного розділу буде підключено та налаштовано PT-контролер. На

рисунку 2.30 наведено детальну інформацію про імітацію SDN контролера в Packet Tracer. Для відкриття даної документації варто відкрити вкладку help в меню даного ПЗ та вибрати contents. Відкриється браузер з документацією, де можна знайти всю інформацію про імітацію даного контролера, як на рисунку.

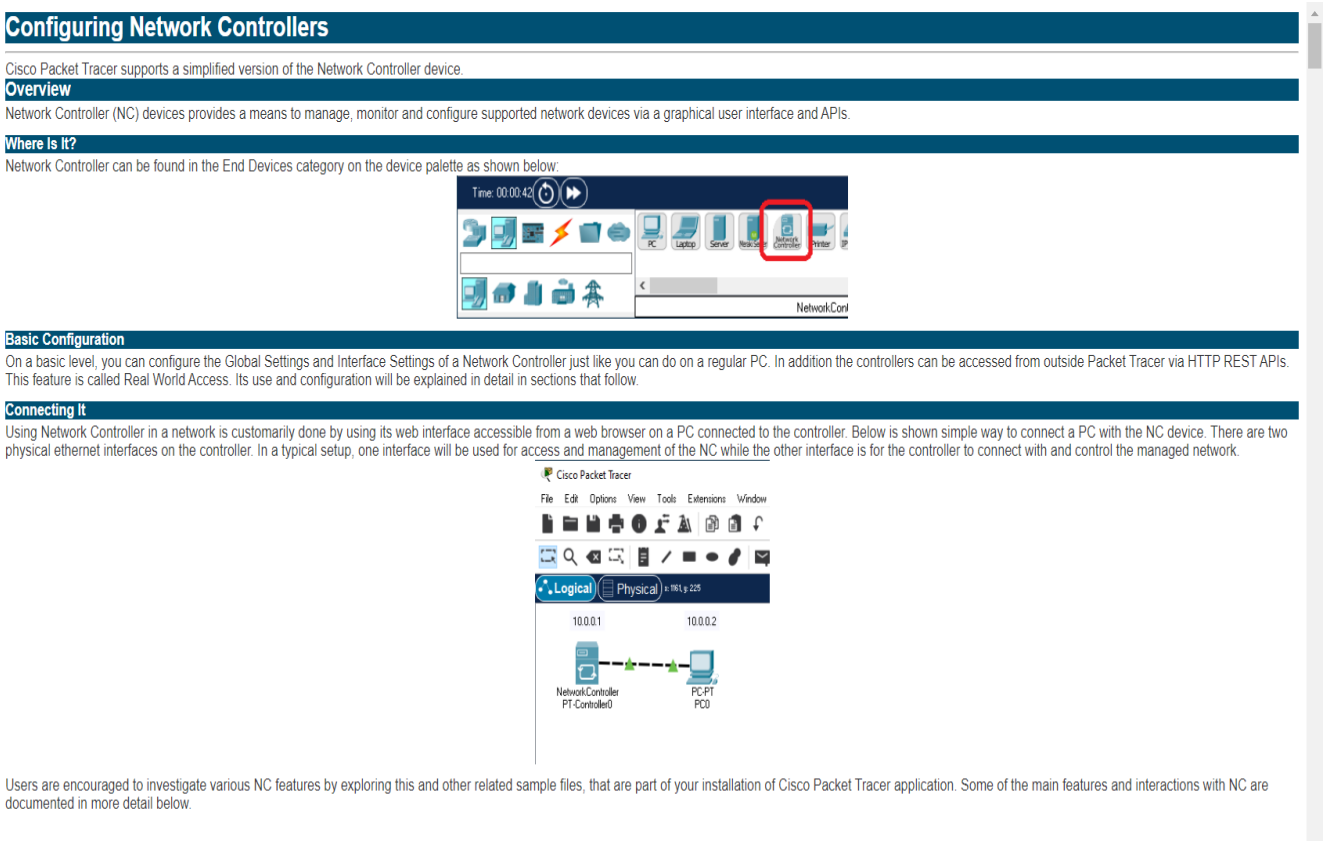


Рисунок 2.30 – Вбудована документація Packet Tracer та відкритий розділ про налаштування SDN контролерів та використання API

Після того як було підключено SDN контролер до мережі, потрібно надати можливість програмувати його з реального світу та отримувати доступ з локальної мережі в якій знаходиться комп’ютер з запущеним Packet Tracer та відкритою симуляцією, для цього потрібно перейти в вкладку config мережевого контролера, а потім до вкладки real world та включити прослуховування на бажаному порту комп’ютера на якому запущений Packet Tracer. Спочатку статус сервера має бути зупинений. Потрібно натиснути увімкнути доступ, щоб увімкнути його. Статус сервера змінюється на прослуховування через порт 58000. Порт можна змінити на будь-яке інше значення, але в даній роботі буде використовуватись порт 58000. Це номер порту буде використаним також в

сценаріях Python. На рисунку 2.31 показано, як можна провести дані налаштування.

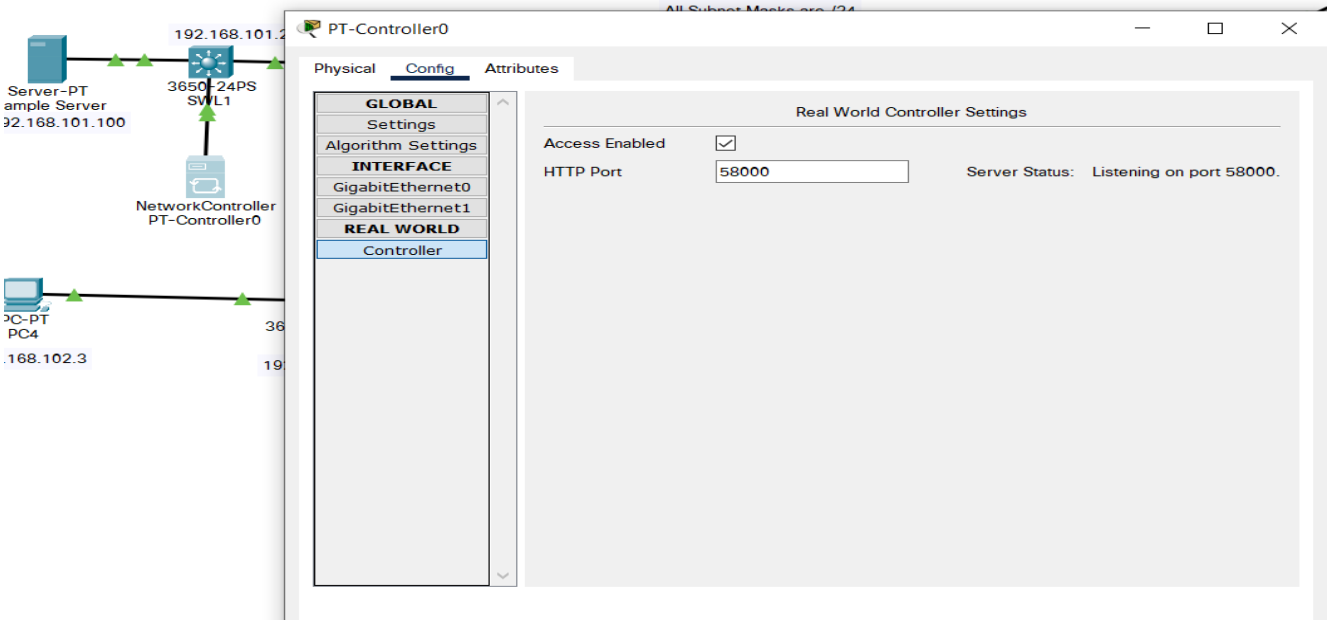


Рисунок 2.31 – Налаштування Cisco Packet Tracer SDN мережевого контролера для отримання доступу з реального світу та можливості використовувати скріпти Python

Після виконання зазначених дій з’явится можливість зайти в UI інтерфейс контролера. На рисунку 2.32 зображено початковий UI інтерфейс після реєстрації нового користувача denys.

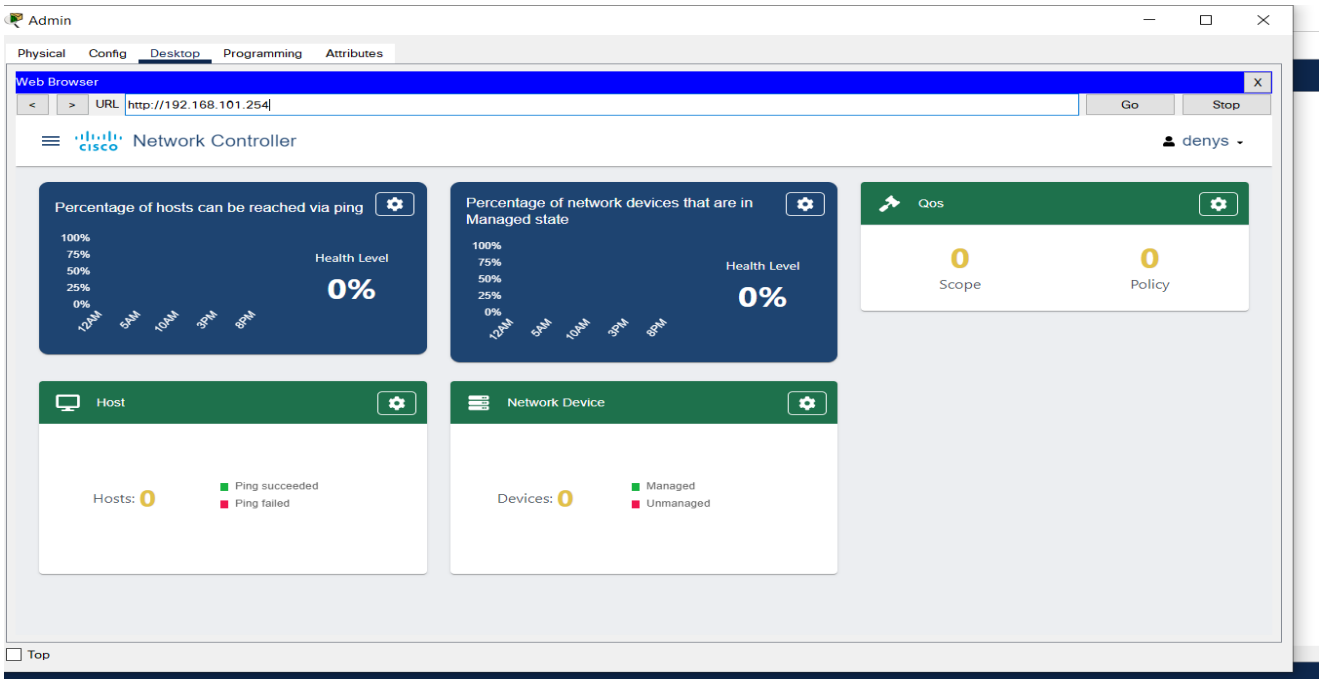


Рисунок 2.32 – UI інтерфейс SDN контролеру після реєстрації користувача

Потім можна провести сканування усіх пристроїв за допомогою протоколу CDP чи вручну додати їх. Для автоматичного сканування потрібно додати паролі та імена користувачів в вкладку provisioning, а вже потім в credentials. На малюнку 2.33 показано як можна додати пошук пристроїв. Після пошуку усіх пристроїв, їх можна переглянути в вкладці Network Devices та на вкладці Topology, як зображено на малюнках 2.34 та 2.35.

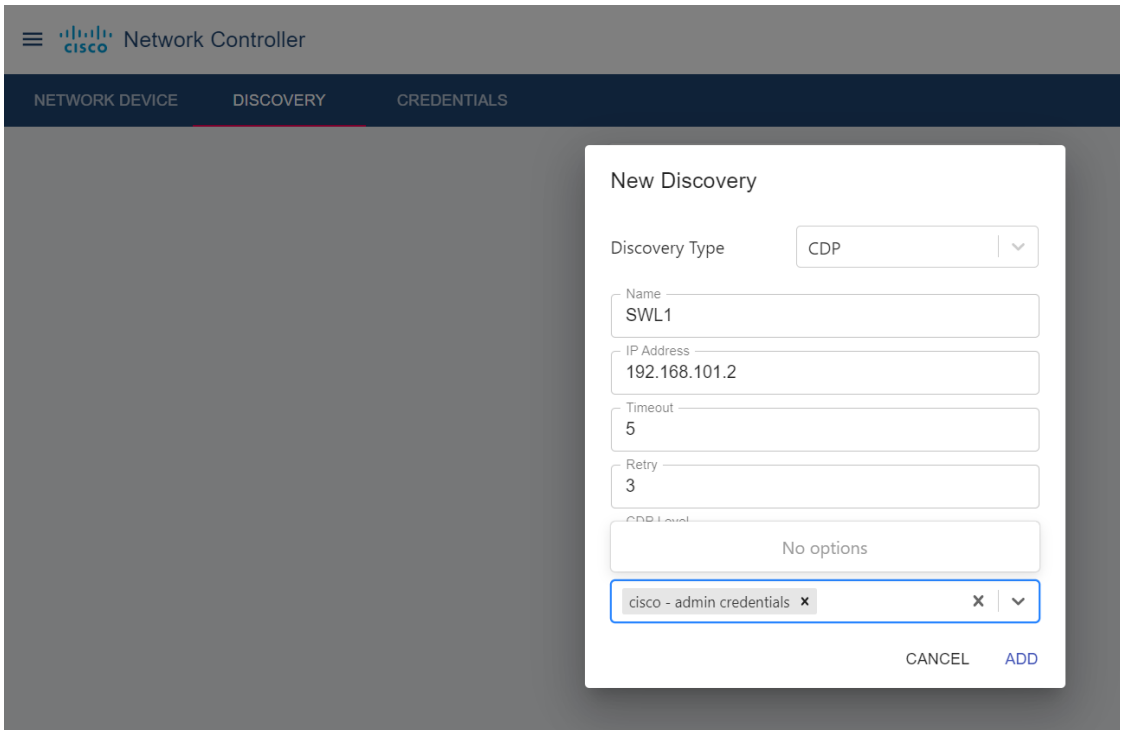
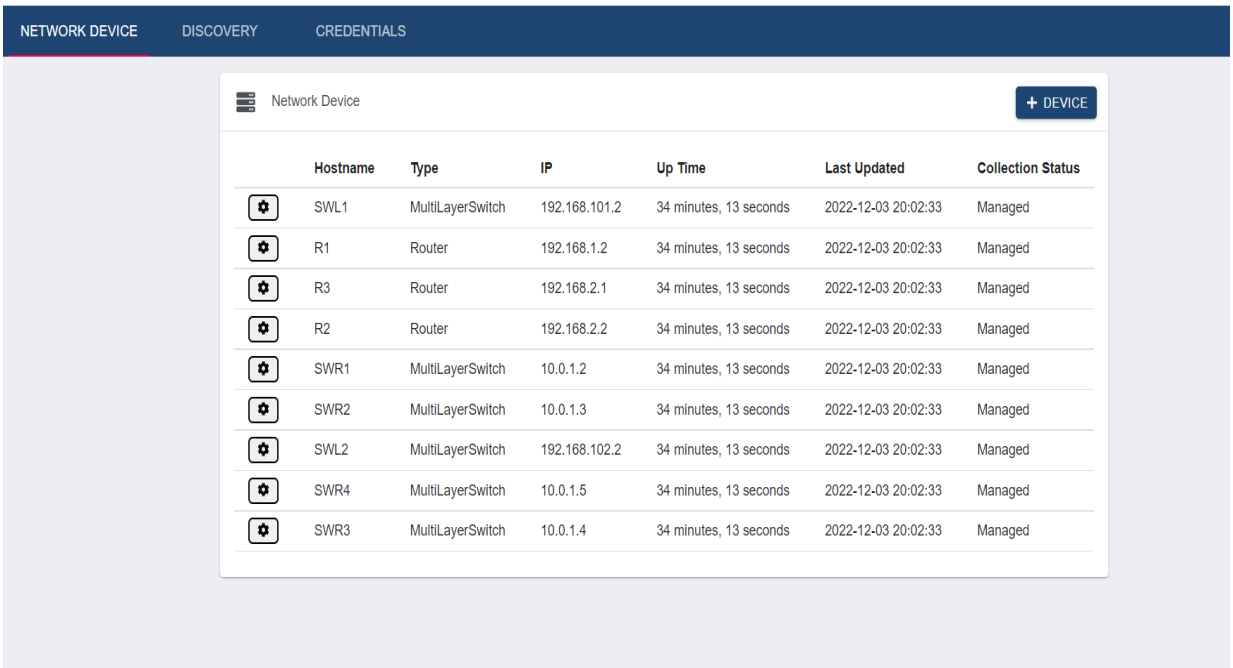


Рисунок 2.33 – Ініціювання пошуку девайсів в мережі



	Hostname	Type	IP	Up Time	Last Updated	Collection Status
	SWL1	MultiLayerSwitch	192.168.101.2	34 minutes, 13 seconds	2022-12-03 20:02:33	Managed
	R1	Router	192.168.1.2	34 minutes, 13 seconds	2022-12-03 20:02:33	Managed
	R3	Router	192.168.2.1	34 minutes, 13 seconds	2022-12-03 20:02:33	Managed
	R2	Router	192.168.2.2	34 minutes, 13 seconds	2022-12-03 20:02:33	Managed
	SWR1	MultiLayerSwitch	10.0.1.2	34 minutes, 13 seconds	2022-12-03 20:02:33	Managed
	SWR2	MultiLayerSwitch	10.0.1.3	34 minutes, 13 seconds	2022-12-03 20:02:33	Managed
	SWL2	MultiLayerSwitch	192.168.102.2	34 minutes, 13 seconds	2022-12-03 20:02:33	Managed
	SWR4	MultiLayerSwitch	10.0.1.5	34 minutes, 13 seconds	2022-12-03 20:02:33	Managed
	SWR3	MultiLayerSwitch	10.0.1.4	34 minutes, 13 seconds	2022-12-03 20:02:33	Managed

Рисунок 2.34 – Перегляд усіх девайсів, які знаходяться в кількох мережах

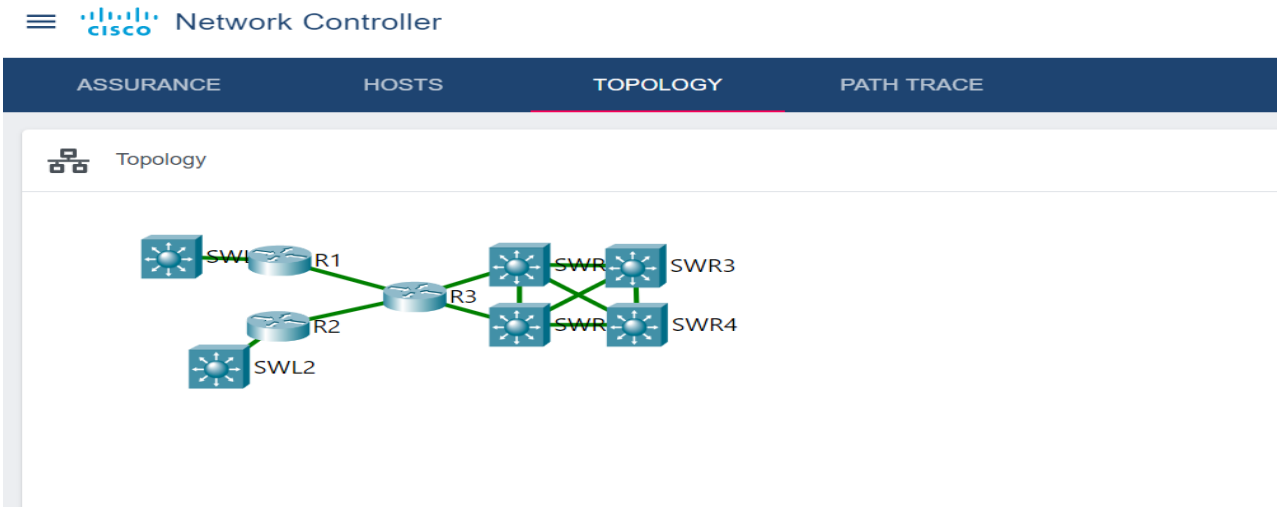


Рисунок 2.34 – Топологія всіх мереж, які доступні для SDN контролеру

Основною перевагою мережевої автоматизації за допомогою контролера є можливість налаштувати параметри мережі та політики безпеки для всіх пристроїв, а потім відкрити цю конфігурацію одним натисканням кнопки.

Далі до 4ого комутатора в симуляції було під’єднано кілька IoT пристроїв, які були також з’єднані між собою, а саме: 2 батареї, 2 вимірювача напруги, 1 сонячна панель та вітрова турбіна. В якості шлюзу було обрано безпровідний пристрій, якій підтримує підключення до всіх вищезгаданих IoT пристроїв. Ці пристрої та шлюз зображені на рисунку 2.35

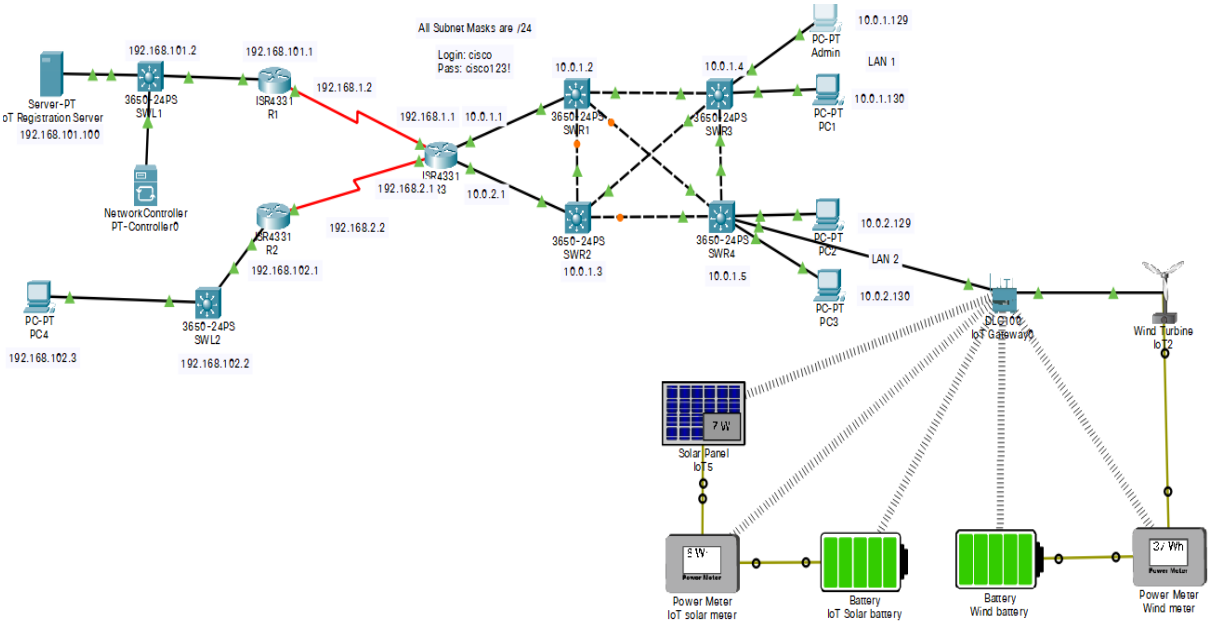


Рисунок 2.35 – Підключення IoT енерго-генеруючих систем до безпровідного шлюзу, який забезпечую підключення до IoT Registration Server

Після виконання підключення, потрібно зареєструвати пристрої на сервері IoT. Сервер було конфігуровано належним чином, що добре видно на рисунку 2.36.

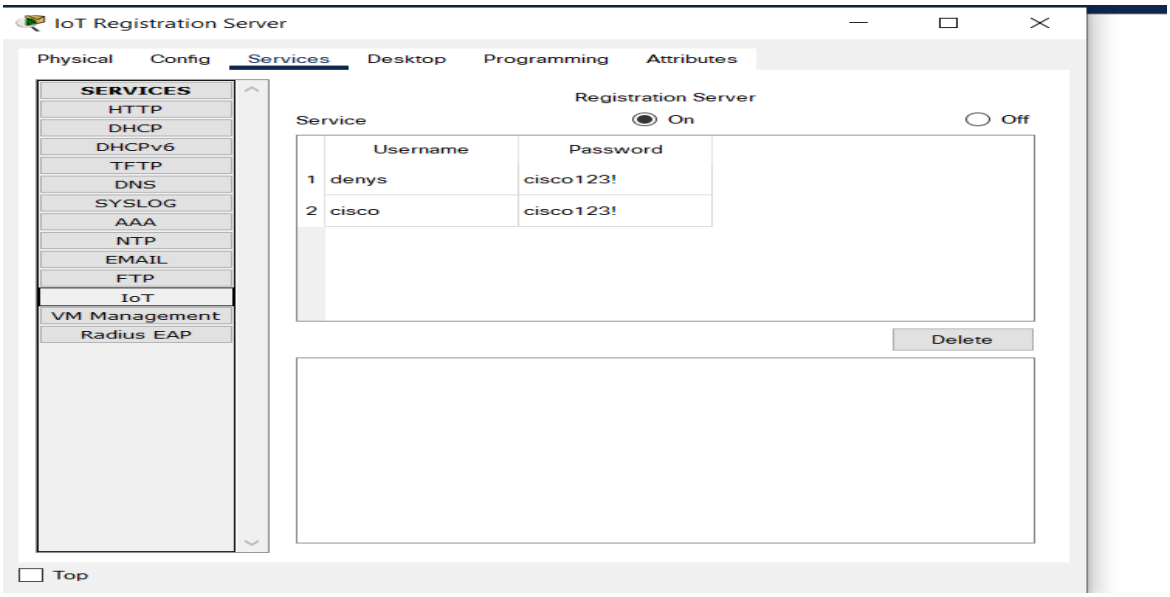


Рисунок 2.36 – Налашування IoT Registration Server

Після даної процедури, потрібно на IoT пристроях налаштувати конфігурацію підключення до серверу реєстрації. На рисунку 2.37 зображено приклад налаштування IoT пристроїв.

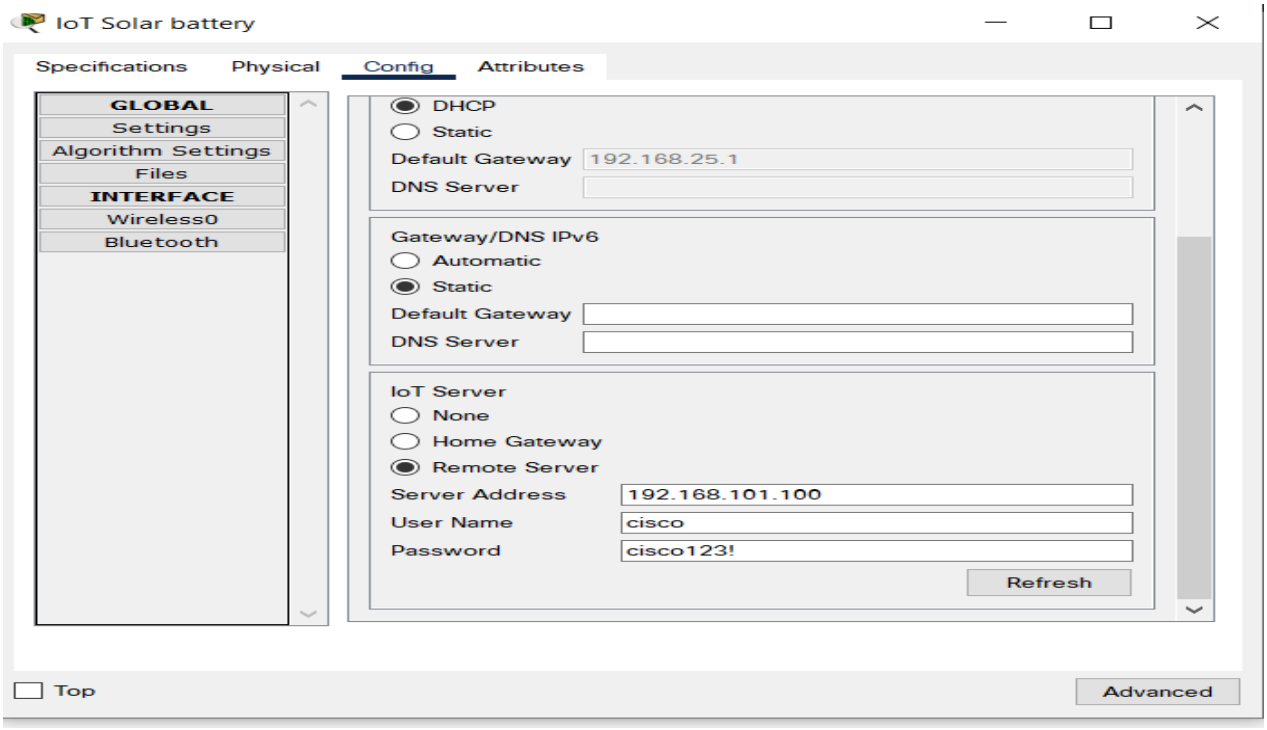


Рисунок 2.37 – Налашування пристрою IoT Solar Battery для підключення до IoT Registration Server

Тепер, можна побачити ці всі пристрої зареєстрованими на сервері. На рисунку 2.38 зображено пристрої, які були успішно зареєстровані в панелі моніторингу IoT. А на рисунку 2.39 зображено процес реєстрації нових пристроїв в SDN контролері.

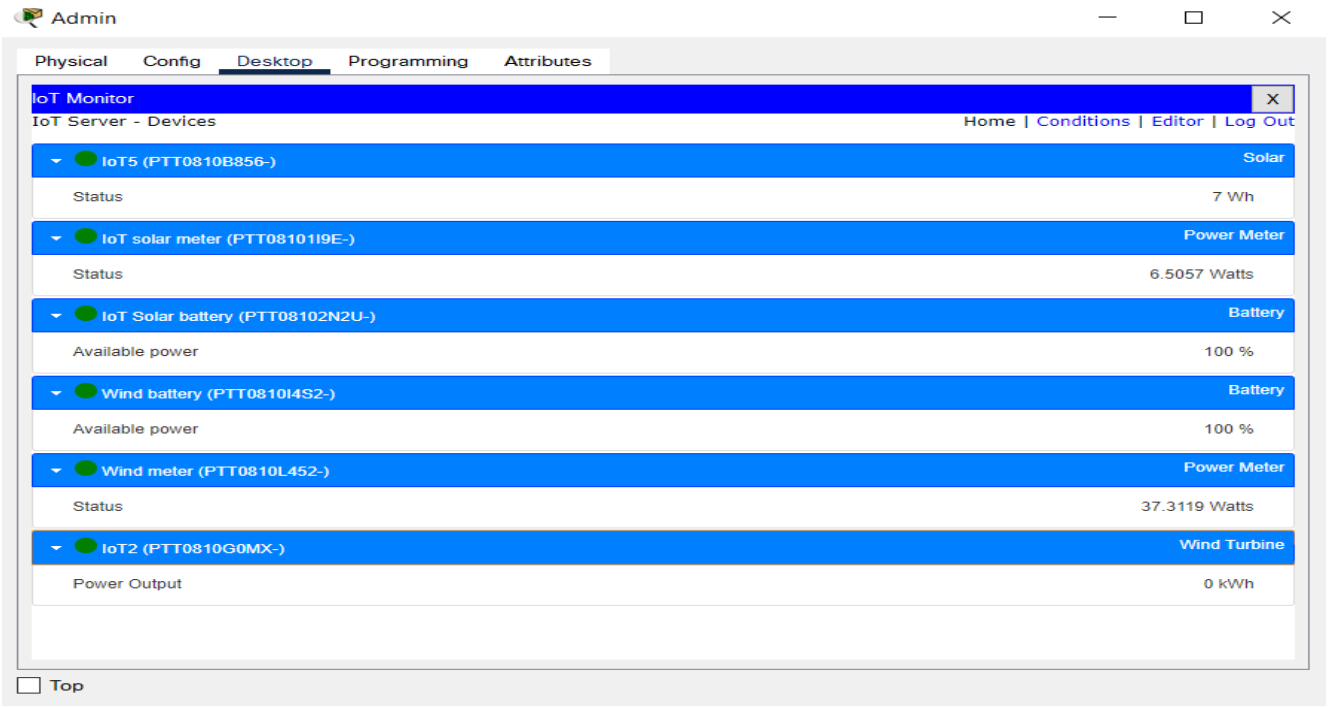


Рисунок 2.38 – Панель моніторингу пристроїв IoT підключених та зареєстрованих на сервері

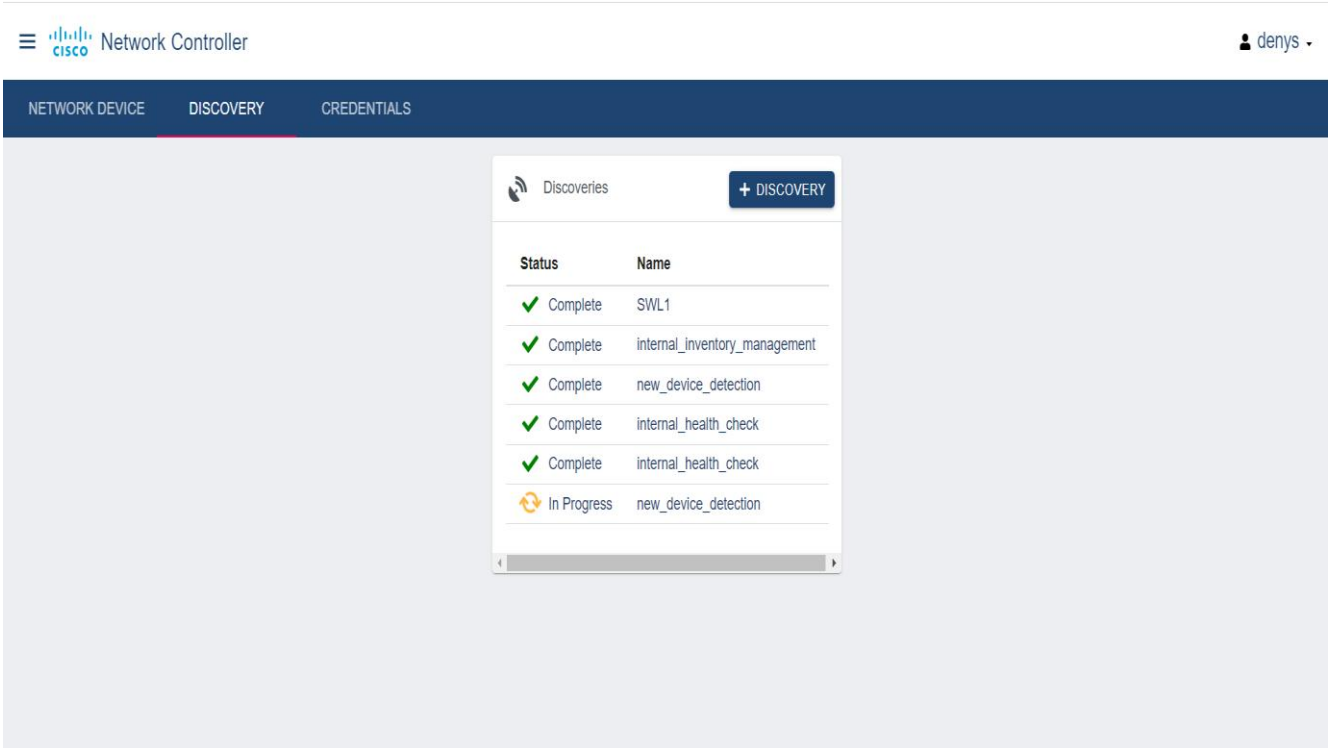


Рисунок 2.39 – Пошук нових пристроїв

Нажаль, Cisco Packet Tracer поки що має обмежений функціонал SDN контролера та немає можливості поки що додати Cisco DNA сервер для створення SDP периметру. Розробники обіцяють в наступних версіях даного ПЗ розширити вищезазначений функціонал. Але сам процес створення периметру SDP для мереж де використовуються IoT пристрої полягає в додаванні Cisco DNA серверу до мережі, та налаштуванні автоматизації сегментації мережі, та створенню віртуальних мереж VN. Потім потрібно зіставити VLAN ідентифікатори з віртуальними мережами. Всі комутатори 2-ого рівня в мережі мають бути замінені на комутатори 3-ого рівня. На рисунку 2.40 показано процес налаштування Cisco DNA серверу та додавання IoT віртуальної мережі. А на малюнку 2.41 зіставлення VLAN ідентифікатору до віртуальної мережи і таким чином створення SDP периметру.

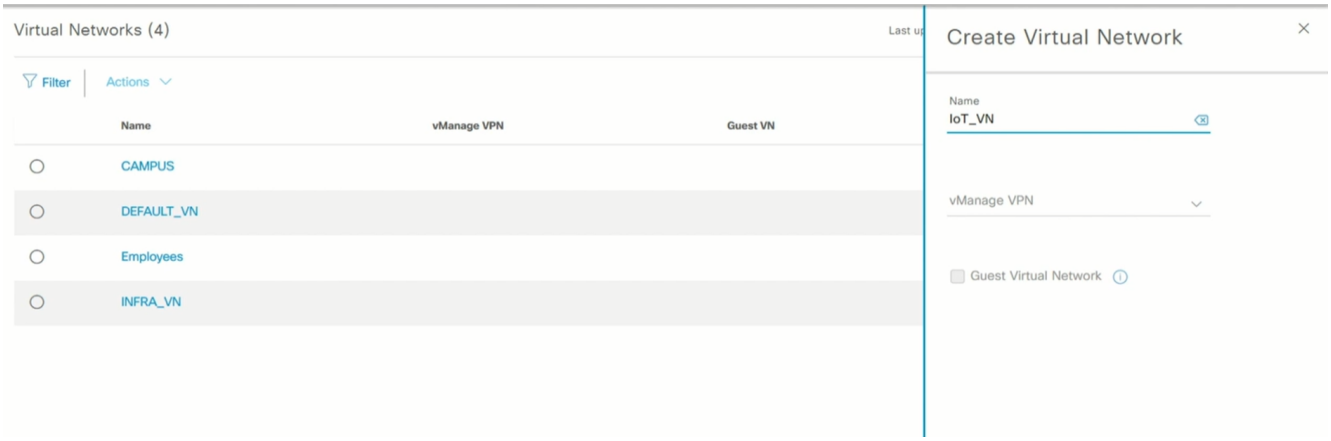


Рисунок 2.40 – Створення віртуальної мережі на Cisco DNA сервері

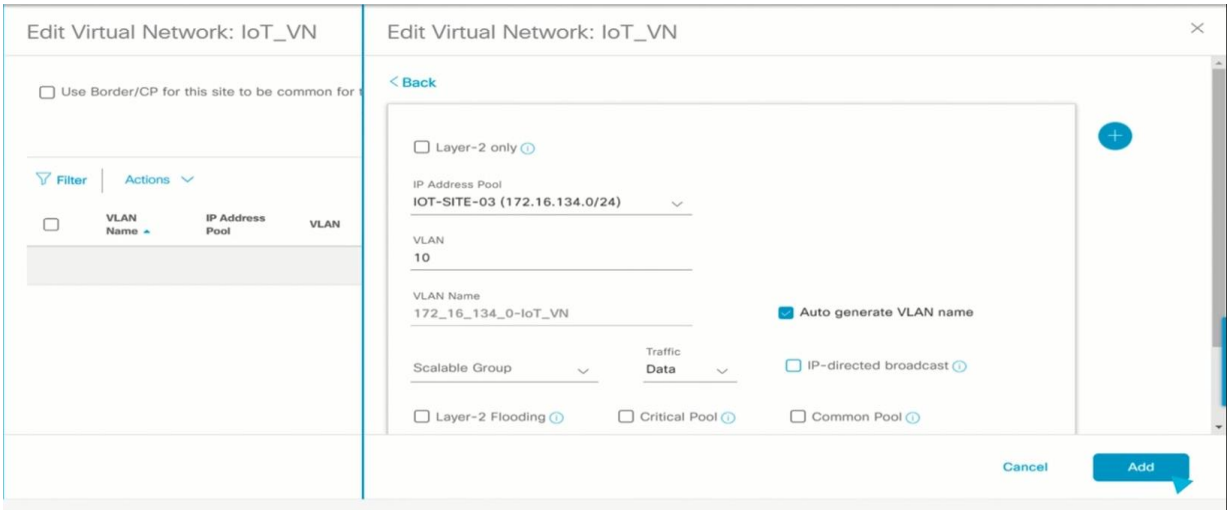


Рисунок 2.40 – Налаштування VN та ствоерення периметру SDP для віртуальної мережі IoT пристроїв

Висновки до розділу 2

У цьому розділі було проаналізовано та реалізовано різні інноваційні методи захисту мереж де використовуються IoT пристрої. Зокрема, було реалізовано розгортання приватного блокчейну для створення блокчейн мережі та захисту IoT пристроїв на рівні передачі даних. Кожна відправка інформації з IoT пристроїв була записана як транзакція в приватному блокчейні Hyper Ledger Fabric. Також було досліджено різні методи імплементації децентралізованої інфраструктури приватних ключів на блокчейні Ethereum, що зробила механізм PKI в дуже багато разів безпечнішим та надійнішим. І останній пункт цього розділу був призначений іншій технології захисту даного типу мереж, а саме SDP поверх SDN. Було розгорнуто прототип програмно визначеної мережі в ПЗ Cisco Packet Tracer на базі спрощеного SDN контролера, а також було описано процес створення програмно визначеного периметру за допомогою Cisco DNA серверу та технології Cisco SDA.

3. НАДАННЯ РЕКОМЕНДАЦІЙ ТА ПРОПОЗИЦІЙ ЩОДО ВПРОВАДЖЕННЯ НАЙКРАЩИХ ПРАКТИК ДЛЯ ЗАХИСТУ IOT ЕКОСИСТЕМ ТА МЕРЕЖ

3.1 Впровадження найкращих практик кібербезпеки IoT пристроїв

Безпеку для IoT екосистеми потрібно розглядати з самого початку проектування, а не модернізувати в кінці циклу розробки програм або в production умовах. На той момент коли застосунок буде розгорнутим в виробництві вже може бути занадто пізно. Безпеку також потрібно розглядати цілісно, від апаратного забезпечення до ПЗ та хмари. Минулий розділ ілюстрував комплексні проекти по захисту IoT від датчика до хмари та розглядав реалізацію найпередовіших технологій захисту мереж. Але окрім застосування новітніх технологій, потрібно розглядати застосування традиційних практик захисту самих IoT пристроїв. Намір полягає в тому, щоб розгорнути систему з різними рівнями захисту, щоб ефективність роботи зловмисника була близькою до нуля. Нижче наведено конкретний список традиційних рекомендацій та ідей щодо безпеки, якими не варто нехтувати та потрібно виконувати при розгортанні екосистем IoT:

- Потрібно використовувати найновішу операційну систему та бібліотеки з усіма відповідними виправленнями;
- Потрібно використовувати апаратне забезпечення, яке включає такі функції безпеки, як Trusted Execution;
- Потрібно використовувати середовища та модулі довіреної платформи;
- Потрібно унеможливлювати декомпіляцію коду в надії на те, що хакер не розшифрує та не переробить його;
- Підписувати, шифрувати та захищати свої мікропрограми та образи програмного забезпечення потрібно тільки довіреними засобами;
- Рандомізування паролей потрібно відбуватись за замовчуванням;
- Потрібно використовувати Root of Trust і безпечне завантаження;
- Потрібно усувати жорстко закодовані паролі в образах ПЗУ;

- Усі IP-порти пристрою мають залишатися закритими за замовчуванням;
- Потрібно використовувати алгоритми Randomization Address Space Layout Randomization Stack Canaries для управління оперативною пам’яттю, які доступні у майже всіх сучасних операційних системах;
- Потрібно використовувати автоматичні оновлення;
- Потрібно надавати виробникам пристрою механізм для виправлення помилок і вразливостей у production умовах;
- Також повинен бути створений план на кінець життя IoT девайсу;
- Потрібно впроваджувати програми винагород за виявлення вразливостей та вад;
- Найкращою практикою також є участь у знаходженні та поширенні інформації про активні загрози в US-CERT, щоб інші учасники негайно дізнавалися про активні експлойти та кіберзагрози;
- Як би не спокусливо було просто створити проект із використанням MQTT, HTTP чи інших незахищених протоколів, завжди потрібно думати про безпеку та автентифікацію, увімкнену через TLS або DTLS;
- Також потрібно шифрувати дані, які передаються з датчика в хмару.

В цьому списку спочатку було визначено найкращу практику що стосуються вчасного оновлення операційних систем. Це пов’язано з тим що найновіші версії операційних систем включають підтримку пристроїв з Trusted Execution. Це середовища виконання, а саме безпечна зона процесора, яка забезпечує захист коду та даних, що знаходяться в цій зоні. Зазвичай це середовище виконання на ядрі головного процесора, де код для безпечного завантаження, грошових переказів або обробки закритого ключа виконуватиметься з вищим рівнем безпеки, ніж інший код. ARM продає захисний кремнієвий IP-блок для виробників SOC, який забезпечує апаратне забезпечення Root of Trust, а також інші служби безпеки. TrustZone поділяє апаратне забезпечення на безпечні та незахищені «світи». TrustZone - це мікропроцесор, відокремлений від незахищеного ядра. IT-спеціалісти працюють із довіреною ОС, спеціально розробленою для забезпечення безпеки, яка має чітко визначений

інтерфейс для реального світу. Захищені активи та функції містяться в довіреному ядрі. Перемикання між світами здійснюється за допомогою апаратного перемикання контексту, усуваючи потребу в захищеному програмному забезпеченні. Інші способи використання TrustZone — це керування системними ключами, транзакціями кредитних карток і керування цифровими правами. Ця форма безпечного ЦП, надійної ОС і RoT називається довіреним середовищем виконання TEE.

Корінь довіри RoT — це апаратно-перевірений процес завантаження, який гарантує, що перший виконуваний код операції починається з незмінного джерела. Це прив’язка процесу завантаження, яка згодом відіграє роль у завантаженні решти системи від BIOS до операційної системи до програми. RoT — це базовий захист від руткітів. Кожна фаза перевіряє наступну фазу процесу завантаження та створює ланцюжок довіри. RoT може мати різні методи запуску, наприклад: Завантажтеся з ПЗП або пам’яті без можливості запису, щоб зберегти образ і кореневий ключ; Завантаження з одноразової програмованої пам'яті; Завантаження із захищеної області пам’яті, яка завантажує код у захищене сховище пам’яті.

RoT також має перевіряти кожну нову фазу завантаження. Кожна фаза завантаження підтримує набір криптографічно підписаних ключів, які використовуються для перевірки наступної фази завантаження. На малюнку 3.1 зображено цей процес валідації.

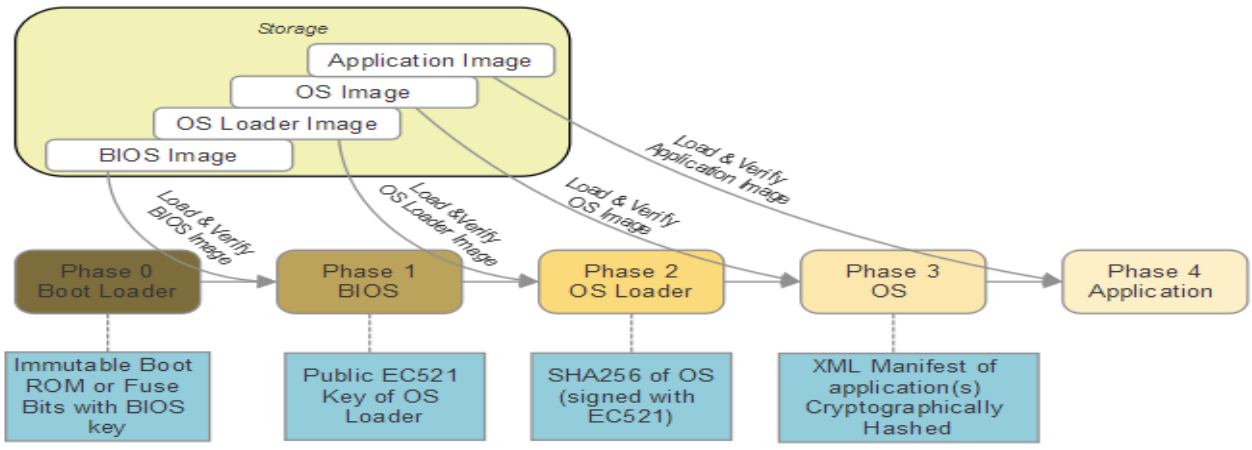


Рисунок 3.1 – Процес валідації кожної фази завантаження RoT

Також в списку по впровадженню найкращих практик захисту пристроїв IoT було згадано про алгоритми Randomization Address Space Layout Randomization Stack Canaries для управління оперативною пам’яттю. Хоча операційна система більше використовує простір віртуальної пам’яті, ніж апаратне забезпечення, важливо враховувати ASLR, як найкращу практику по захисту IoT пристроїв. Цей тип контрзаходів спрямований на переповнення буфера, а також на атаки звернень до системних бібліотек. Ці атаки базуються на тому, що зловмисник розуміє структуру пам’яті та змушує звертатися до певного безпечного коду та бібліотек. Виклик цих бібліотек стає особливо трудомістким, якщо простір пам’яті рандомізується під час кожного завантаження. Linux надає можливість ASLR за допомогою патчів PAX і Exec Shield. Microsoft операційні системи для IoT також забезпечують захист стека та блоків процесу. В свою чергу, Stack Canaries захищає простір стека від переповнення та запобігає виконанню коду з нього.

Ще не менш важливою практикою захисту IoT пристроїв є унеможливлення декомпіляції прошивки та бінарних файлів які є в памяті IoT пристроїв. Бінарна обфускація — це спосіб для розробників зробити код програми складним для розуміння або змінити його. Він також використовується, щоб приховати дані від легкого перегляду. Його можна класифікувати як техніку проти реверсування, яка збільшує час обробки для реверсування. Обфускація також може використовувати алгоритми шифрування та дешифрування разом із жорстко закодованим або створеним кодом ключем шифру. Результатом використання цих інструментів, таких як пакувачі та шифрувальники, є трансформована версія оригінального виконуваного файлу, яка все ще поводиться точно так само, як поведінка вихідного потоку коду. У світі зловмисного програмного забезпечення двійкова обфускація є загальноприйнятою технікою, яку використовують віруси, щоб перемогти антивірусне програмне забезпечення на основі сигнатур. Коли вірус заражає файли, він заплутує свій код за допомогою поліморфізму або метаморфізму. Сигнатури — це набори послідовностей байтів, унікальні для певного шкідливого програмного забезпечення.

Результатом використання цих інструментів, таких як пакувачі та шифрувальники, є трансформована версія оригінального виконуваного файлу, яка все ще поводиться точно так само, як поведінка вихідного потоку коду. Власні виконувані файли більш відомі як файли PE для Windows і файли ELF для Linux. Ці файли скомпільовані до низькорівневого формату; тобто використовуючи мову асемблера, як інструкції x86. Кожен виконуваний файл має структуру заголовка, розділу коду, розділу даних та інших відповідних розділів. Розділ коду містить фактичні коди інструкцій низького рівня, тоді як розділ даних містить фактичні дані, які використовує код. Заголовок містить інформацію про файл, розділи та спосіб відображення файлу як процесу в пам'яті. Це показано на рисунку 3.2.

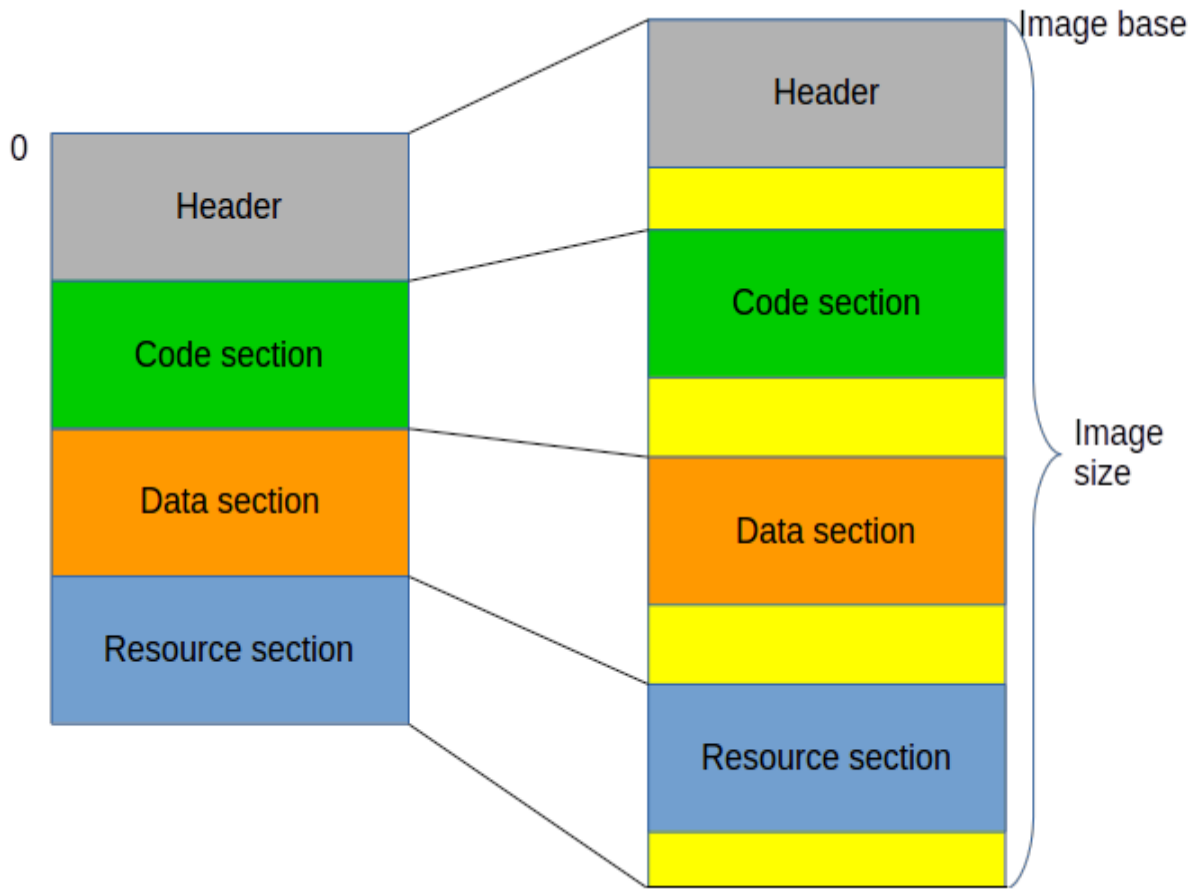


Рисунок 3.2 – Структура виконуваних файлів, які були компільовані

Інформацію заголовка можна класифікувати як необроблену та віртуальну. Необроблена інформація містить відповідну інформацію про фізичний файл, таку як зміщення та розмір файлу. Зсуви відносяться до зсуву файлу 0. У той час як віртуальна інформація складається з відповідної інформації щодо зсувів пам'яті в

процесі, віртуальні зсуви зазвичай відносяться до бази зображення, яка є початком образу процесу в пам’яті. База образу — це адреса в просторі процесу, виділена операційною системою. По суті, заголовок повідомляє , як операційна система має відображати файл (необроблений) та його розділи в пам’яті (віртуальні). Крім того, кожен розділ має атрибут, який повідомляє нам, чи можна цей розділ використовувати для читання, запису чи виконання.

Бібліотеки або модулі, що містять функції, необхідні для коду, також перераховані в частині файлу, яку можна побачити в інших розділах, крім розділів коду та даних. Це називається таблицею імпорту. Це список функцій API та бібліотек, з яких він походить. Після відображення файлу операційна система завантажує всі бібліотеки в одному просторі процесу. Бібліотеки завантажуються так само, як і виконуваний файл, але у більш високу область пам’яті того самого процесу. Коли все зіставлено та завантажено належним чином, ОС зчитує адресу точки входу із заголовка, а потім передає виконання коду на цю адресу.

Обфускатори також класифікуються як модифікатори коду, які змінюють структуру коду, зберігаючи потік програми. Техніка CFF перетворює невеликий код для виконання в циклі, яким керує контрольний прапор. Однак обфускація не обмежується технікою CFF. Одним із основних методів обфускації є спотворення або шифрування назви функцій, щоб декомпілятори не змогли правильно розпізнати функцію. Прикладами таких високорівневих інструментів обфускації є Obfuscar, CryptoObfuscator і Dotfuscator. На рисунку 3.3 зображено приклад унеможливлення декомпіляції JavaScript функції. Оригінальний код знаходиться ліворуч, а його заплутана версія – праворуч.

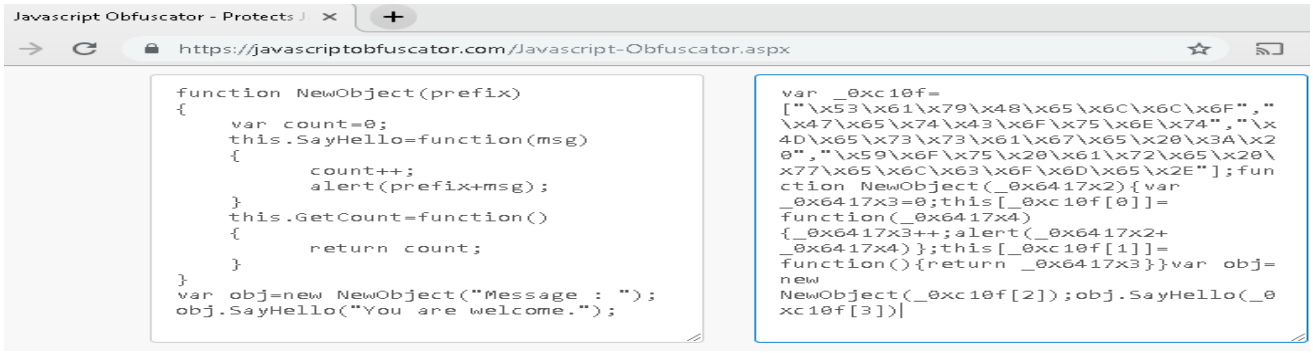


Рисунок 3.3 – Унеможливлення декомпіляції функції JavaScript

3.2 Унеможливлення декомпіляції коду прошивки IoT пристроїв та коду запущених служб на прикладі прототипів в Packet Tracer

В реалізації SDN мережі в ПЗ Cisco Packet Tracer доступна зміна прошивки IoT пристроїв та їх програмування на мові Python та Javascript. Це може бути особливо корисно в реальному світі для користувачів реальних IoT девайсів чи організацій, які впроваджують мережі IoT. Але також це може становити загрозу, бо хакери можуть прочитати код прошивки пристрою та прошивок служб запущених на ньому, щоб підробити цей пристрій та зробити свій з аналогічною версією клієнтського ПЗ з часиною свого зловмисного коду. Наприклад, в мережі даного проекту SDN є 6 IoT пристроїв кожен з яких програмується та підключається до IoT Registration Server. В реальному світі це б дало зловмиснику спробу зчитати прошивку та зробити свій пристрій чи емуляцію пристрою для того щоб скомпрометувати IoT Registration Server.

Спочатку було розглянуто JavaScript код прошивки пристроїв сонячної панелі та вітряного генератору. На рисунку 3.4 зображено частину коду прошивки прототипу IoT пристрою сонячної панелі. А на рисунку 3.5 іншого прототипу IoT пристрою вітряної турбіни.

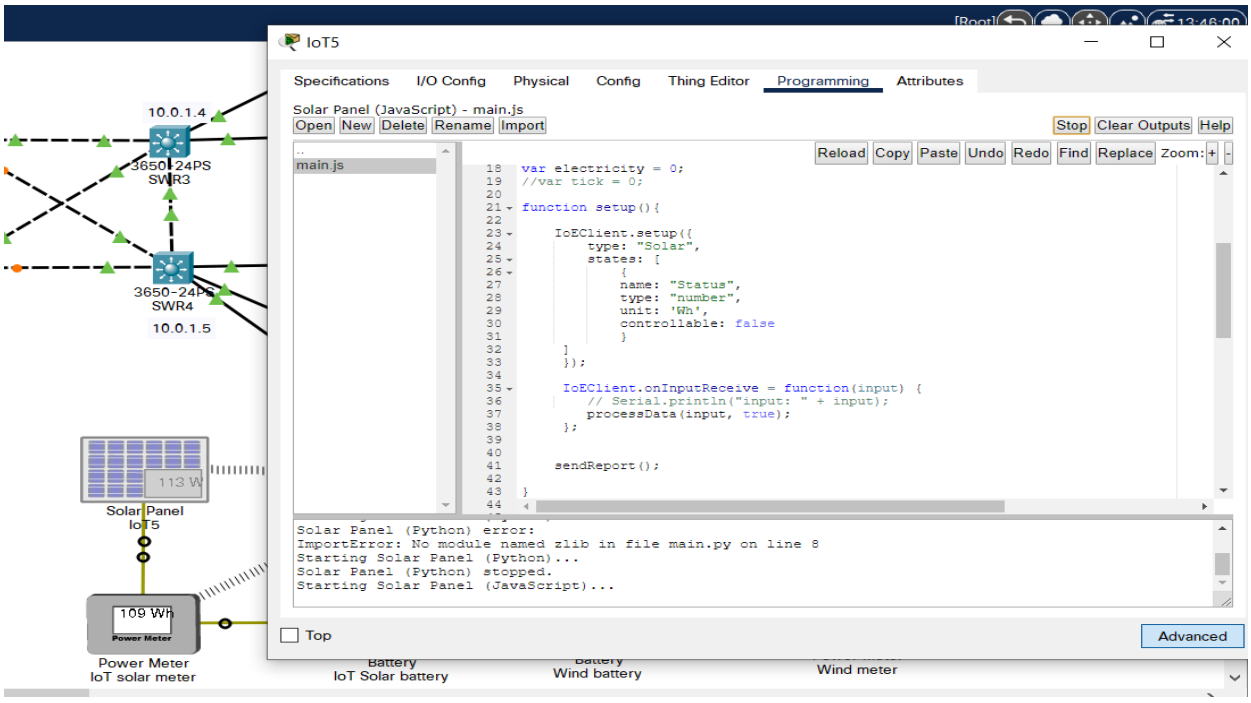


Рисунок 3.4 – Код прошивки прототипу IoT пристрою сонячної панелі в Cisco Packet Tracer

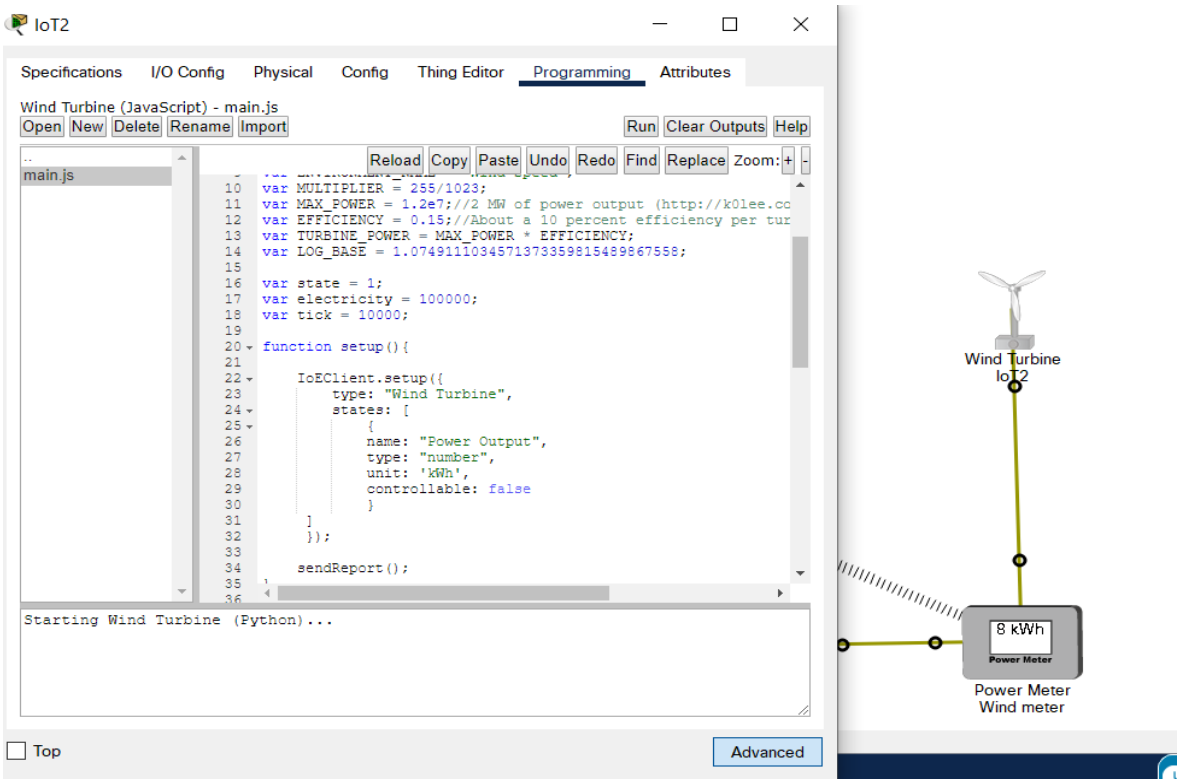


Рисунок 3.5 – Код прошивки прототипу IoT пристрою вітряної турбіни в Cisco Packet Tracer

Потім до цього коду було застосовано метод обфускації з метою унеможливити зчитування та підробку цього коду. На рисунку 3.6 показано результат роботи обфускатора.

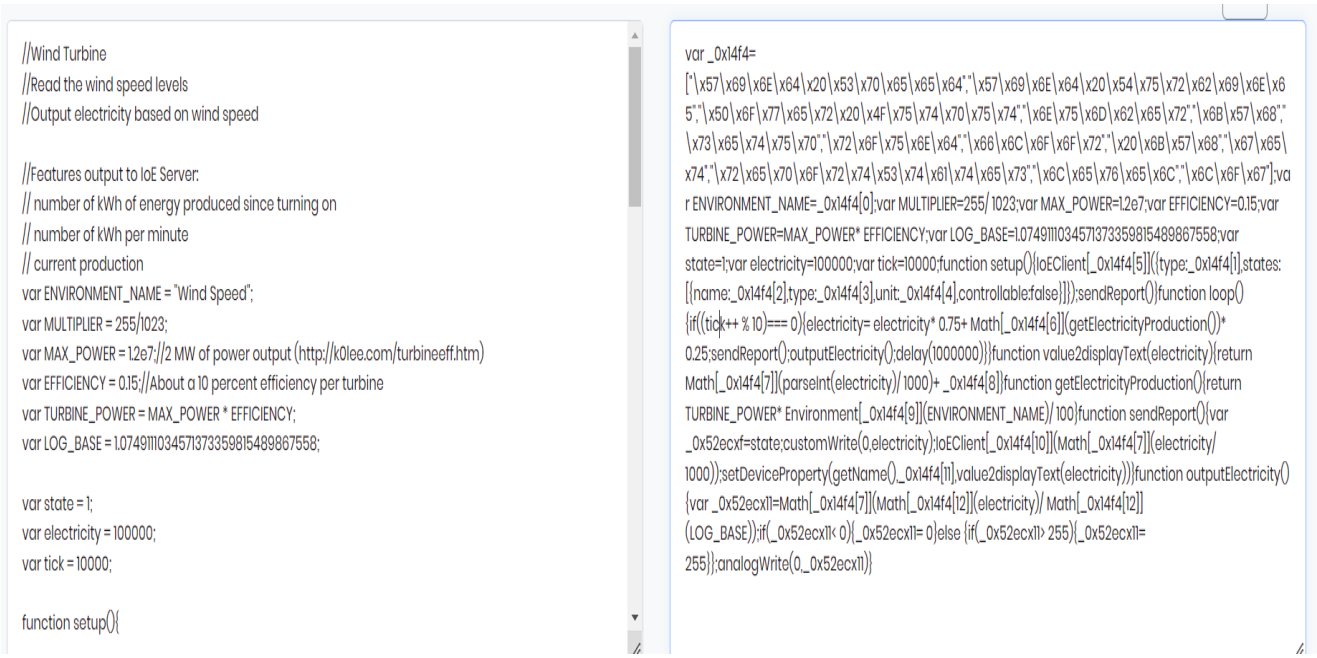


Рисунок 3.6 – Унеможливлюваня зчитування коду прошивки прототипу IoT пристрою

Ця операція була застосована для всіх IoT пристроїв в даній симуляції Cisco Packet Tracer. Це можна побачити на рисунку 3.7

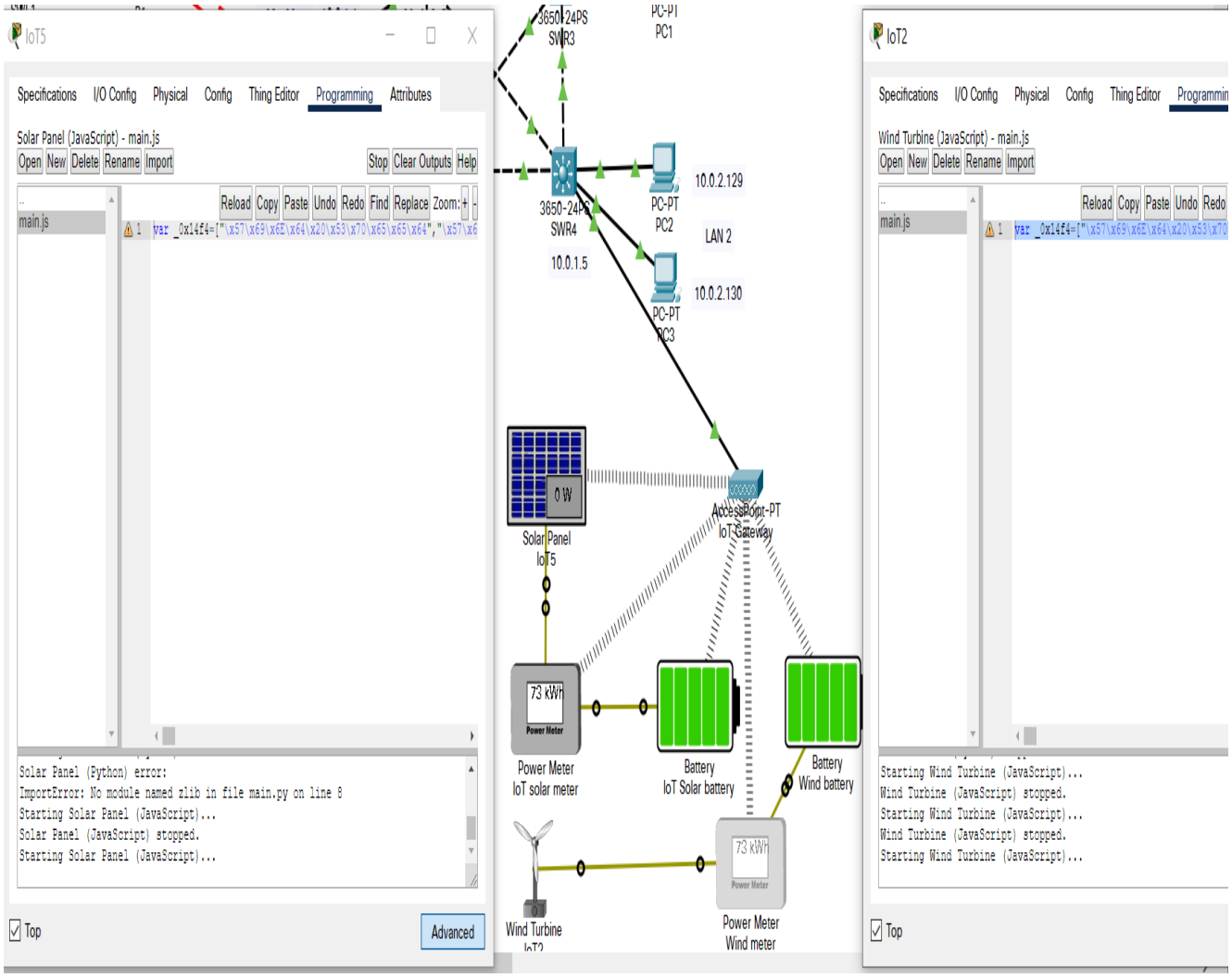


Рисунок 3.7 – Завантаження захищеної прошивки до прототипів IoT пристроїв

Перейменування імен змінних за допомогою випадково згенерованих текстових рядків, перетворення тексту коду на шістнадцятковий текст і поділ тексту для коду для об’єднання тексту – це деякі методи обфускації, які використовуються для таких сценаріїв, як JavaScript і сценаріїв Python.

Деякі програми мають зашифровані текстові рядки, включаючи назву функцій API, і розшифровуються під час виконання перед виконанням динамічного імпорту. Це запобігає таким інструментам, як StringsorBinText, від побудови переліку API, які прошивка пристрою може використовувати.

3.3 Надання рекомендацій щодо проведення оцінки впливу конфіденційності PIA в IoT

Проблеми з конфіденційністю в IoT величезні, враховуючи величезну кількість даних, що збираються, поширюються, зберігаються. Збільшення обсягу та типів даних, які можна зібрати та дистилювати за допомогою технічних і бізнес-аналітичних систем, може створювати лякаюче детальні та точні профілі кінцевих користувачів

Захист конфіденційності є проблемою для кожної організації та галузі. Комунікації в організації, яка піклується про конфіденційність і захищає конфіденційність, життєво важливі для забезпечення врахування інтересів клієнтів. Далі в цьому останньому пункті розділу буде визначено корпоративні відділи та індивідуальні кваліфікації, необхідні для вирішення політики конфіденційності та розробки конфіденційності IoT.

IoT PIA має вирішальне значення для розуміння того, як пристрої IoT у контексті більшої системи або системи систем можуть впливати на конфіденційність кінцевого користувача. PIA необхідні для забезпечення якомога повнішого аналізу ризиків. Окрім базових принципів безпеки, непомірна втрата конфіденційності може мати суттєвий вплив і призвести до серйозних фінансових або юридичних наслідків для виробника чи оператора IT-систем та систем Інтернету речей. Наприклад, можна розглянути дитячу іграшку, оснащену можливостями Wi-Fi, керування смартфоном і підключенням до внутрішніх серверів системи. Припустити, що іграшка має мікрофон і динамік, а також можливості захоплення та розпізнавання голосу.

Тепер потрібно розглянути функції безпеки пристрою, зберігання конфіденційних параметрів автентифікації та інші атрибути, необхідні для безпечного зв’язку з серверними системами. Якби пристрій було фізично або логічно зламано, чи розкрив би він будь-які загальні або типові параметри безпеки, які можна було б використати для компрометації інших іграшок того ж виробника? Чи належним чином захищені комунікації за допомогою шифрування, автентифікації та контролю цілісності? Чи вони повинні бути? Яка природа даних

і що вони можуть містити? Чи збираються дані користувача у внутрішніх системах для будь-якої аналітичної обробки? Чи достатньо загальної безпеки інфраструктури та процесу розробки для захисту споживачів?

Ці всі запитання потрібно ставити в контексті PIA. Запитання мають стосуватися серйозності впливу від витоку інформації або неправомірного використання інформації після того, як вона надходить на пристрій і серверні системи. Наприклад, чи можливо буде захопити звукові команди дитини та почути імена та іншу особисту інформацію? Чи може зловмисник визначити геолокацію трафіку, потенційно розкриваючи місцезнаходження дитини? Якщо так, то вплив може включати злісне переслідування дитини або членів сім'ї.

Іншим гідним прикладом необхідності перегляду конфіденційності для IoT є ринок розумних автомобілей. Так само, як і у випадку з переносними пристроями, про які йшлося раніше, здатність постійно відстежувати чийсь транспорт викликає занепокоєння.

Повідомлення мають бути захищені цілісністю та перевірені на те, щоб походити з надійних джерел. У деяких випадках вони також повинні бути захищені конфіденційністю. Транспортна галузь розробляє рішення конфіденційності для підключених транспортних засобів. Наприклад, коли підключений транспортний засіб передає повідомлення, існує занепокоєння, що використання одних і тих самих облікових даних для підпису повідомлень протягом певного періоду часу може призвести до постійного відстеження автомобіля та власника. Щоб боротися з цим, інженери безпеки вказали, що транспортні засоби будуть забезпечені сертифікатами, які:

- Мають коротку тривалість існування;
- Надаються пакетами, щоб дозволити використовувати пул облікових даних для операцій підписання.

У розумному транспортному середовищі транспортним засобам буде надано великий пул сертифікатів псевдонімів, які постійно змінюються, щоб підписувати повідомлення, що передаються пристроями бортового обладнання (OBE) у транспортному засобі. Цей пул сертифікатів може бути дійсним лише протягом

тижня, після чого інша партія набуде чинності для наступного періоду часу. Це зменшує можливість відстежувати місцезнаходження транспортного засобу протягом дня, тижня або будь-якого більшого періоду часу на основі сертифікатів, які він приєднав до своїх власних передач. На малюнку 3.8 можна побачити схему підключення розумних автівок з захищеними пристроями бортового обладнання.

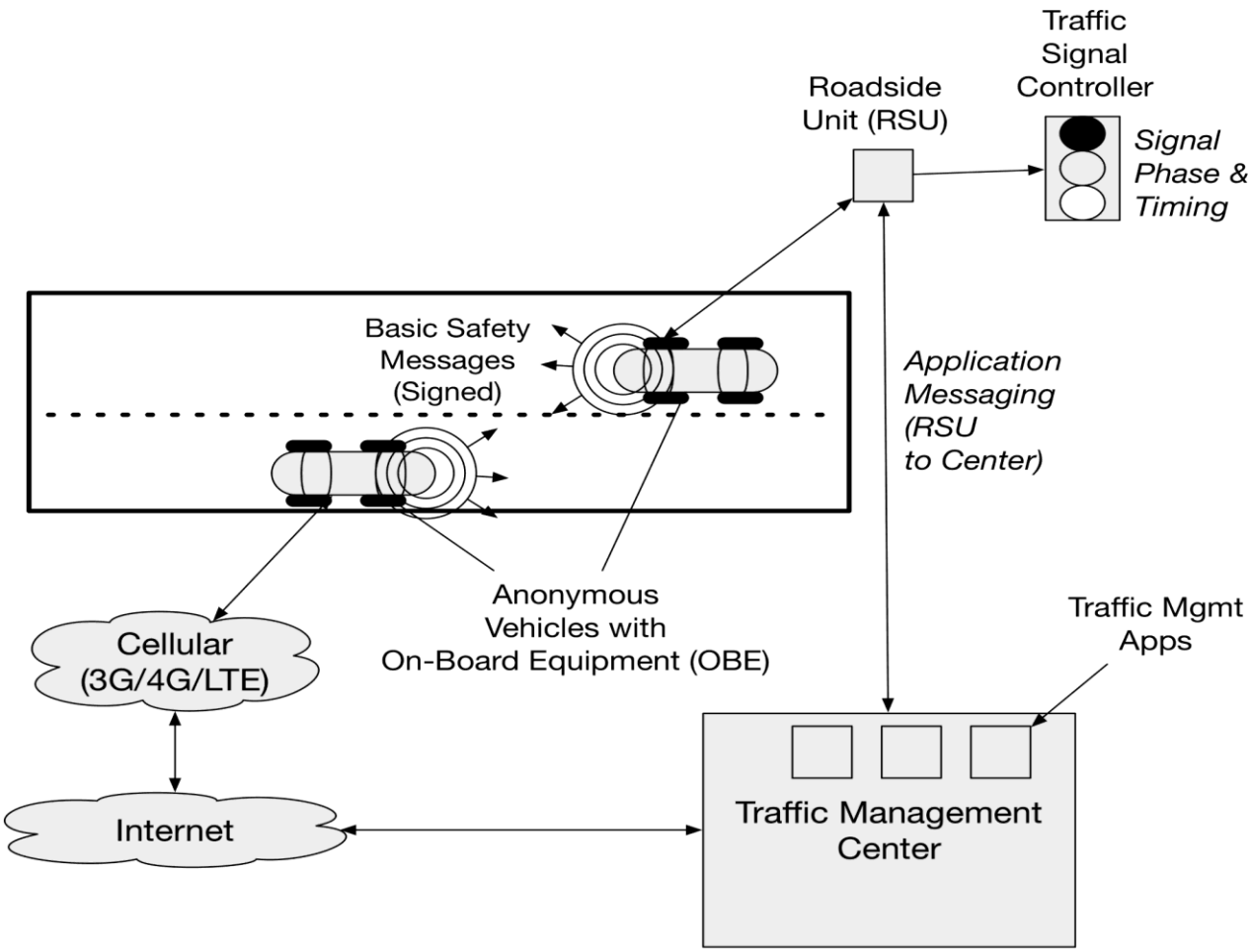


Рисунок 3.8 – Схема підключення розумного транспорту з захищеними пристроями бортового обладнання

За іронією долі, все більше транспортних відділів починають використовувати переваги широко поширених транспортних засобів і мобільних пристроїв, розгортаючи зонди Bluetooth уздовж перевантажених автомагістралей. Деякі дорожні агентства використовують зонди для вимірювання часу, який потрібен пристрою Bluetooth, який проїжджає повз (позначається його MAC-

адресою), щоб подолати задану відстань між встановленими на узбіччі зондами. Це надає дані, необхідні для адаптивного керування системою руху. Якщо служби дорожнього руху не будуть обережними та не знищать будь-яку коротко- чи довгострокову колекцію MAC-адрес Bluetooth, кореляційний аналіз даних потенційно можна використовувати для розпізнавання руху окремого транспортного засобу у регіоні.

Продовжуючи приклад із підключеним транспортним засобом, також видно, що оператори інфраструктури також не повинні мати можливість зіставляти надані сертифікати з транспортними засобами. Це вимагає змін у традиційному дизайні безпеки PKI, історично створеному для надання сертифікатів, які спеціально ідентифікують і автентифікують осіб і організації через розпізнане ім’я X.509, організацію, домен та інші типи атрибутів. У сфері підключених транспортних засобів PKI, який надаватиме облікові дані транспортним засобам у Сполучених Штатах, відомий як Система керування обліковими даними безпеки (SCMS) і зараз створюється для різних пілотних розгортань підключених транспортних засобів по всій країні. SCMS має вбудовані засоби захисту конфіденційності, починаючи від дизайну псевдоніму сертифіката IEEE 1609.2 і закінчуючи внутрішніми організаційними поділами, спрямованими на запобігання інсайдерським атакам PKI на конфіденційність драйверів.

Захист конфіденційності — це серйозне завдання, яке стає ще більш складним через безліч форм систем IoT, незліченних організацій і відмінностей. Крім того, гігантська кількість даних, що збираються, індексуються, аналізуються, перерозподіляються, повторно аналізуються та продаються, створює проблеми для контролю власності на дані, подальшої передачі та прийнятного використання. У цьому останньому пункті даного розділу було надано рекомендації по впровадженню принципів конфіденційності, розробку конфіденційності та те, як виконувати оцінку впливу на конфіденційність для підтримки розгортання IoT.

Висновки до розділу 3

У цьому розділі було спочатку проаналізовано впровадження найкращих практик та традиційних рекомендацій кібербезпеки для забезпечення безпеки мереж де використовуються IoT пристрої та забезпечення безпеки їх самих. Було визначено та надано конкретні рекомендації по розгортання систем розумних та підключених пристроїв. Зокрема, основна увага була приділена використанню апаратного забезпечення, яке включає такі функції безпеки, як Trusted Execution та застосуванню середовища та модулей довіреної платформи. Достатньо уваги було приділено рекомендації по унеможливленню декомпіляціх коду прошивки IoT пристроїв та коду запущених служб та застосунків на розумних пристроях. Також було розглянуто проблеми конфіденційності в екосистемах IoT та належні методи захисту. Більшу частину останнього пункту було присвячено рекомендаціям по унеможливленню відстеження розумних автомобілей та побутових розумних пристроїв, іграшок чи тощо.

ВИСНОВКИ

Еволюція інформаційних технологій значним чином сприяє покращенню та розвитку впровадження розумних рішень в різних сферах нашого життя. Але одночасно з розвитком виникають і проблеми з безпекою цих рішень. Тому головною ціллю даної кваліфікаційної роботи було дослідити сферу захисту мереж до яких підключаються розумні пристрої. Хоча майбутнє кібербезпеки завжди буде непередбачуваним для світових лідерів та корпорацій, надзвичайно важливо підготувати оцінку можливих загроз і потенційних інновацій у сфері безпеки, щоб зберегти постійну довіру клієнтів і зацікавлених сторін.

У цій роботі було:

1. Зроблене дослідження та аналіз предметної області, а саме в першому розділі було досліджено еталонну модель екосистеми IoT на предмет забезпечення кібербезпеки на кожному рівні. Також було розглянуто криптографічні методи захисту зв’язку захисту вбудованого програмного забезпечення та автентифікації пристроїв. Було поставлене завдання та визначено подальші кроки наступних розділів. Визначені основні методи забезпечення безпеки для екосистем IoT, такі як Blockchain-based та SDP поверх SDN.
2. Реалізовано та розгорнуто приклад приватного блокчейну на основі рішення Hyperledger Fabric для IoT. Також було проведено аналіз реалізації рішення приватної інфраструктури ключів PKI для IoT на основі блокчейну. Блокчейн Ethereum покращив безпеку ключів в дуже багато разів та зробив її надійнішою.
3. Також в середині даної роботи було реалізовано прототип програмно визначеної мережі в ПЗ Packet Tracer. Що дозволило зробити симуляцію розмежування програмно визначених периметрів безпеки SDP поверх SDN для різних IoT пристроїв. Для цього було описано розгортання прототипу даної мережі крок за кроком.

4. Застосовано прототип контролеру SDN, який мав зв’язок з реальною локальною мережою в якій був запущений Packet Tracer, що значно спростило програмування даної мережі та пристроїв IoT.
5. Розглянуто застосування Cisco DNA серверу та досліджено технологію Cisco SDA для створення програмно визначених периметрів безпеки.
6. У останньому розділі даної роботи було надано конкретні рекомендації по кіберзахисту даних мереж. А саме унеможливленню декомпіляції коду прошивки IoT пристроїв та коду запущених клієнтських застосунків та служб для них.
7. Надано рекомендації по унеможливленню відстеження розумних автівок та пристроїв. А також розглянуто проблеми конфіденційності розумних побутових пристроїв та їх вирішення. Особливу увагу було приділено методу забезпечення сертифікатами, які мають не довгу тривалість існування та змінюються.

ПЕРЕЛІК ПОСИЛАНЬ

1. Alasdair Gilchrist IoT Security Issues [Електронний ресурс]. – Режим доступу:https://books.google.com.ua/books?id=xipDDgAAQBAJ&printsec=frontcover&dq=iot+security&hl=uk&sa=X&redir_esc=y#v=onepage&q=iot%20security&f=false

2. Adrienne Raglin Anshuman Venkateswaran Huan Liu Abductive Causal Reasoning for Internet of Things [Електронний ресурс]. – Режим доступу: https://ieeexplore.ieee.org/document/9060242

3. Hristo Kotselov Exploring the security and privacy issues arising from the Internet of Things [Електронний ресурс]. – Режим доступу:https://www.academia.edu/35949041/Exploring_the_security_and_privacy_issues_arising_from_the_Internet_of_Things

4. Madhusanka Liyanage, An Braeken, Pardeep Kumar IoT Security: Advances in Authentication [Електронний ресурс]. – Режим доступу: https://books.google.com/books?id=Bk6_DwAAQBAJ&printsec=frontcover&dq=iot+security&hl=uk&sa=X&ved=2ahUKEwjc7rrdh_r7AhUrlosKHayCA9sQ6AF6BAgKEAI

5. Sunil Cheruvu, Anil Kumar, Ned Smith Demystifying Internet of Things Security: Successful IoT Device/Edge and Platform Security Deployment [Електронний ресурс]. – Режим доступу: https://learning.oreilly.com/library/view//9781484228968/

6. Shancang Li. Li Da Xu Securing the Internet of Things [Електронний ресурс]. – Режим доступу: https://learning.oreilly.com/library/view/-/9780128045053/

7. Han Liu Dezhi Han Dun Li Fabric-iot: A Blockchain-Based Access Control System in IoT [Електронний ресурс]. – Режим доступу: https://ieeexplore.ieee.org/document/8964343

8. Nehemiah Adebayo Amos O. Bajeh Blockchain Technology: A Panacea for IoT Security Challenge. [Електронний ресурс]. – Режим доступу: https://publications.eai.eu/index.php/IoT/article/view/1402

9. Bela Shrimali Hiren B.Patel Blockchain state-of-the-art: architecture, use cases, consensus, challenges and opportunities. [Електронний ресурс]. – Режим доступу: https://www.sciencedirect.com/science/article/pii/S131915782100207X

10. Saman M. Omer Kamaran Hussein Manguri SDN for IoT Environment: A Survey and Research Challenges. [Електронний ресурс]. – Режим доступу: https://www.researchgate.net/publication/358833611_SDN_for_IoT_Environment_A_Survey_and_Research_Challenges

11. Sahrish Khan, Munam Ali Shah Software Defined Network (SDN) Based Internet of Things (IoT): A Road Ahead. [Електронний ресурс]. – Режим доступу: https://www.researchgate.net/publication/319602888_Software_Defined_Network_SDN_Based_Internet_of_Things_IoT_A_Road_Ahead

12. Securing Internet of Things (IoT) devices using Hyperledger Fabric (Blockchain technology). [Електронний ресурс]. – Режим доступу: https://www.researchgate.net/publication/344629764_Securing_Internet_of_Things_IoT_devices_using_Hyperledger_Fabric_Blockchain_technology

13. Antonis Mygiakis Alexander Papageorgiou DPKI: A Blockchain-Based Decentralized Public Key Infrastructure System. [Електронний ресурс]. – Режим доступу: https://www.researchgate.net/publication/342249361_DPKI_A_Blockchain-Based_Decentralized_Public_Key_Infrastructure_System

14. Getting Started with Hyper Ledger Fabric. [Електронний ресурс]. – Режим доступу: https://www.hyperledger.org/use/fabric

15. . Getting Started with Etherum Blockchain. [Електронний ресурс]. – Режим доступу: https://ethereum.org/uk/

16 Alberto Rodriguez-Natal SD-Access: Practical Experiences in Designing and Deploying Software Defined Enterprise Networks. [Електронний ресурс]. – Режим доступу: https://www.researchgate.net/publication/344972197_SD-Access_Practical_Experiences_in_Designing_and_Deploying_Software_Defined_Enterprise_Networks

17. Packet Tracer 7.x - Internet of Things tutorials. [Електронний ресурс]. – Режим доступу: https://www.packettracernetwork.com/internet-of-things/