

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Кіберзахист об'єктів критичної інфраструктури

ЛАБОРАТОРНИЙ ПРАКТИКУМ

*Рекомендовано Методичною радою КПІ ім. Ігоря Сікорського
як навчальний посібник для здобувачів ступеня магістра за освітніми програмами
«Системи, технології та математичні методи кібербезпеки»,
«Системи технічного захисту інформації»
спеціальності 125 «Кібербезпека та захист інформації»*

Київ
КПІ ім. Ігоря Сікорського
2023

Назва: Кіберзахист об'єктів критичної інфраструктури. Лабораторний практикум [Електронний ресурс] : навч. посіб. для студ. спец. 125 «Кібербезпека та захист інформації» / КПІ ім. Ігоря Сікорського; уклад.: І.В. Стьопочкина, К.І. Ійн. – Електронні текстові дані (1 файл: 6,1 Мбайт). – Київ : КПІ ім. Ігоря Сікорського, 2023. – 59 с.

*Гриф надано Методичною радою КПІ ім. Ігоря Сікорського (протокол № [] від [] р.)
за поданням Вченої ради Фізико-технічного інституту (протокол № [] від [] р.)*

Електронне мережне навчальне видання

Кіберзахист об'єктів критичної інфраструктури

Лабораторний практикум

Укладачі: *Стьюопочкина Ірина Валеріївна, канд. техн. наук,
Ільїн Костянтин Іванович*

Відповідальний
редактор *Смирнов Сергій Анатолійович, к.ф.-м.н., с.н.с.*

Рецензент *Яковлев Сергій Володимирович, к.т.н., доцент кафедри
математичних методів захисту інформації*

Навчальний посібник надає рекомендації щодо виконання лабораторних робіт із дисципліни “Кіберзахист об'єктів критичної інфраструктури.” Тематика робіт стосується симуляції кіберфізичних атак із використанням Sooja, для пристрій, що працюють на основі Kontiki OS, дослідженю вразливостей популярних протоколів PoT, оцінюванню ризиків для об'єктів критичної інфраструктури. Також надано вказівки для альтернативних робіт, які рекомендовані в разі наявності відповідного апаратного забезпечення (Raspberry Pi, датчиків температури та вологості, апаратного сніфера Ubertooth), які сприяють поглибленню опануванню дисципліни. Наведено теоретичні відомості та вказівки щодо ходу виконання роботи, а також варіанти завдань. Це допомагає застосувати відповідний матеріал на власному досвіді.

© КПІ ім. Ігоря Сікорського, 2023

ЗМІСТ

ЗМІСТ	4
ВСТУП	6
Лабораторний практикум №1	7
Оцінювання дієвості політики безпеки на основі аналізу динаміки ризиків	7
1. Теоретичні відомості	7
2. Завдання на роботу	8
3. Варіанти	9
4. Контрольні питання	10
Лабораторний практикум №1 (альтернативний)	11
Отримання даних за використанням Raspberry Pi, DHT22, MQTT	11
1. Теоретичні відомості	11
2. Завдання на роботу	19
3. Варіанти завдань	21
4. Контрольні питання	21
Лабораторний практикум №2	22
Оцінювання вразливостей об'єкту критичної інфраструктури на основі показника CVSS	22
1. Теоретичні відомості	22
2. Завдання на роботу	26
3. Варіанти завдань	26
4. Контрольні питання	27
Лабораторний практикум №2 (альтернативний)	28
Перехоплення bluetooth сигналів за допомогою Ubertooth One	28
1. Теоретичні відомості	28
2. Завдання на роботу	28
3. Вимоги до звіту	31
4. Варіанти завдань	34
5. Контрольні питання	34
Лабораторний практикум №3	35
Використання симулатора Сооja для моделювання об'єктів критичної інфраструктури	35
1. Теоретичні відомості	35
2. Завдання на роботу	36
3. Вимоги до звіту	37
4. Варіанти	38
5. Контрольні питання	38
Лабораторний практикум №4	39
Використання симулатора Сооja для моделювання атак на об'єкти критичної інфраструктури	39
1. Теоретичні відомості	39
2. Завдання на роботу	40
3. Вимоги до звіту	41
4. Варіанти	41
5. Контрольні питання	42
Лабораторний практикум №5	43
Вдосконалення недоліків протоколів інтернету речей	43
1. Теоретичні відомості	43
2. Завдання на роботу	48
3. Вимоги до звіту	48
4. Варіанти	48
5. Контрольні питання	48

Лабораторний практикум №5 (альтернативний).....	49
Впровадження MQTT з протоколом безпеки транспортного рівня використовуючи бібліотеку OpenSSL	49
1. Теоретичні відомості	49
2. Вимоги до звіту	55
3. Варіанти завдань	58
4. Контрольні питання.....	58
Література.....	59

ВСТУП

Сучасні об'єкти критичної інфраструктури характеризуються широким впровадженням кіберфізичних пристройів, які здатні виконувати сухо фізичні функції, такі, як вимірювання показників температури, тиску, вологості – так і взаємодіяти із багатофункціональними пристроями із використанням мережних протоколів, як кібернетичні пристрої. Ці пристрої називаються кіберфізичними, і їхні властивості висувають багато викликів в сфері кібернетичної безпеки. В даному посібнику представлено підходи до аналізу загроз та вразливостей кіберфізичних систем та реалізації засобів кіберзахисту. В матеріалах практикуму використано досвід Florida International University, фахівці якого провадили тренінги в сфері кібербезпеки в рамках ініціативи USAID для навчальних закладів України, де взяли участь укладачі навчального посібника.

Лабораторний практикум складається із 5 робіт, до трьох із яких передбачено альтернативні, які обираються студентами у випадку наявності відповідних апаратних пристройів.

В лабораторному практикумі 1 розглянуто методику оцінки дієвості політики безпеки на основі динаміки ризиків. Наводяться критерії дієвості, відповідна методика розрахунків за якими має діяти здобувач.

В альтернативному лабораторному практикумі розглянуто технічні концепції одержання даних, необхідних для моніторингу безпеки із використанням пристройів Raspberry Pi, датчиків DHT22 із використанням протоколу MQTT, розглядаються ризики, які притаманні цьому виду передачі даних.

В лабораторному практикумі 2 розглянуто методику оцінювання вразливостей об'єкту критичної інфраструктури на основі показника CVSS. Описано загрози, типові для об'єктів критичної інфраструктури.

В альтернативному лабораторному практикумі надано основи переходоплення Bluetooth сигналів із використанням апаратного сніфера Ubooth One, що дозволяє провадити тестування та оцінювання захищеності системи.

В лабораторному практикумі 3 надано методику використання симулятора Сооja для моделювання складових мереж об'єктів критичної інфраструктури.

В лабораторному практикумі 4 показано можливості використання Сооja для моделювання атак на складові об'єктів критичної інфраструктури, зокрема, безпровідні мережі сенсорів.

Лабораторний практикум №5 присвячений криптографічним засобам захисту в складі протоколів інтернету речей. Здобувачі мають змогу обґрунтівувати вибір та виконати впровадження криптографічних засобів захисту в склад одного із протоколів інтернету речей – протоколу Modbus.

Альтернативний практикум дає можливість впровадити використання криптографічних засобів в склад протоколу прикладного рівня MQTT, який використовується при передачі даних в PoT.

1. Лабораторний практикум №1

Оцінювання дієвості політики безпеки на основі аналізу динаміки ризиків

Мета роботи: опанувати підходи до оцінки дієвості політики безпеки об'єкта критичної інфраструктури. Реалізувати на практиці підхід щодо аналізу поведінки ризиків.

1.1. Теоретичні відомості

В якості критеріїв дієвості політики безпеки можна використати наступні показники [1,2]:

- 1) значення найбільш суттєвих ризиків за період оцінювання впровадженої політики (спад чи зростання);
- 2) яким чином позначаються фінансові вкладення на поведінці ризиків; чи є між ними кореляція;
- 3) чи враховано вимоги наявних чеклістів з безпеки;
- 4) якими є результати тестування на проникнення за умов впровадженої політики безпеки;
- 5) наскільки якісно виконуються організаційні заходи із забезпечення політики безпеки, ініційовані керівництвом об'єкта критичної інфраструктури та вимогами законодавства;
- 6) чи представляється моделювання поведінки мережі успішним за наявних налаштувань політики безпеки (для політики безпеки, яка ще не впроваджена на об'єкті)

Розглянемо підхід до аналізу поведінки ризиків впродовж певного періоду на основі найпростішої моделі – лінійної регресії [3].

Основні визначення та співвідношення

Регресією називається зміна результату в залежності від зміни незалежних факторів (аргументів).

Використаємо для прогнозу частоти певних загроз, за якими ми досліджуємо ризик, рівняння лінійної регресії виду $y = ax + b + \varepsilon$, де y – результуюча змінна, x – незалежна змінна (аргумент), a та b – параметри регресії, ε – похибка. Зміст y наступний – це частота атаки, яку треба підрахувати із спостережень за системою за період оцінки (наприклад, на основі реєстрації у журналах системи захисту).

Теоретично, в якості y можуть бути і інші показники (величина збитків від зіпсованих матеріальних активів, репутаційні збитки та ін.). Для розрахунку параметрів регресії a і b використовують метод найменших квадратів (МНК).

Методика роботи з регресійною моделлю:

1) Сформувати таблицю на основі спостережень за таблиці зі статистичними даними, де часу спостереження (змінна x) ставиться у відповідність значення показника, в даному випадку – кількості атак.

2) Розрахувати параметри лінійної регресії:

$$a_{\text{МНК}} = \frac{\text{cov}_{x,y}}{\sigma^2} \quad (1)$$

$$\text{та } b = \bar{y} - a_{\text{МНК}}. \quad (2),$$

де

$$\bar{x} = \frac{\sum_{i=1}^T x_i}{T} - \text{середнє значення } x, \quad (3)$$

$$\bar{y} = \frac{\sum_{i=1}^T y_i}{T} - \text{середнє значення } y, \quad (4)$$

$$\sigma^2 = \frac{\sum_{i=1}^T (x_i - \bar{x})^2}{T-1} - \text{дисперсія}, \quad (5)$$

$$\text{cov}_{x,y} = \frac{\sum_{i=1}^T (x_i - \bar{x})(y_i - \bar{y})}{T-1} - \text{коваріація змінних } x \text{ та } y. \quad (6)$$

3) Перевірити адекватність використаної моделі. Перевірка включає обчислення коефіцієнту детермінації R^2 та похибки ε :

$$R^2 = \frac{\sum_{i=1}^T (y_i^* - \bar{y})^2}{\sum_{i=1}^T (y_i - \bar{y})^2}, \quad (7)$$

$$\varepsilon = \frac{1}{T} \cdot \sum_{i=1}^T \left| \frac{y_i - y_i^*}{y_i} \right| \cdot 100, \quad (8)$$

$$\text{де } y_i^* = ax_i + b. \quad (9)$$

Про адекватність використання лінійної залежності (регресії) свідчить коефіцієнт детермінації близький до одиниці, який сигналізує про наявність лінійного зв'язку між x та y .

4) Оцінити точність моделювання залежності кількості атак від часу за допомогою похибки апроксимації ε . Моделювання здійснюється на відрізку часу функціонування об'єкту критичної інфраструктури із певною політикою безпеки.

1.2. Завдання на роботу

- Використовуючи таблицю відповідно до варіанту, побудувати залежність у вигляді лінійної регресії (графік) та обчислити відповідні параметри (1)-(6). Обчислити коефіцієнт детермінації та

похибку апроксимації (7), (8), зробити висновки про адекватність моделювання та точність.

2. Зробити висновки про динаміку ризиків: зростаюча, спадаюча; про що це свідчить.
3. Зробити висновки про дієвість політики безпеки, та фактори, які можуть впливати на ефективність заходів захисту.
4. Дати рекомендації, які заходи треба підсилити для покращення ситуації.
5. Результати оформити у вигляді звіту, для перевірки розуміння матеріалу використати контрольні запитання.

1.3. Варіанти

Тут x - місяць від початку використання політики безпеки, дієвість якої оцінюється, y- кількість атак в поточному місяці.

1. y - кількість попереджених атак.

x	1	2	3	4	5	6	7	8	9	10	11	12
y	5	7	12	13	16	32	18	20	24	18	26	25

2. y - кількість успішних атак DDoS.

x	2	3	4	5	6	7	8	9	10	11	12	13
y	24	10	20	14	25	20	19	18	16	15	14	10

3. y - кількість зареєстрованих атак.

x	1	2	3	4	5	6	7	8	9	10	11	12
y,	6	5	11	13	15	20	24	25	24	26	28	19

4. y – кількість виявлених несанкціонованих використань зйомних пристройів.

x	1	3	5	7	9	11	13	15	17	19	21	23
y	25	20	24	19	22	18	20	21	15	16	14	10

5. y - кількість успішних атак з використанням шкідливого ПЗ.

x	1	3	5	7	9	11	13	15	17	19	21	23
	5	6	7	1	2	3	5	4	3	2	1	0

6. y- кількість збоїв обслуговування внаслідок організаційних порушень.

x	1	3	5	7	9	11	13	15	17	19	21	23
y	27	20	16	17	14	13	12	8	5	4	11	2

7. у - кількість зареєстрованих DDoS атак.

x	1	3	5	7	9	11	13	15	17	19	21	23
y	2	5	7	11	14	10	8	13	14	16	18	15

8. у - кількість успішних DDoS атак.

x	1	3	5	7	9	11	13	15	17	19	21	23
y,	3	5	8	10	16	17	18	16	20	17	21	20

9. у - кількість успішних атак з використанням шкідливого ПЗ.

x	1	3	5	7	9	11	13	15	17	19	21	23
y,	5	0	3	4	3	2	1	0	2	1	3	0

10. у - зареєстровані випадки спроб НСД.

x	1	3	5	7	9	11	13	15	17	19	21	23
y	7	4	3	6	3	2	1	0	2	1	2	5

1.4. Контрольні питання

- 1) Які види атак та яким чином можна зареєструвати?
- 2) Які види атак, на вашу думку, є найбільш типовими для об'єктів критичної інфраструктури.
- 3) Яким чином перевіряється адекватність та точність моделювання?
- 4) У яких випадках не можна робити висновки про дієвість політики безпеки по регресійній моделі (навіть за умови коефіцієнту детермінації, близького до одиниці) ?
- 5) Які фактори впливають на дієвість політики безпеки?
- 6) Які додаткові критерії можна розглянути при оцінці дієвості заданої політики безпеки.

2. Лабораторний практикум №1 (альтернативний)

Отримання даних за використанням Raspberry Pi, DHT22, MQTT

Мета роботи: ознайомити студентів із платформою Інтернету речей (Internet of Things, IoT) шляхом отримання певних даних за допомогою датчика із використанням протоколу MQTT.

Необхідні інструменти/програмне забезпечення

- Raspberry Pi 4 або 3;
- DHT22 Temperature Sensor (або DHT22);
- MQTT Communication Protocol ;
- ThingsBoard Platform.

2.1. Теоретичні відомості

Інтернет речей (IoT) - це система, яка складається з кіберфізичних об'єктів, які з'єднані Інтернетом, з'єднані між собою і здатні приймати і збирати дані по мережі без взаємодії з людиною-оператором. Однією з платформ, які можна використовувати як пристрій IoT, є Raspberry Pi. Тому для цієї лабораторної ми будемо використовувати Pi і температурний датчик для збору даних за допомогою одного з найпопулярніших протоколів зв'язку IoT MQTT [4]. Для того, щоб побачити зібрані дані, ми будемо використовувати платформу ThingsBoard, яка є платформою з відкритим вихідним кодом на стороні сервера, що дозволяє користувачеві контролювати та керувати пристрої IoT.

Налаштування пристрою та зразок вправи

Після налаштування Pi підключіть датчик DHT22, як показано на рисунку 2.1.

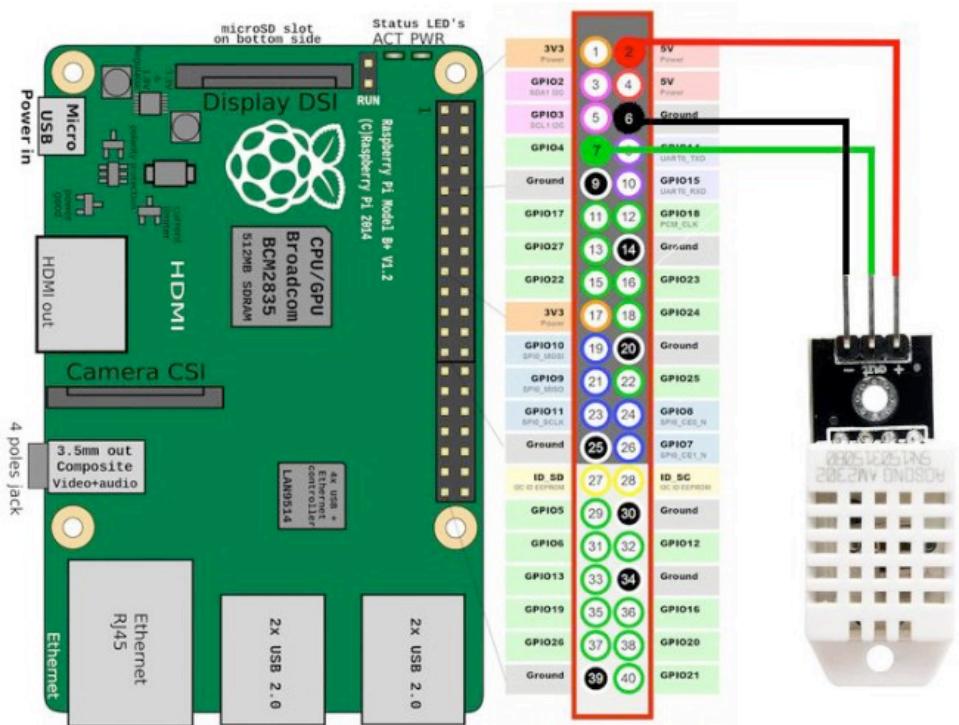


Рис.2.1. Схема підключення

Підключіть землю від датчика до GPIO 6 на RPi.

Підключіть живлення (+) від датчика до GPIO 2 (VCC 3-5V) на RPi.

Підключіть вихід даних датчика до GPIO 7 на RPi.

Потрібно встановити необхідні бібліотеки для датчика DHT та MQTT.

Для цього відкриваємо термінал і вводимо:

```
$ cd Desktop
$ mkdir lab1-mqtt
$ cd lab1-mqtt
$ sudo pip install paho-mqtt
$ sudo apt-get install mosquitto-clients
$ sudo apt-get install python-dev
$ git clone https://github.com/adafruit/Adafruit_Python_DHT.git
$ cd Adafruit_Python_DHT
$ sudo python setup.py install
$ cd ..
```

Завантажте сценарій python, що входить до лабораторного документа.

Помістіть його в папку lab1-mqtt.

```
$ ls
```

Ви повинні мати ці два каталоги всередині папки lab1-mqtt.

Adafruit_Python_DHT

lab1-mqtt

Створимо обліковий запис у ThingsBoard (рис.2.2).

Перейдіть за цим посиланням <https://demo.thingsboard.io/login>

The screenshot shows the registration form for ThingsBoard. It consists of several input fields and a reCAPTCHA section. At the bottom, there is a large orange 'Sign up' button and a link for existing users to sign in.

First name *

Last name *

Email *

Create a password *

Repeat your password *

I'm not a robot  reCAPTCHA
Privacy - Terms

Accept [Privacy Policy](#)

Sign up

Already have an account? [Sign in](#)

Рис.2.2. Реєстрація в ThingsBoard

Після створення облікового запису увійдіть, і ви повинні отримати щось подібне до рис. 2.3.

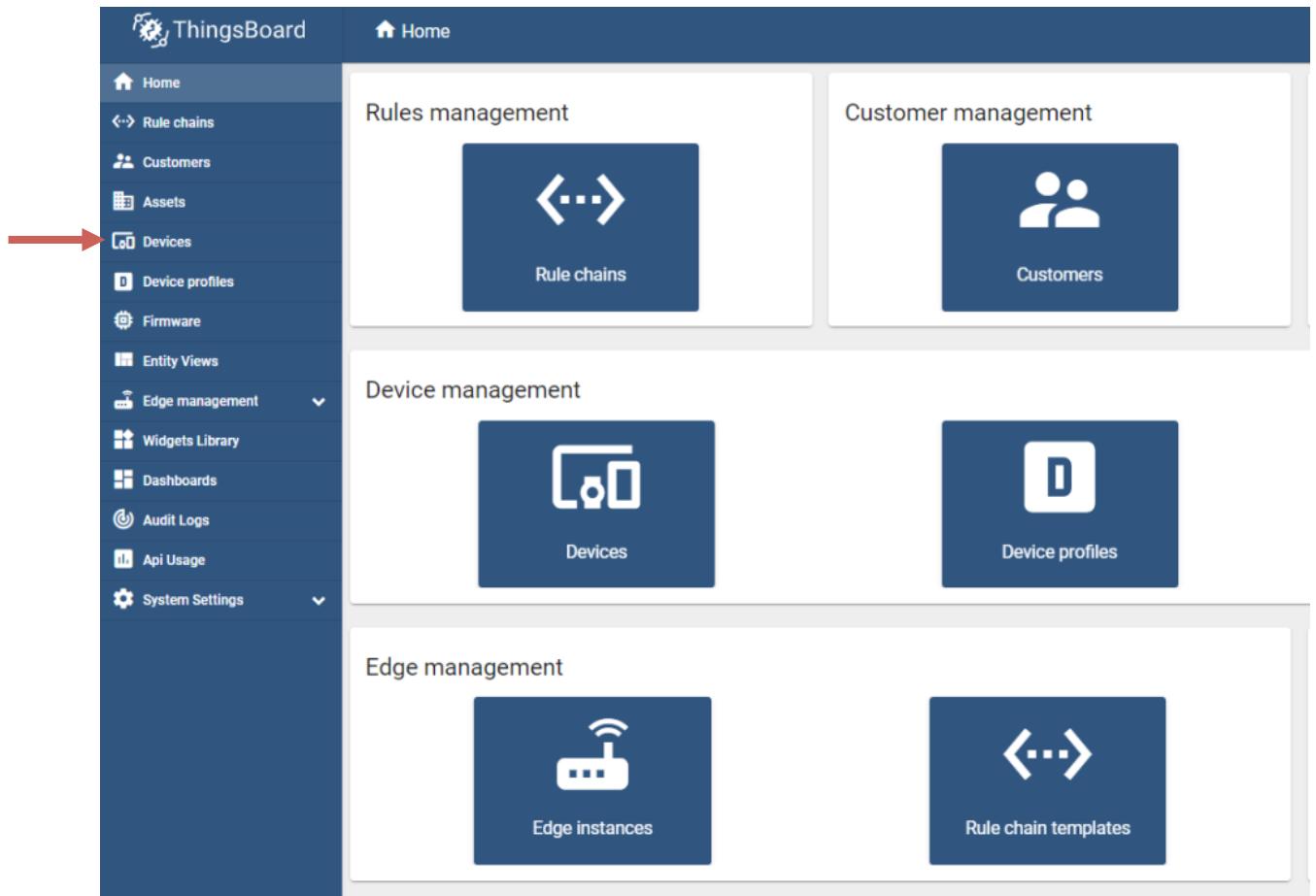


Рис.2.3. Інтерфейс платформи

Далі потрібно перевірити, чи можна надсилати дані на платформу ThingsBoard з RPi.

Перейдіть на вкладку пристрой та додайте новий пристрій з назвою Test-DHT22 і натисніть додати (рис.2.4-2.5).

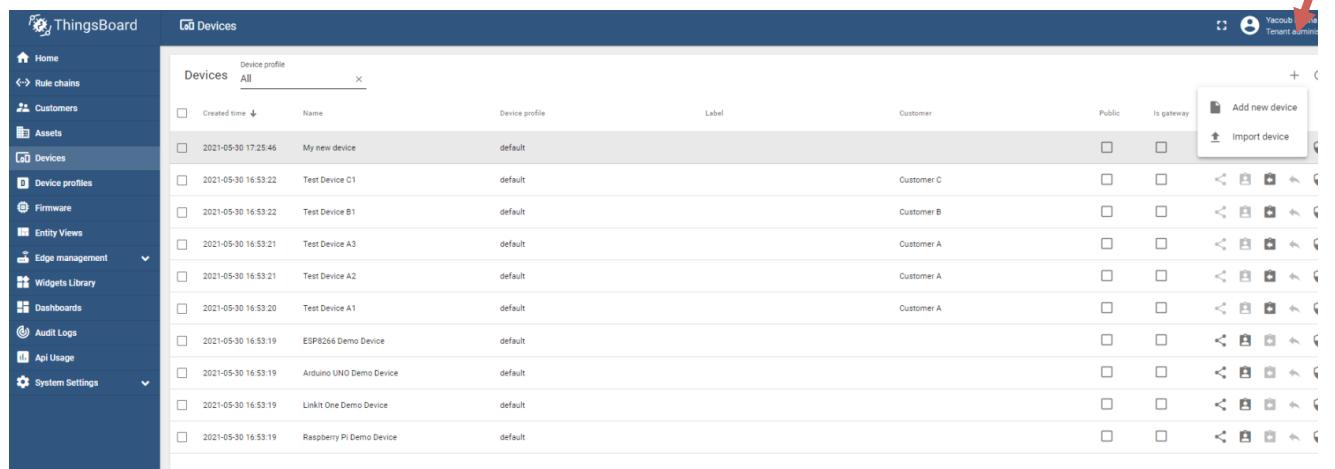


Рис.2.4. Додавання нового пристрою. Крок 1

Add new device

1 Device details **2 Credentials** **3 Customer**

Name *
Test-DHT22

Label

Transport type *
Default

Supports basic MQTT, HTTP and CoAP transport

Device profile *
 Select existing device profile default

Create new device profile

Is gateway

Description

[Next: Credentials](#)

[Cancel](#) [Add](#)



Рис. 2.5. Додавання нового пристрою. Крок 2

Після цього ми можемо побачити, що створено пристрій Test-DHT22 (але ви можете використовувати й датчик DHT11, зробивши відповідні зміни в коді скрипта).

Щоб підключитися до цього пристрою, нам потрібен маркер доступу, тому натисніть на Test-DHT22, і ви побачите щось подібне після натискання на Копіювати маркер доступу (рис. 2.6).

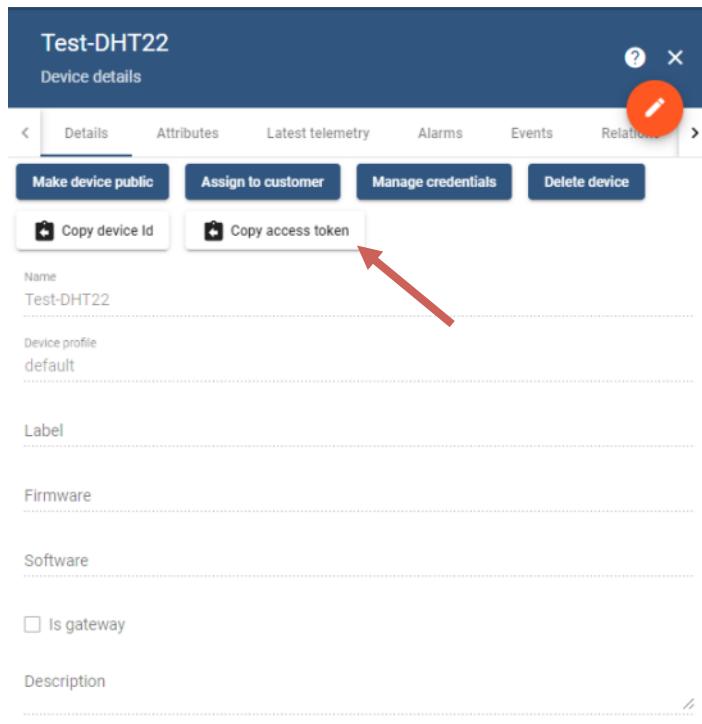


Рис. 2.6. Маркер доступу

Далі відкрийте термінал.

Примітка: Ви повинні замінити "маркер доступу" на свій скопійований маркер доступу.

```
$ mosquitto_pub -d -q 1 -h "demo.thingsboard.io" -p "1883" -t  
"v1/devices/me/telemetry" -u "access token" -m {"temperature":38}
```

Примітка: якщо ви отримали помилку, яка говорить, що порт уже використовується.

Виконайте цю команду.

```
$ sudo pkill mosquitto
```

І спробуйте ще раз:

```
$ mosquitto_pub -d -q 1 -h "demo.thingsboard.io" -p "1883" -t  
"v1/devices/me/telemetry" -u "access token" -m {"temperature":38}
```

Виконавши цю команду, ми публікуємо повідомлення за допомогою mosquitto-клієнта на порту “1883” на тему “v1/devices/me/telemetry” та використовуємо маркер доступу пристрою та публікуємо повідомлення {"temperature": 38}

Після успішного виконання цієї команди вивід має виглядати так.

Client mosqpub|xxx sending CONNECT

Client mosqpub|xxx received CONNACK

*Client mosqpub|xxx sending PUBLISH (d0, q1, r0, m1,
'vl/devices/me/telemetry', ... (16 bytes))*

Client mosqpub|xxx received PUBACK (Mid: 1)

Client mosqpub|xxx sending DISCONNECT

Після цього відкрийте пристрій Test-DHT22, потім натисніть Остання телеметрія, і ви побачите temperature:38

Ми маємо успішний зв'язок між RPi та ThingsBoard.

Тепер подивимося реальні дані датчика DHT22.

Перейдіть на вкладку Пристрої на ThingsBoard і знайдіть пристрій, який уже створено з назвою DHT22 Demo Device, клацніть на ньому та скопіюйте маркер доступу та виконайте ці дії.

Перейдіть до каталогу, який ви створили lab1-mqtt, там має розташовуватись скрипт python.

```
$ cd lab1-mqtt  
$ gedit mqtt-dht22.py
```

Ви зможете побачити весь код. Вставте скопійований маркер доступу туди, де він вказаний: 'DHT22_DEMO_TOKEN'

```
THINGSBOARD_HOST = 'demo.thingsboard.io'
```

```
ACCESS_TOKEN = 'DHT22_DEMO_TOKEN'
```

Збережіть файл, вийдіть і запустіть.

```
$ python3 mqtt-dht22.py
```

Щоб побачити температуру та вологість, перейдіть у ThingsBoard, потім - на вкладку «Інформаційні панелі», клацніть на DHT22: Temperature & Humidity Demo Dashboard, а потім відкрийте інформаційну панель (рис. 2.7).

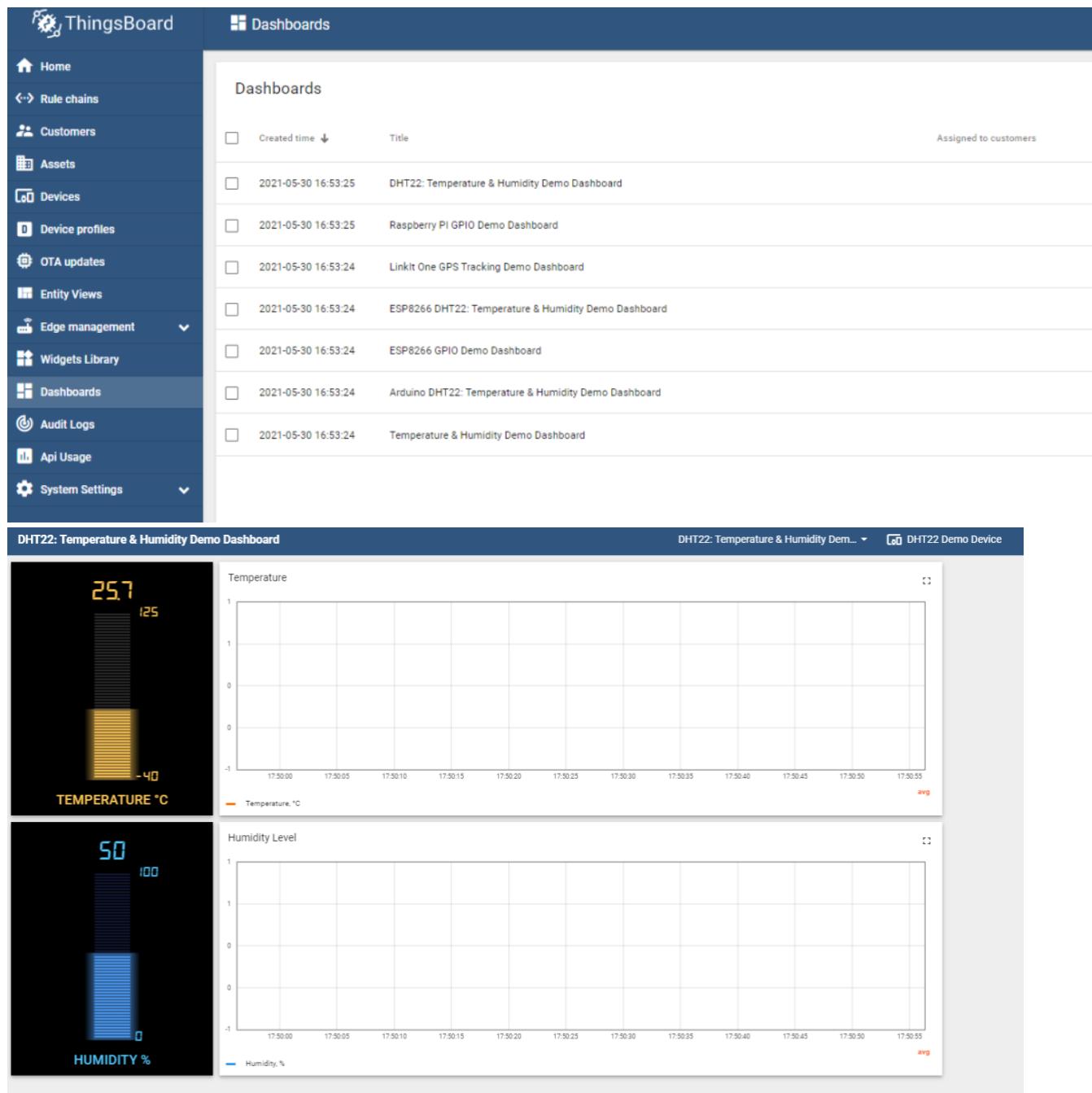


Рис. 2.7. Налаштування інформаційної панелі

2.2. Завдання на роботу

Необхідно продемонструвати наступні скріншоти за результатами роботи, в тому числі скріншоти з дашборда платформи інтернета речей, які ілюструють зміни вологості та температури вашого датчика (для цього попередньо треба налаштувати показ цих даних у вкладці Dashboard). – рис.2.8 - 2.9.

```
pi@raspberrypi:~/Desktop/lab1-mqtt/lab1-mqtt $ python mqtt-dht22.py
Temperature: 14°C, Humidity: 151%
Temperature: 28°C, Humidity: 49%
Temperature: 28°C, Humidity: 47%
Temperature: 28°C, Humidity: 47%
Temperature: 28°C, Humidity: 47%
Temperature: 28°C, Humidity: 47%
Temperature: 28°C, Humidity: 46%
Temperature: 28°C, Humidity: 45%
Temperature: 28°C, Humidity: 45%
Temperature: 28°C, Humidity: 44%
Temperature: 28°C, Humidity: 44%
Temperature: 28°C, Humidity: 44%
Temperature: 28°C, Humidity: 45%
Temperature: 28°C, Humidity: 38%
Temperature: 28°C, Humidity: 44%
Temperature: 28°C, Humidity: 44%
Temperature: 28°C, Humidity: 43%
Temperature: 28°C, Humidity: 45%
Temperature: 28°C, Humidity: 44%
Temperature: 28°C, Humidity: 44%
Temperature: 28°C, Humidity: 45%
Temperature: 28°C, Humidity: 45%
Temperature: 28°C, Humidity: 44%
Temperature: 28°C, Humidity: 45%
Temperature: 28°C, Humidity: 44%
```

Рис. 2.8. Показники вологості та температури від датчика

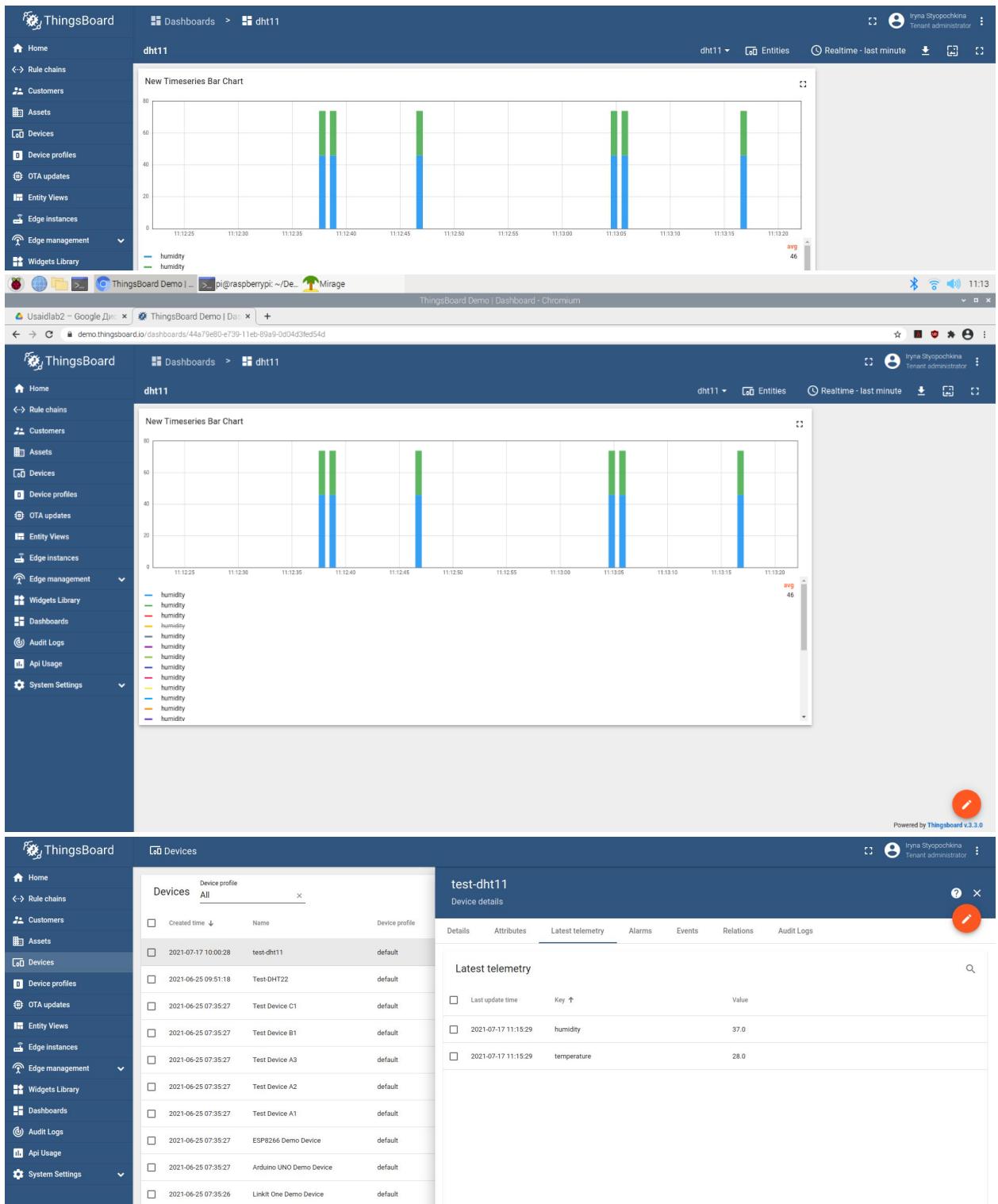


Рис. 2.9. Інтерфейс дашборда з показниками від датчиків

Також треба надати підтвердження того, що дані успішно передаються вами на платформу ThingsBoard (вкладка Остання телеметрія покаже відповідні передані вами значення) (рис. 2.10).

Рис. 2.10. Телеметрія

2.3. Варіанти завдань

Завдання виконується індивідуально, відповідно до показників використовуваного датчика.

2.4. Контрольні питання

1. Які протоколи прикладного рівня можуть використовуватись для передачі даних від сенсорів та датчиків до центру обробки?
2. Які можливості надає Raspberry Pi?
3. Чим edge computing відрізняється від fog computing, і, в свою чергу, від cloud computing? Які види обробки даних мають місце у даній лабораторній роботі?
4. Які переваги має парадигма edge computing?
5. Які виклики та небезпеки стоять перед протоколами передачі даних в ІoT?

3. Лабораторний практикум №2

Оцінювання вразливостей об'єкту критичної інфраструктури на основі показника CVSS

Мета роботи: ознайомитись із моделлю загроз, типовою для об'єктів критичної інфраструктури на прикладі промислового інтернету речей. Опанувати методику розрахунку показника CVSS на основі Common Vulnerability Scoring System Version 3.1 на прикладі загроз об'єкта критичної інфраструктури.

3.1. Теоретичні відомості

Розглянемо типову класифікацію загроз для об'єктів промислового інтернету речей.

Загрози ІоТ:

- Відмова в обслуговуванні (як наслідок зловмисної діяльності чи зловживання);
- Шкідливе ПЗ - ransomware, віруси, шпигунські програми та ін. (як наслідок зловмисної діяльності чи зловживання);
- Маніпуляція програмним та апаратним ПЗ (промисловими роботами, віддаленими контролерами пристрій, зміна конфігурації керуючих пристрій);
- Несанкціонована модифікація інформації та даних (спрямована на примушення системи (SCADA, MES (monitoring engineering system), Historian-сервер) генерувати невідповідні рішення, які базуються на сфабрикованих зловмисником даних);
- АРТ-атаки;
- Компрометація персональних даних (імена та ролі користувачів систем управління технологічними процесами);
- Brute-force атаки (особливо актуально для об'єктів Industry 4.0, де дозволено прості паролі, паролі за замовчуванням для пристрій);
- MITM-атаки;
- Атаки на вразливі протоколи ІоТ з метою використання даних зв'язку ІоТ (контроль наз сеансами зв'язку, розкриття паролів тощо);

- Сніффінг;
- Фізичні атаки (фізичне пошкодження пристрійв інсайдерами чи зловмисниками, які проникли на об'єкт);
- Підробка пристрійв (з потрібою зловмиснику конфігурацію) диверсантами та впровадження їх на об'єкт;
- Ненамисні порушення некомпетентними працівниками типу зміни конфігурації і підвищення привілеїв чи відкриття портів.
- Неправильне використання пристрійв ІoT і порушення їх роботи як наслідок;
- Пошкодження, заподіяні супровідниками (ненавмисно) чи зловмисниками. В Industry 4.0 до пристрійв інтернету речей може мати доступ “третя сторона”, зокрема для обслуговування та програмних поновлень. Часто цей доступ не контролюється у повній мірі.
- Несправність (поломка) кінцевих пристрійв ІoT;
- Несправність (поломка) складових системи управління (PLC, RTU, DCS);
- Експлуатація вразливостей ПЗ (або прошивки кінцевих пристрійв ІoT);
- Збої у процесах супроводу в разі несправностей.
- Простої внаслідок: а) відключення мережі зв'язку; б) відключення джерела живлення; в) втрати служб підтримки та логістики робот об'єкта (MES, CRM, ERP)
- Катастрофи природного, екологічного характеру.

Кожному об'єкту критичної інфраструктури можуть бути притаманні вразливості, які тягнуть за собою потенційні загрози із наведеного вище переліку. У числовому виді оцінити серйозність наявних вразливостей дозволяє числовий показник CVSS [5].

До метрик фреймворка CVSS 3.1 включено:

Base metrics (основні метрики):

Attack vector (показує, яким чином здійснює доступ до вразливого компонента атакуючий, наприклад мережею);

Attack complexity (умовна оцінка складності атаки за думкою експертів – низька, середня, висока);

Privileges required (рівень привілеїв, які потрібні для здійснення атаки);

User Interaction (чи потрібна взаємодія з користувачем, чи атака здійснюється автоматично);

Scope (вразливий компонент та мета атаки – чи це одне і те саме);
Confidentiality (який рівень порушень конфіденційності);
Integrity (який рівень порушень цілісності);
Availability (який рівень порушень доступності).

Можливі значення метрик наведено у супровідній документації фреймворка.

Крім наведених метрик також оцінка здійснюється із урахуванням допоміжних метрик:

Temporal metrics - оцінюють потужність наявних експлойтів, доступність коду, наявність патчів тощо, впевненість в описі вразливості, а саме:

Exploit Code Maturity (характеристики шкідливого коду, що може бути використаний для експлуатації вразливості);

Remediation Level (дає додаткову інформацію про стан вразливості, наявність патчів);

Report Confidence (дає інформацію по рівень впевненості щодо технічних деталей вразливості, зокрема чи підтверджена вона авторами технології чи продукту та ін.);

Environmental Metrics – ці метрики дозволяють аналітику налаштовувати показник CVSS залежно від важливості постраждалого ІТ-ресурсу для об'єкту критичної інфраструктури. Вимірювання відбувається з точки зору додаткових/альтернативних механізмів безпеки, конфіденційності, цілісності та доступності. Додаткові метрики є модифікованим еквівалентом базових.

Для базових та додаткових метрик їхні “лінгвістичні” значення переводяться у відповідну цифрову шкалу (табл. 3.1). Так само надано таблиці числових значень, відповідних встановленим метрикам атак.

Таблиця 3.1. Бали за значення рейтингу показника

Рейтинг	Бали CVSS
None	0.0
Low	0.1 - 3.9
Medium	4.0 - 6.9
High	7.0 - 8.9
Critical	9.0 - 10.0

Також в CVSS 3.1 прийнято давати текстове представлення множини метрик у вигляді вектора-рядка (Табл.3.2).

Таблиця 3.2. Текстове кодування результатів оцінки за метриками

Група метрики	Назва метрики та кодове позначення	Можливі значення	Обов'язкове?
Base	Attack Vector (AV)	[N,A,L,P]	Yes
	Attack Complexity (AC)	[L,H]	Yes
	Privileges Required (PR)	[N,L,H]	Yes
	User Interaction (UI)	[N,R]	Yes
	Scope (S)	[U,C]	Yes
	Confidentiality (C)	[H,L,N]	Yes
	Integrity (I)	[H,L,N]	Yes
	Availability (A)	[H,L,N]	Yes
Temporal	Exploit Code Maturity (E)	[X,H,F,P,U]	No
	Remediation Level (RL)	[X,U,W,T,O]	No
	Report Confidence (RC)	[X,C,R,U]	No
Environmental	Confidentiality Requirement (CR)	[X,H,M,L]	No
	Integrity Requirement (IR)	[X,H,M,L]	No
	Availability Requirement (AR)	[X,H,M,L]	No
	Modified Attack Vector (MAV)	[X,N,A,L,P]	No
	Modified Attack Complexity (MAC)	[X,L,H]	No
	Modified Privileges	[X,N,L,H]	No

	Required (MPR)		
	Modified User Interaction (MUI)	[X,N,R]	No
	Modified Scope (MS)	[X,U,C]	No
	Modified Confidentiality (MC)	[X,N,L,H]	No
	Modified Integrity (MI)	[X,N,L,H]	No
	Modified Availability (MA)	[X,N,L,H]	No

Наприклад,

Якщо значення базових метрик виглядають так: “Attack Vector: Network, Attack Complexity:High, Privileges Required: Low, User Interaction: Required, Scope: Changed, Confidentiality: Low, Integrity: Low, Availability: None” (без зазначення додаткових метрик), то множина метрик представиться у вигляді рядка:

CVSS:3.1/AV:N/AC:H/PR:L/UI:R/S:C/C:L/I:L/A:N .

3.2. Завдання на роботу

1. Зібрати дані про типові для об'єктів інфраструктури відкритого ключа атаки, проаналізувати можливі шляхи здійснення (згідно варіанту). В звіті надати короткі відомості про атаки. Записати значення характеристик атак, заповнити табл. 2 для конкретних випадків.
2. Використовуючи калькулятор фреймворку, обчислити значення показника CVSS.
3. Зробити висновки про рівень небезпечності для заданого об'єкту.
4. Надати рекомендації по усуненню проблеми на об'єкті критичної інфраструктури.
5. Результати п.1-4 оформити у вигляді звіту, для перевірки розуміння матеріалу використати контрольні питання.

3.3. Варіанти завдань

№ варіанту=№ по списку mod 10 +1

1. SQL-ін'єкція з метою оволодівання персональними даними користувачів системи управління технологічними процесами.

2. DoS атака з метою відмови в обслуговуванні складових системи управління.
3. Несанкціонована модифікація інформації та даних (спрямована на примушення системи (SCADA, MES (monitoring engineering system), Historian-сервер) генерувати невідповідні рішення, які базуються на сфабрикованих словмисником даних).
4. Експлуатація вразливостей ПЗ (або прошивки кінцевих пристрій IoT).
5. Атаки на вразливі протоколи IoT з метою використання даних зв'язку IoT (контроль над сеансами зв'язку, сніффінг інформації тощо) – на прикладі Modbus;
6. Атаки на протоколи Industry 4.0 для керування роботами із використанням безпровідного зв'язку (2-3 приклади).
7. Атаки на CAN(Controller Area Network), засновані на переповненні буфера та невеликій кількості даних, які можна передати в одному пакеті.
8. MITM атаки на DNP3 (Distributed Network Protocol), який використовується для віддаленого конфігурування програмно-логічних контролерів.
9. Replay-атаки та sniffing LoRaWAN.
10. Атаки на протокол SigFox – з метою несанкціонованого одержання показань пристрій системи промислового моніторингу.

3.4. Контрольні питання

- 1) Які покращення внесені в CVSS 3.0 порівняно з CVSS 3.1?
- 2) Які види атак є найбільш небезпечними для об'єктів критичної інфраструктури.
- 3) Які підходи до визначення критичності наявних вразливостей пропонує CVSS?
- 4) Концепція Industry 4.0 (див. <https://www.spotlightmetal.com/iot-basics-what-does-industry-40-mean-a-842216/>)

4. Лабораторний практикум №2 (альтернативний)

Перехоплення bluetooth сигналів за допомогою Ubertooth One

Мета роботи: впевнитись у можливості перехоплення сигналів безпровідного зв'язку через прослуховування трафіку Bluetooth та використання їх для подальшого аналізу. Для цього ми будемо використовувати апаратний сніффер Bluetooth (Ubertooth One).

Необхідне програмне та апаратне забезпечення:

- Raspberry Pi або операційна система Linux (Ubuntu);
- Ubertooth One (Bluetooth USB dongle) .

4.1. Теоретичні відомості

Ubertooth One - це платформа бездротової розробки з відкритим вихідним кодом 2,4 ГГц. Пристрій спроектовано як вдосконалений приймач Bluetooth з функціями, що дозволяють використовувати його в якості платформи для прослуховування і моніторингу сигналів Bluetooth [6].

Ubertooth One побудований на базі мікроконтролера ARM Cortex-M3 і може захоплювати і демодулювати сигнали в діапазоні ISM 2,4 ГГц з вузькою смugoю пропускання всього 1 МГц. Також він включає: пакети базової швидкості Bluetooth, пакети BLE і 802.11 FHSS.

Ubertooth One відповідає стандартному форм-фактору USB-ключа.

Для використання Ubertooth One ми повинні встановити важливі залежності, а також оновити прошивку пристрою.

4.2. Завдання на роботу

Налаштування Ubertooth One

Необхідно встановити потрібні інструменти. В Terminal вводимо:

```
$ cd Desktop  
$ mkdir Ubertooth-one  
$ cd Ubertooth-one  
$ sudo apt-get install cmake libusb-1.0-0-dev make gcc g++ libbluetooth-dev wget pkg-config libpcap-dev python3-numpy python3-qtpy python3-distutils python3-setuptools  
$ pip install --upgrade setuptools  
$ pip3 install --upgrade setuptools  
$ sudo apt-get install python3-pyside  
$ sudo apt-get install python-pyside
```

```
$ sudo apt-get install python-qt4
```

```
$ apt-get install gcc-arm-none-eabi libnewlib-arm-none-eabi lsusb
```

Тепер потрібно встановити Bluetooth baseband library (libbtbb) для декодування пакетів Bluetooth.

```
$ wget https://github.com/greatscottgadgets/libbtbb/archive/2020-12-R1.tar.gz -O libbtbb-2020-12-R1.tar.gz
$ tar -xf libbtbb-2020-12-R1.tar.gz
$ cd libbtbb-2020-12-R1
$ mkdir build
$ cd build
$ cmake ..
$ make
$ sudo make install
$ sudo ldconfig
$ cd ..
$ cd ..
```

Після цього переконайтесь, що ви знаходитесь у каталозі Ubertooth-one, наступними кроками є завантаження інструментів Ubertooth, які потрібні для переходоплення сигналів Bluetooth, налаштовують Ubertooth та оновлюють його прошивку.

```
$ wget
https://github.com/greatscottgadgets/ubertooth/releases/download/2020-12-R1/ubertooth-2020-12-R1.tar.xz
$ tar -xf ubertooth-2020-12-R1.tar.xz
$ cd ubertooth-2020-12-R1/host
$ mkdir build
$ cd build
$ cmake ..
$ make
$ sudo make install
$ sudo ldconfig
$ cd ..
$ cd ..
$ cd ..
```

Встановимо Wireshark. В каталозі Ubertooth-one:

```
$ sudo apt-get install wireshark wireshark-dev libwireshark-dev cmake
$ cd libbtbb-2020-12-R1/wireshark/plugins/btbb
$ mkdir build
$ cd build
```

```
$ cmake -DCMAKE_INSTALL_LIBDIR=/usr/lib/x86_64-linux-gnu/wireshark/libwireshark3/plugins ..
```

```
$ make
```

```
$ sudo make install
```

```
$ cd ..
```

Зробимо те саме для Bluetooth. В каталозі Ubertooth-one:

```
$ cd libbtbb-2020-12-R1/wireshark/plugins/btbredr
```

```
$ mkdir build
```

```
$ cd build
```

```
$ cmake -DCMAKE_INSTALL_LIBDIR=/usr/lib/x86_64-linux-gnu/wireshark/libwireshark3/plugins ..
```

```
$ make
```

```
$ sudo make install
```

```
$ cd ..
```

Встановимо інструмент Crackle, який може розшифрувати шифрування BLE. В каталозі Ubertooth-one:

```
$ git clone https://github.com/mikeryan/crackle.git
```

```
$ cd crackle
```

```
$ make
```

```
$ cd ..
```

Перевіримо, яка версія прошивки у нас на Ubertooth-one.

```
$ lsusb
```

Ви повинні отримати щось подібне:

Bus 001 Device 014: ID Ubertooth One

```
$ ubertooth-util -v
```

Версія прошивки повина співпадати з версією, яку ми щойно встановили. Якщо відрізняється, потрібно зробити:

```
$ cd Ubertooth-one
```

```
$ cd ubertooth-2020-12-R1
```

```
$ cd ubertooth-one-firmware-bin
```

```
$ sudo Ubertooth-dfu -d bluetooth_rxtx.dfu -r
```

Після чого виконайте команду:

```
$ ubertooth-util-v
```

Ви повинні отримати прошивку 2020 року.

Перехоплення сигналів

Щоб ми могли бачити сигнали, які ми отримуємо від Ubertooth-one, виконаємо цю команду:

```
$ ubertooth-specan-ui
```

Тепер запустимо цю команду, щоб побачити список пристрійв Bluetooth, які у вас є, щоб ми могли розпочати збір пакетів з цього пристрою.

```
$ sudo hcitool lescan
```

Ви побачите такий список, і нам потрібна адреса Bluetooth, щоб почати перехоплення:

D5:AA:D0:41:A3:60 Mi Smart Band 4

78:A5:04:62:71:3D Wireless headphones

[...]

Нам потрібен один із них, щоб розпочати збір та збереження даних у файлі

Приклад

```
$ ubertooh-btle -f D5:AA:D0:41:A3:60 -c blue.pcap
```

Після цього відкриємо Wireshark і перевіримо дані

```
$ sudo wireshark
```

4.3. Вимоги до звіту

У протоколі мають бути скріншоти з результатами (рис. 4.1-4.4).

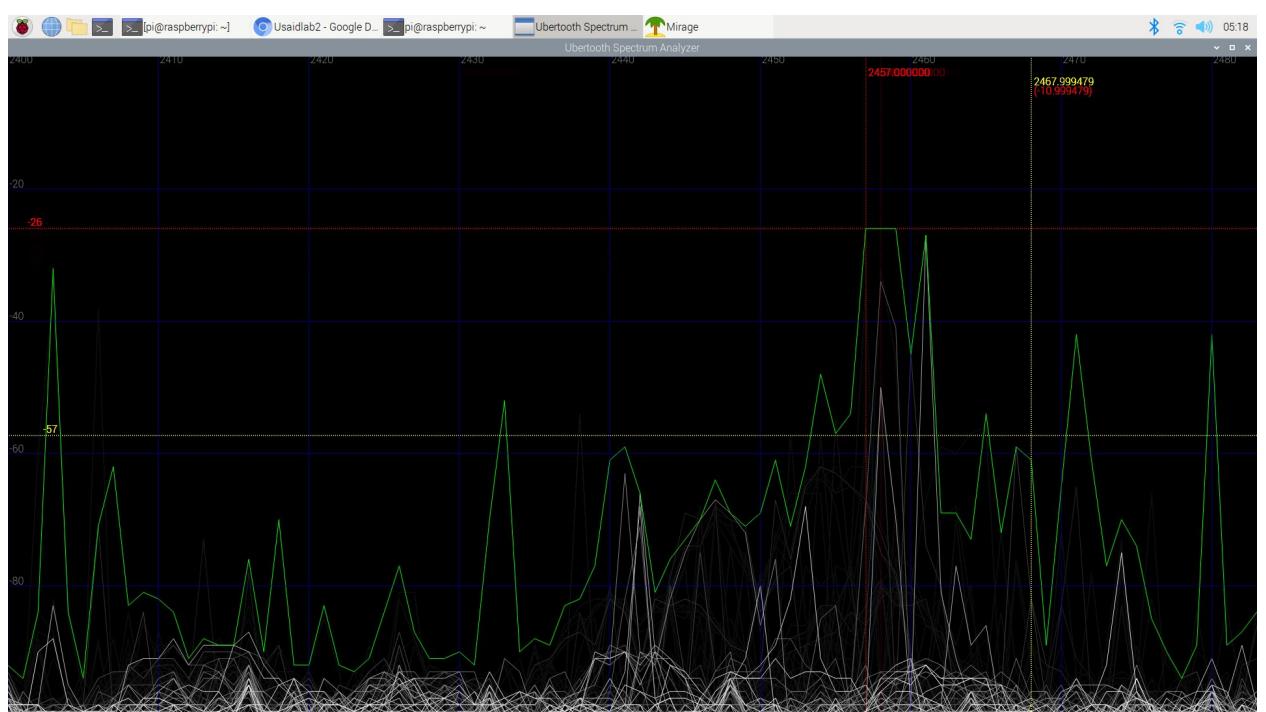
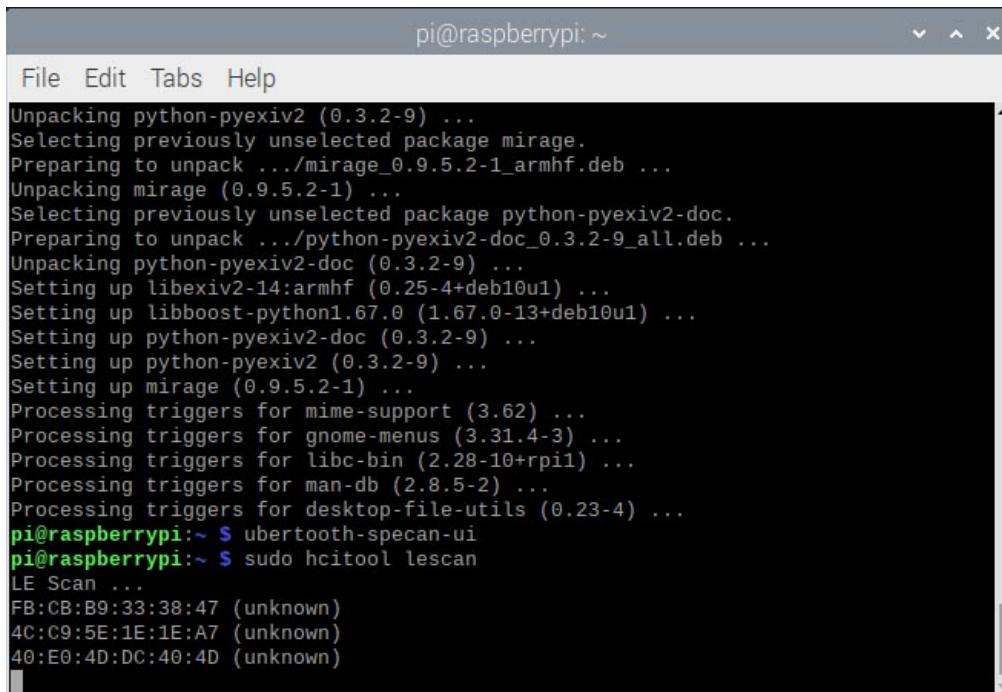


Рис. 4.1. Наявні в околі сніфера сигнали “виявлених” Bluetooth пристрійв



```
pi@raspberrypi: ~
File Edit Tabs Help
Unpacking python-pyexiv2 (0.3.2-9) ...
Selecting previously unselected package mirage.
Preparing to unpack .../mirage_0.9.5.2-1_armhf.deb ...
Unpacking mirage (0.9.5.2-1) ...
Selecting previously unselected package python-pyexiv2-doc.
Preparing to unpack .../python-pyexiv2-doc_0.3.2-9_all.deb ...
Unpacking python-pyexiv2-doc (0.3.2-9) ...
Setting up libexiv2-14:armhf (0.25-4+deb10u1) ...
Setting up libboost-python1.67.0 (1.67.0-13+deb10u1) ...
Setting up python-pyexiv2-doc (0.3.2-9) ...
Setting up python-pyexiv2 (0.3.2-9) ...
Setting up mirage (0.9.5.2-1) ...
Processing triggers for mime-support (3.62) ...
Processing triggers for gnome-menus (3.31.4-3) ...
Processing triggers for libc-bin (2.28-10+rpi1) ...
Processing triggers for man-db (2.8.5-2) ...
Processing triggers for desktop-file-utils (0.23-4) ...
pi@raspberrypi:~ $ ubertooh-specan-ui
pi@raspberrypi:~ $ sudo hcitool lescan
LE Scan ...
FB:CB:B9:33:38:47 (unknown)
4C:C9:5E:1E:1E:A7 (unknown)
40:E0:4D:DC:40:4D (unknown)
```

Рис.4.2. МАС-адреси виявлених Bluetooth пристрой

Рис. 4.3. Структура перехоплених даних

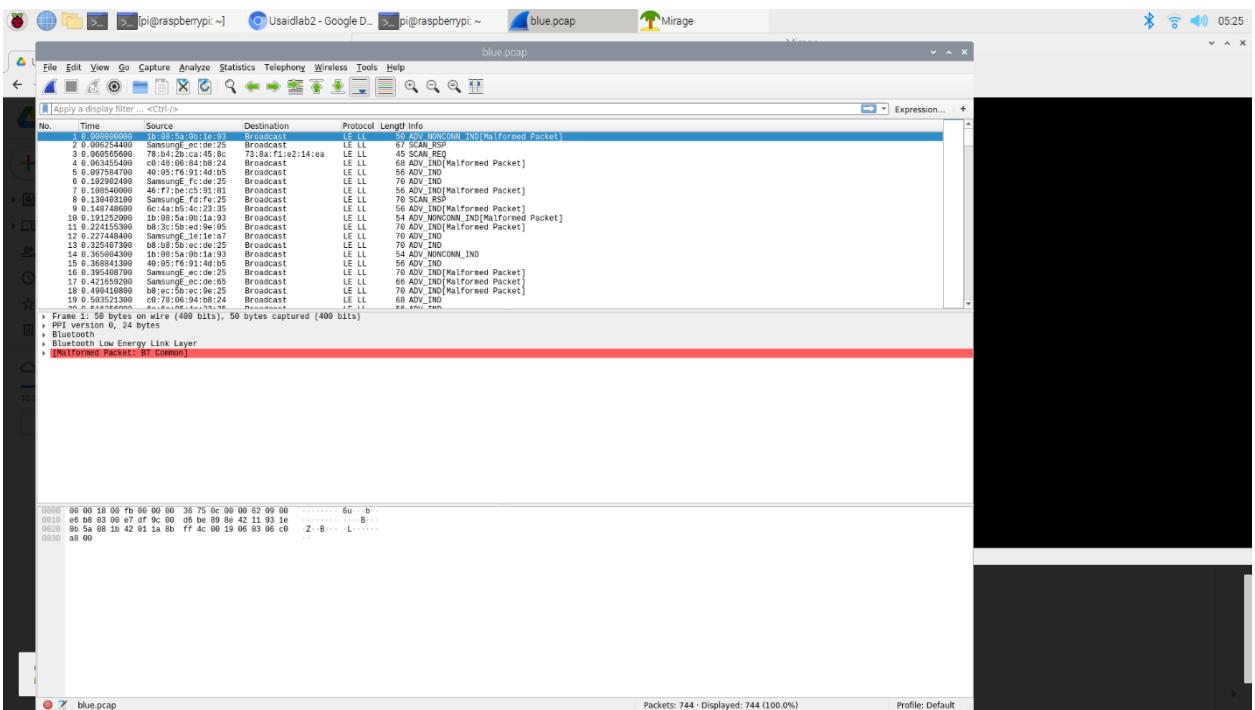


Рис. 4.4. Вигляд даних у wireshark. Бачимо, що дані шифровані.

Далі (теоретично) можливо використовувати утиліти для спроб зламу (типу crackle). Оскільки використані ключі та керування ними (частота поновлень зокрема) не завжди є достатньо надійними, спроби можуть бути вдалими і привести до порушень безпеки.

4.4. Варіанти завдань

Завдання виконується індивідуально, контролюються індивідуальні налаштування системи та передані дані.

4.5. Контрольні питання

1. Які функції виконує пристрій Ubertooth One?
2. Для чого в роботі використано Wireshark?
3. Для чого в роботі використано утиліту Crackle?
4. Які кроки налаштування Ubertooth One треба виконати, для його належної роботи?
5. Яку небезпеку несе можливість виявлення MAC-адрес оточуючих Bluetooth пристрій та даних, які вони передають?

5. Лабораторний практикум №3

Використання симулятора Сooja для моделювання об'єктів критичної інфраструктури

Мета роботи: ознайомитись із можливостями симулятора Сooja для Contiki OS. Навчитись використовувати його для моделювання поведінки базових пристройів промислового інтернету речей.

5.1. Теоретичні відомості

Contiki [7] - це операційна система для мережних систем з обмеженою пам'яттю. Акцент у використанні цієї ОС робиться на пристроях Інтернету речей, які використовують безпровідний зв'язок. На даний момент Contiki використовувалось для підтримки систем забезпечення роботи розумного міста. Зокрема, вуличного освітлення та звукового вуличного освітлення, радіаційного моніторингу, сигналізації. Перевагою цієї ОС є те, що вона є системою з відкритим кодом, що випускається під ліцензією BSD.

Риси:

- вбудований стек TCP / IP з підтримкою IPv4 та IPv6.
- потребує невелику кількість RAM (близько 10 КБ) та ROM (30 КБ). Повна версія системи з графічним інтерфейсом користувача, потребує біля 30 КБ RAM.
- Можливості планування під управлінням подіє-орієнтованого ядра (зручно для IoT).
- Система Contiki включає в себе симулятор Сooja, який імітує поведінку вузлів Contiki. Вузли належать до одного з трьох наступних класів: а) емульовані вузли Сooja; б) код Contiki, зкомпільований та виконаний на хості, який імітується симулятором; в) вузли Java, для яких поведінка вузла повинна бути описана класом Java.

За допомогою Сooja можна моделювати складені системи із вищевказаних типів сенсорних вузлів. Емульовані вузли можуть також використовуватися для включення інших вузлів, які не є вузлами Contiki, в модельовану мережу.

Приклади пристройів, які можуть бути змодельовані – мікроконтролери різних типів, сенсори, датчики тощо. Велика кількість їх програмно описана на мові C і знаходитьться у вихідних файлах симулятора.

5.2. Завдання на роботу

1. Із <https://sourceforge.net/projects/contiki/files/Instant%20Contiki/> - скачати та розпакувати в папку платформу для моделювання. Ця платформа є віртуальною платформою, яка використовується Cooja щоби запускати Contiki-NG як 'Cooja motes'. Наразі Contiki вбудовано в віртуальну машину Ubuntu. Cooja емулює запрограмовану поведінку вузлів (motes) та дозволяє підключити справжній пристрій.
2. Встановити та запустити віртуальну машину. Рекомендовано використовувати віртуальну машину VMWare з точки зору повноти та зручності налаштувань функціональності (хоча образ працює і під VirtualBox). Виконати File->Open->файл з папки п.1.
3. Запустити симулятор:
`cd contiki/tools/cooja
ant run`
4. Вибрати File-> New Simulation, при цьому використовуємо дані запропоновані по замочуванню. Натиснути Create.
5. Додати вузли:
[Add Mote-> Create New Mote Type-> Sky mote.](#)
6. Типи пристріїв які ми будемо використовувати: UDP-sink приймає інформацію, UDP-sender надсилає інформацію. Обрати відповідний файл із /home/user/contiki/examples/ipv6/rpl-collect/udp-sink.c. В home/users/contiki/examples/ є й інші приклади пристріїв.
7. Обравши udp-sink.c, натискаємо Compile. Повідомлення у вікні вида LD udp-sink.sky означає, що створився необхідний файл (це фактично прошивка для емульсованого пристрою). Натиснути Create. Задаємо кількість пристріїв цього типу (наприклад, один), Random positioning для задання фізичного розташування), та Add mote. Вузол графічно відобразиться.
8. Аналогічно додаємо декілька (наприклад, 5) пристріїв типу UDP sender із /home/user/contiki/examples/ipv6/rpl-collect/udp-sender.c. Вони графічно відобразяться у відповідному віконці.
9. Для покращення візуалізації у вікні емульсованої мережі Network вибрати у вкладці View опції Mote IDS, RadioTraffic, BackGroundGrid, MoteType, RadioEnvironment. Натискаючи на зображення пристрою у віконці можна побачити його радіус дії. Пристрій, які надсилають

інформацію, повинні знаходитись у радіусі дії приймача (рис. 5.1).

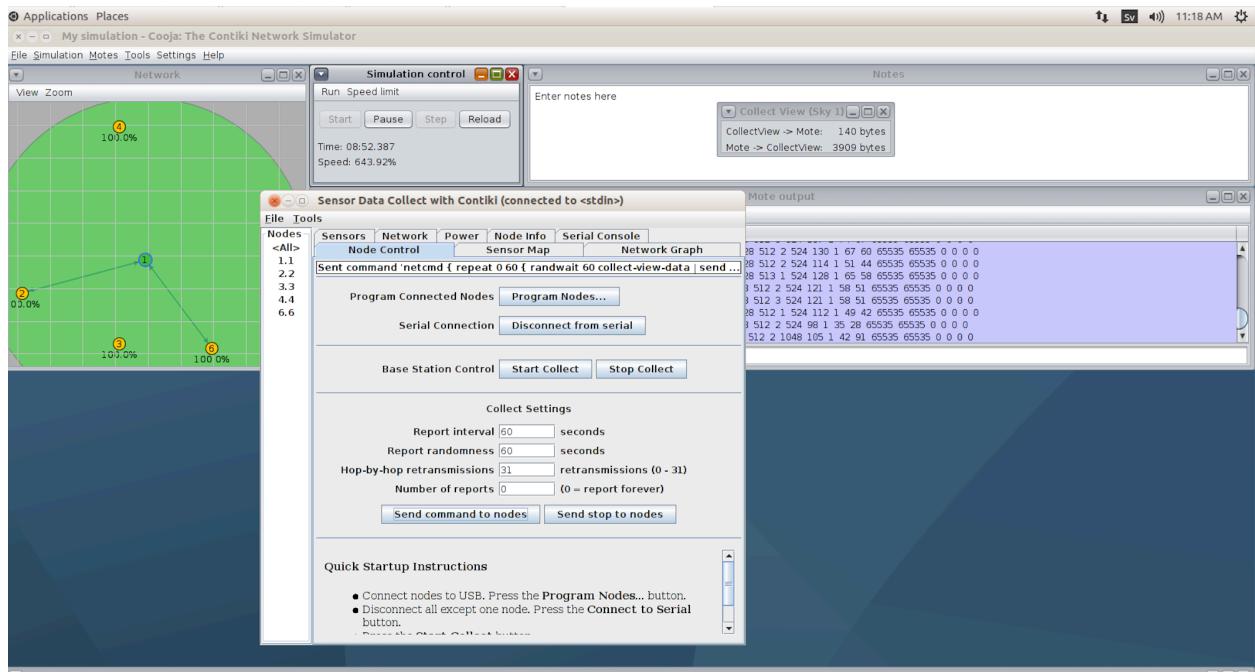


Рис. 5.1. Графічний інтерфейс Cooja

10. Тепер спробуємо збирати дані про поведінку мережі:

Tools->Collect View->Sky 1.

Simulaton Control->Start

Start Collect

Send Command to notes.

Зачекайте декілька хвилин, щоби накопичити дані.

11. Подивитись таблицю із даними про поведінку мережі у вкладці

Node Control->Node Info. Звернути увагу на споживання

електроенергії. Далі виконати такі дії у інтерфейсі:

Send Stop Nodes

Stop Collect.

Звернути увагу на дані у стовпці Power таблиці даних (рис.5.2).

Node Control	Sensor Map	Network Graph	Sensors	Network	Power	Node Info	Serial Console	lost	Hops	Rtmetric	ETX	Churn	Beacon Interval	Reboots	CPU Power	LPM Power	Listen Power	Transmit Power	Power	On-time	Listen Duty Cycle	Transmit Duty Cycle	Avg Inter-packet Time	Mir
0	0.000	0.000	0.000	0	0	0.000	0.000	0	0	0.000	0.000	0	0 min, 0 sec	0	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0 min, 25 sec	
0	1.000	512.000	16....	0	0	min, 08 sec	0	2.700	0.082	34.375	1.234	38.390	0 min, 52 sec	57.291	2.323	0 min, 52 sec	0	0	0	0	0	0	0 min, 52 sec	
0	1.000	512.000	16....	0	0	min, 08 sec	0	2.660	0.083	34.170	1.326	38.239	0 min, 43 sec	56.950	2.497	0 min, 52 sec	0	0	0	0	0	0	0 min, 52 sec	
0	1.000	512.000	16....	0	0	min, 08 sec	0	2.680	0.082	34.627	1.161	38.550	0 min, 40 sec	57.711	2.186	0 min, 36 sec	0	0	0	0	0	0	0 min, 36 sec	
0	1.000	512.000	16....	0	0	min, 08 sec	0	2.649	0.083	33.982	1.197	37.912	0 min, 34 sec	56.637	2.254	0 min, 22 sec	0	0	0	0	0	0	0 min, 22 sec	
0	1.000	1186.0...	16....	0	1	min, 32 sec	0	0.738	0.141	1.698	2.601	5.178	0 min, 36 sec	2.830	4.897	0 min, 34 sec	0	0	0	0	0	0	0 min, 34 sec	
000	1.000	646.800	16....	0	0.000	0 min, 24 sec	0	2.286	0.094	27.770	1.504	31.654	0 min, 41 sec	46.284	2.831	0 min, 33 sec	0	0	0	0	0	0	0 min, 33 sec	

Рис.5.2. Таблиця із даними про поведінку змодельованих пристрій мережі

5.3. Вимоги до звіту

У звіт вивести скріншоти із результатами п. 7-11.

5.4. Варіанти

Кількість UDP-sender та UDP-sink:

1. 5 та 1;
2. 6 та 1;
3. 7 та 1;
4. 8 та 1;
5. 9 та 1;
6. 10 та 1;
7. 11 та 1;
8. 12 та 1;
9. 13 та 1.
10. 14 та 1.

5.5. Контрольні питання

- 1) Призначення та основні переваги Contiki OS.
- 2) Які види пристроїв ви побачили серед прикладів у `/home/user/contiki/examples/`.
- 3) Яким чином здійснюється моделювання мережі у Сooja? Яким чином задається поведінка пристрою?
- 4) Чому важливо задавати фізичне розташування змодельованих пристроїв, а не лише їх внутрішні налаштування?
- 5) Які задачі безпеки можна вирішувати із використанням Сooja.

6. Лабораторний практикум №4

Використання симулятора Сооja для моделювання атак на об'єкти критичної інфраструктури

Мета роботи: ознайомитись із можливостями симулятора Сооja для для моделювання атаки на емульовану мережу безпровідних пристройів.

6.1. Теоретичні відомості

В мережах об'єктів критичної інфраструктури використовується протокол RPL (це протокол маршрутизації для малопотужних мереж, зокрема безпровідних мереж з низьким енергоспоживанням, як правило, чутливий до втрати пакетів). Впровадження RPL здійснюється із використанням ОС Contiki. RPL створює топологію, подібну дереву (DAG або спрямований ацикличний графік). Кожному вузлу в мережі присвоюється ранг (Rank), який збільшується в міру віддалення команд від кореневого вузла (DODAG). Вузли надсилають пакети, використовуючи найменший діапазон як критерій вибору маршруту.

Серед використовуваних у протоколі повідомлень є DIS-повідомлення ICMPv6 (інформаційні запити DODAG), які слугують для запиту інформації від сусіднього DODAG (аналогічно повідомленням запиту маршрутизатора, який використовується для виявлення існуючих мереж).

Атака DIS – це пряма атака на мережеві ресурси з використанням DODAG. Вона може бути віднесена до атаки переповнення, оскільки її метою є відмова в обслуговуванні вузлів внаслідок великої кількості DIS-запитів. Вузол під контролем зловмисника може передавати широкомовно або одноразово службові повідомлення DIS на інші вузли, що призводить до збільшення витрат на обробку трафіку та надмірне використання енергії. Тому DIS-атаку можна віднести до класу DoS-атак, оскільки метою є порушення роботи та недоступність мережі.

Розглянемо вихідні файли, які можна використовувати для імітації діяльності шкідливого вузла, що розсилає DIS-запити.

Файл rpl_private.h визначає приватні декларації для ContikiRPL, такі як значення за замовчуванням для керуючих повідомлень ICMP, пов'язані з ними таймери, режим роботи, таблиці маршрутизації DAG серед інших констант RPL. На рисунку 6.1 показаний фрагмент коду із відповідними змінами у файлі rpl_private.h. Зміна значень здійснюється для того, щоби імітувати діяльність зловмисного вузла, який постійно надсилає DIS-повідомлення. Існують дві директиви, що визначають інтервал DIS-

повідомлень та таймери затримки запуску. Обидва значення будуть встановлені до 0, у нормальному стані це ненульові параметри.

Файл rpl_timers.c відповідає за керування таймерами в ContikiRPL, відповідно зміни у ньому показано на рис. 6.2.

```
/* DIS related */
#define RPL_DIS_SEND          1
#ifndef RPL_DIS_INTERVAL_CONF
#define RPL_DIS_INTERVAL      RPL_DIS_INTERVAL_CONF
#else
#define RPL_DIS_INTERVAL      0 /*Value was 60*/
#endif
#define RPL_DIS_START_DELAY    0 /*Value was 5*/
/*-----*/
```

Рис.6.1. Зміни в файлі rpl_private.h

```
static void
handle_periodic_timer(void *ptr)
{
    rpl_purge_routes();
    rpl_recalculate_ranks();

    /* handle DIS */
#ifdef RPL_DIS_SEND
    //next_dis++; //added next_dis++;int i=0;while (i<20) {i++; dis_output(NULL);}
    next_dis++;
    int i=0;
    while (i<20) {i++; dis_output(NULL);}
    if(rpl_get_any_dag() == NULL && next_dis >= RPL_DIS_INTERVAL) {
        next_dis = 0;
        dis_output(NULL);
    }
#endif
    ctimer_reset(&periodic_timer);
}
```

Рис. 6.2. Зміни у файлі rpl_timers.c

6.2. Завдання на роботу

1. В симуляторі Сооja вибрати File-> New Simulation. Натиснути Create.
2. Додати вузли:
Add Mote-> Create New Mote Type-> Sky mote.
В home/users/contiki/examples/ є й інші приклади пристрійв.
3. Обрати та скомпілювати файл із
`/home/user/contiki/examples/ipv6/rpl-collect/udp-sink.c`.
4. Аналогічно додати декілька пристрійв (згідно варіанту, див. попередню роботу) типу UDP sender із
`/home/user/contiki/examples/ipv6/rpl-collect/udp-sender.c`. Вони графічно відобразяться у вікні Network.
5. Простежити, щоби пристрой, які надсилають інформацію, знаходились у радіусі дії приймача.

6. Тепер спробуємо зібрати дані про поведінку мережі:
 Tools->Collect View->Sky 1
 Simulaton Control->Start
 Start Collect
 Send Command to notes.
 Зачекайте декілька хвилин, щоби накопичити дані.
7. Подивитись таблицю із даними про поведінку мережі у вкладці Node Control->Node Info. Зберегти дану таблицю, вона потім знадобиться для порівняння.
8. Виконати дії:
 Send Stop Nodes
 Stop Collect.
9. У папці /Contiki/Core/net/rpl (зробити копію до змін).
10. На доданок у п. 1-8 створити шкідливий вузол: Add mote - > Create mote type, Udp-sender2.c Compile, create.
11. Змінити файли:
 - 11.1. rpl_private.h (рис. 6.1)
 - a. rpl_timers.c (рис. 6.2)
12. Пересвідчитись, що діяльність новоутвореного вузла затримує іншу частину мережі. Порівняти швидкість емуляції із швидкістю у попередньому випадку (без зловмисного вузла).
13. Collect View->Sky1, Node Control-> Start Collect, Send Command to nodes. Зачекати рельних 5 хв. (не симуляційних), а потім подивитись в Node Info.
14. Звернути увагу на стовпець Power, порівняти його значення із значеннями з попередньої симуляції (без DIS-атаки). Звернути увагу на Listen Power; CPU Power (цей вид відповідає за перегрів пристрою та вихід з ладу).

6.3. Вимоги до звіту

У звіт вивести скріншоти із результатами п. 9-14 та графік споживання енергії (а саме, Power, Listen Power; CPU Power) до та після атаки для кожного вузла.

6.4. Варіанти

Кількість UDP-sender та UDP-sink до DIS-атаки:

1. 5 та 1;
2. 6 та 1;

3. 7 та 1;
4. 8 та 1;
5. 9 та 1;
6. 10 та 1;
7. 11 та 1;
8. 12 та 1;
9. 13 та 1.
10. 14 та 1.

6.5. Контрольні питання

- 1) Призначення файлів rpl_private.h та rpl_timers.c.
- 2) Сутність DIS- атаки для безпровідної мережі.
- 3) Які зміни треба впровадити до файлів rpl_private.h та rpl_timers.c для моделювання DIS- атаки?
- 4) Яким чином можна розпізнати здійснення DIS атаки в мережі об'єкта критичної інфраструктури?

7. Лабораторний практикум №5

Вдосконалення недоліків протоколів інтернету речей

Мета роботи: Ознайомитись із недоліками протоколів пристройів інтернету речей, що є частиною об'єктів критичної інфраструктури та типовими програмними рішеннями з їх усунення.

7.1. Теоретичні відомості

Типові вразливості протоколів пристройів інтернету речей показані у таблиці 1.

Таблиця 7.1. Протоколи, які використовуються у мережах об'єктів критичної інфраструктури

Протоколи	Рівень	Задачі	Вразливості	Топологія
1	2	3	4	5
Modbus (проводний)	Фізичний Канальний Прикладний	Організація зв'язку між цифровими пристроями в області релейного захисту і автоматики. Може використовуватись для передачі даних через послідовні лінії зв'язку RS-485, RS-422, RS-232, а також мережі TCP/IP (Modbus TCP). Розроблений для використання в програмно-логічних контролерах.	У базовому варіанті відсутні аутентифікація і шифрування. Комунікація проходить в незахищено му режимі.	Шина
CAN(Controll er Area Network), (проводний)	Фізичний Канальний Прикладний(CANope n, DeviceNet)	Об'єднання в єдину мережу різних виконуючих пристройів і датчиків. Зв'язок даних в пересувних системах, машинах, технічному обладнанні й промисловій автоматизації	Невелика кількість даних, які можна передати в одному пакеті(до 8 байт). Великий	Шина

Протоколи	Рівень	Задачі	Вразливості	Топо-логія
			розмір службових даних в пакеті. Незахищений заголовок. Переповнення буфера.	
DNP3(Distributed Network Protocol), (провідний)	Фізичний Канальний Прикладний	Підтримка роботи з подіями типу: зміна стану і подія з міткою часу. Обмін даними по мережах Ethernet і через інтерфейси RS232/RS485. Використання при передачі повідомлення великого розміру. Широкомовна розсилка повідомлень. Віддалене конфігурування програмно-логічних контролерів. Наявне розширення для безпечної аутентифікації.	Атака “людина посередині”. Переповнення довжини поля.	Точка-точка, шина, зірка, дерево
6LoWPAN (безпровідний)	Канальний Мережний	Забезпечити взаємодію безпровідних персональних мереж IEEE 802.15 з мережами IP. Стиснення заголовка IP пакета. Зменшення споживання електроенергії.	Відмова в обслуговуванні. Атака на фрагментацію пакетів.	Mesh-мережа
LoRaWAN (безпровідний)	Канальний	Сумісність з існуючими мережами/технологіями безпровідної передачі даних. Обслуговування десятків-	Атаки підслуховування, bit flipping, ACK	Зірка

Протоколи	Рівень	Задачі	Вразливості	Топологія
		сотень тисяч пристройів. Забезпечення великої зони дії і малого енергоспоживання кінцевих пристройів. Зчитування показників лічильників газу, води, електроенергії.	spoofing, replay attack.	
SigFox (безпровідний)	Фізичний	Передача невеликого об'єму даних. Забезпечення великого територіального охоплення безпровідною мережею. Застосування: безпровідні сенсорні мережі, автоматизація збору показань пристройів обліку, системи промислового моніторингу і керування.	Відсутнє шифрування	Зірка

Типові атаки на пристрой інтернету речей та методи запобігання ним показані у табл.7.2.

Таблиця 7.2. Атаки на пристрой промислового ІоТ

Вид пристроя	Приклад атаки	Причини	Методи запобігання
Інтернет-інфраструктура компаній	DDoS	Легкий доступ до мережі через невеликі пристрой (н-д, телефон)	Контроль підключення всіх пристрой, обмеження доступу до мережі
Індустріальні логічні контролери	Stuxnet (APT)	Базування на стандартній операційній системі та зєднання з WAN	Обмеження доступу\звязку контролерів з зовнішнім інтернетом

Вид пристрою	Приклад атаки	Причини	Методи запобігання
			та використання не стандартних ОС
Роутери та IP-камери	DNS flood and DDoS	Застарілі процесори на базі Linux та незмінні дефолтні логіни\паролі	Постійне оновлення системного забезпечення та зміна дефолтних логінів\паролів на складні з використанням літер, символів та цифр
Опалювальна система	DDoS	Безконтрольність системи, відсутність моніторингу атак та впливів на систему	Створення умов для постійного моніторингу системи\мережі та попередження атак
Мобільні пристрої	DNS flood	Відсутність реакції на зміну віддачі\завантаження в мережі	Контроль та швидке реагування на послабшення звязку пристрій

Modbus [8] — протокол з відкритим вихідним кодом. Це означає, що він може бути включений в широкий діапазон типів пристройв від будь-якого постачальника обладнання.

Протокол використовує просту структуру повідомлень, що робить її менш складною для розгортання. Може вимагати всього кілька днів для реалізації. Це дає явну конкурентну перевагу в порівнянні з іншими протоколами, які можуть вимагати багато місяців для вивчення і розгортання. Протокол підтримує послідовні або Ethernet-з'єднання. Використовується з двома типами послідовних з'єднань: RS-232 і RS-485.

Деякі версії протоколу Modbus TCP також можуть бути відправлені через Ethernet або TCP/IP. Ці повідомлення Modbus упаковані як однобітові або 16-бітні пакети повідомлень. Мережний протокол Modbus не є частиною фізичного рівня мережі. Зв'язок передається над фізичним рівнем, що дозволяє використовувати його у багатьох різних типах мереж. Ця

властивість нефізичного рівня робить Modbus протоколом прикладного характеру.

Modbus відноситься до протоколів прикладного рівня мережової моделі OSI. Контролери на шині Modbus взаємодіють, використовуючи master-slave модель, засновану на транзакціях, що складаються із запиту і відповіді.

Зазвичай в мережі є тільки один клієнт, так званий «головний» (англ. master) пристрій, і кілька серверів - «підлеглих» (англ. slaves) пристройів. Головний пристрій ініціює транзакції (передає запити). Підлеглі пристрої передають запитувані головним пристроєм дані, або виконують запитувані дії. "Головний" може звертатися індивідуально до підлеглого або ініціювати передачу широкомовного повідомлення для всіх підлеглих пристройів. Підлеглий пристрій формує повідомлення і повертає його у відповідь на запит, адресований саме йому. При отриманні широкомовного запиту відповідь не формується.

Специфікація Modbus описує структуру запитів і відповідей. Їх основа - елементарний пакет протоколу, так званий PDU (Protocol Data Unit). Структура PDU не залежить від типу лінії зв'язку і включає в себе код функції і поле даних. Код функції кодується однобайтовим полем і може приймати значення в діапазоні 1 ... 127. Діапазон значень 128 ... 255 зарезервований для кодів помилок. Поле даних може бути змінної довжини. Розмір пакета PDU обмежений 253 байтами.

Для передачі пакету по фізичних лініях зв'язку PDU поміщається в інший пакет, що містить додаткові поля. Цей пакет має назву ADU (Application Data Unit). Формат ADU залежить від типу лінії зв'язку.

2 байти	2 байти	2 байти	1 байт	1 байт	до 252 байт
ID транзакції	ID протоколу	довжина пакету	адреса підлеглого пристрою	код функції	дані (інформаційне поле)

Рис.1. Структура пакета Modbus TCP

Пояснення:

- ID транзакції - два байти, зазвичай нулі;
- ID протоколу - два байти, нулі;
- Довжина пакету - два байти, старший, потім молодший, довжина наступної за цим полем частини пакета;
- Адреса підлеглого пристрою - адреса підлеглого пристрою, до якого адресовано запит. Зазвичай ігнорується, якщо з'єднання встановлено з певним пристроєм. Може використовуватися, якщо з'єднання встановлено з мостом, який виводить, наприклад, в мережу RS485.

- Код функції - це наступне однобайтне поле кадру. Воно говорить підлеглому пристрою, які дані або виконання якої дії вимагає від нього ведучий пристрій;

- Дані - поле містить інформацію, необхідну підлеглому пристрою для виконання заданої головним пристроєм функції або містить дані, що передаються підлеглим пристроям у відповідь на запит головного. Довжина і формат поля залежить від номера функції;

Поле контрольної суми в Modbus TCP відсутнє.

7.2. Завдання на роботу

1. Проаналізувати версії протокола Modbus, наприклад за <https://www.modbus.org/specs.php> .
2. Запропонувати засоби, які можуть гарантувати безпеку протоколу. Приклад реалізації див., наприклад, тут: <https://github.com/stephane/libmodbus>
3. Реалізувати програмні процедури, які покращують захищеність протоколу із використанням криптографічних засобів бібліотеки openSSL (наприклад, звідси <https://github.com/openssl/openssl>). Вибір криптографічних функцій обґрунтувати.
4. Показати дієвість запропонованих засобів на прикладі взаємодії “клієнт-сервер” із використанням сніфера, який перехоплює пакети та аналізує їх вміст. В якості сніфера можна використати wireshark.

7.3. Вимоги до звіту

У скріншотах показати результати сніффінгу до впровадження у протокол відповідних засобів (шифрування, автентифікація, електронно-цифровий підпис тощо) та після, виділити різницю.

7.4. Варіанти

- Парні варіанти: modbus v3.1.6
- Непарні варіанти: modbus v3.0.8.

7.5. Контрольні питання

1. Які вразливості має протокол Modbus.
2. Які засоби усунення вразливостей варто використовувати у більшості протоколів інтернету речей, які використовуються на об'єктах критичної інфраструктури.

8. Лабораторний практикум №5 (альтернативний)

Впровадження MQTT з протоколом безпеки транспортного рівня використовуючи бібліотеку OpenSSL

Мета роботи: полягає в тому, щоб навчитися встановлювати безпечне/зашифроване з'єднання MQTT між клієнтами MQTT і Mosquitto Broker, що працюють на комп'ютері з використанням OpenSSL, який є одним з найбільш широко використовуваних криптографічних інструментів з відкритим кодом у системах на базі UNIX.

Необхідні інструменти/програмне забезпечення

- Linux operating system (Ubuntu)
- OpenSSL Library
- Mosquitto Clients/Broker

8.1. Теоретичні відомості

Налаштування

Message Queuing Telemetry Transport (MQTT) вважається одним із найпоширеніших протоколів зв'язку, коли йдеться про машинно-машинний зв'язок (M2M). MQTT — це легкий протокол, який складається з мережі публікації-підписки, яка транспортує повідомлення між пристроями за допомогою брокера (контролера). Тому, використовуючи OpenSSL, ми можемо генерувати ключі та сертифікати для нашого брокера та клієнтів MQTT. Тому ми повинні мати центр сертифікації (СА), який є довіреним об'єктом та використовує цифрові сертифікати “*data files*”, які використовуються для зв'язування об'єкта за допомогою відкритого ключа.

8.2. Завдання

Завантажимо необхідні бібліотеки для встановлення Mosquitto Broker / Clients.

В додатку необхідна папка бібліотеки, яку необхідно встановити в Ubuntu для виконання цієї лабораторної роботи.

Завантажте папку на машину Ubuntu та дотримуйтесь інструкцій.

Створіть папку з назвою Lab-5-MQTT-TLS:

```
$ cd Desktop  
$ cd Lab-5-MQTT-TLS
```

Помістіть туди файл MQTT-TLS-Libraries.zip, з додатку, та розархівуйте його:

```
$ cd MQTT-TLS-Libraries
```

Має бути п'ять файлів *.deb*, які потрібно встановити:

```
$ sudo apt install libev4
```

```
$ sudo dpkg -i libdl2_2.18.5-0.2_amd64.deb
```

```
$ sudo dpkg -i libwebsockets16_4.0.20-1_amd64.deb
```

```
$ sudo dpkg -i libmosquitto1_1.6.12-1_amd64.deb
```

```
$ sudo dpkg -i mosquitto-clients_1.6.12-1_amd64.deb
```

Після їх успішного встановлення потрібно встановити Mosquitto:

```
$ sudo dpkg -i mosquitto_1.6.12-1_amd64.deb
```

```
$ cd ..
```

Щоб переконатися, що Mosquitto встановлено, запустіть його:

```
$ mosquitto
```

Ви повинні побачити, що версія Mosquitto 1.6.12.

Примітка:

Якщо ви отримуєте повідомлення про помилку, що адреса вже використовується - це тому, що як тільки Mosquitto буде встановлено, він почне прослуховувати порт 1883.

Далі потрібно створити наші ключі та сертифікати.

```
$ mkdir certs
```

```
$ cd certs
```

```
$ mkdir ca
```

```
$ cd ca/
```

```
$ openssl req -new -x509 -days 365 -extensions v3_ca -keyout ca.key -out ca.crt
```

Вихідні дані

Generating a RSA private key

```
.....+++++.....+++++
```

writing new private key to 'ca.key'

Enter PEM pass phrase: (*ви повинні ввести пароль, наприклад:*

ipt123456)

Verifying - Enter PEM pass phrase: (*введіть її знову*)

```
-----
```

You are about to be asked to enter information that will be incorporated into your

certificate request.

What you are about to enter is what is called a Distinguished Name or a DN.

There are quite a few fields, but you can leave some blank

For some fields there will be a default value,

If you enter '.', the field will be left blank.

Country Name (2 letter code) [XX]:UA
State or Province Name (full name) []: Ukraine
Locality Name (eg, city) [Default City]: Kyiv
Organization Name (eg, company) [Default Company Ltd]: IPT
Organizational Unit Name (eg, section) []:
Common Name (eg, your name or your server's hostname) []: ваше ім'я
Email Address []: ваш email

Переконаємося, що у нас є необхідні файли:

\$ ls

ca.crt ca.key

Залишимо цей каталог:

\$ cd ..

Тепер у нас є центр сертифікації. Створимо ключі та сертифікати брокера.

Створіть приватний ключ для брокера:

\$ mkdir broker

\$ cd broker

\$ openssl genrsa -out broker.key 2048

Вихідні дані

Generating RSA private key, 2048 bit long modulus (2 primes)

.....+++++.....+++++
.....+++++.....+++++

e is 65537 (0x010001)

\$ ls

broker.key

Створимо файл запиту на підпис із цього ключа:

\$ openssl req -out broker.csr -key broker.key -new

Вихідні дані

You are about to be asked to enter information that will be incorporated into your

certificate request.

What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.

Country Name (2 letter code) [XX]:UA

State or Province Name (full name) []: **Ukraine**
Locality Name (eg, city) [Default City]: **Kyiv**
Organization Name (eg, company) [Default Company Ltd]: **IPT**
Organizational Unit Name (eg, section) []:
Common Name (eg, your name or your server's hostname) []: **ваше ім'я**
Email Address []: **ваши email**

Please enter the following 'extra' attributes to be sent with your certificate request

A challenge password []: (**Введіть твой самий пароль, що й раніше, ipt123456**)

An optional company name []: **IPT**

В результаті мають бути створені два файли:

\$ **ls**

broker.csr broker.key

Передамо файл запиту на сертифікат до центру сертифікації

\$ **openssl x509 -req -in broker.csr -CA ./ca/ca.crt -CAkey ./ca/ca.key**

CAcreateserial -out broker.crt -days 100

Вихідні дані

Signature ok

subject=C = UA, ST = Ukraine, L = Kyiv, O = IPT, CN = **ваше ім'я**,
emailAddress =**ваши email**

Getting CA Private Key

Enter pass phrase for ./ca/ca.key: (**введіть свій пароль: ipt123456**)

Повинні з'явитися три файли:

\$ **ls**

broker.crt broker.csr broker.key

Далі потрібно видалити файл *broker.csr*

\$ **rm broker.csr**

Залиште цю директорію:

\$ **cd ..**

Потрібно зробити те ж саме для клієнтів MQTT:

\$ **mkdir client**

\$ **cd client**

\$ **openssl genrsa -out client.key 2048**

Вихідні дані

Generating RSA private key, 2048 bit long modulus (2 primes)

```
.....+++++
.....+++++
e is 65537 (0x010001)
```

```
$ openssl req -out client.csr -key client.key -new
```

Вихідні дані

You are about to be asked to enter information that will be incorporated into your certificate request.

What you are about to enter is what is called a Distinguished Name or a DN. There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '!', the field will be left blank.

```
-----
Country Name (2 letter code) [XX]: UA
State or Province Name (full name) []: Ukraine
Locality Name (eg, city) [Default City]: Kyiv
Organization Name (eg, company) [Default Company Ltd]: IPT
Organizational Unit Name (eg, section) []:
Common Name (eg, your name or your server's hostname) []: ваше ім'я
Email Address []: ваши email
```

Please enter the following 'extra' attributes
to be sent with your certificate request

A challenge password []: (*введіть пароль: ipt123456*)
An optional company name []: IPT

```
$ openssl x509 -req -in client.csr -CA ..../ca/ca.crt -CAkey ..../ca/ca.key -
```

CAcreateserial -out client.crt -days 100

Вихідні дані

Signature ok

subject=C = FR, ST = France, L = Strasbourg, O = opeNest, CN = localhost,
emailAddress = contact@openest.io

Getting CA Private Key

Enter pass phrase for/ca/ca.key: **ipt123456**

```
$ ls
```

client.crt client.csr client.key

Видалимо файл *client.csr*

```
$ rm client.csr
```

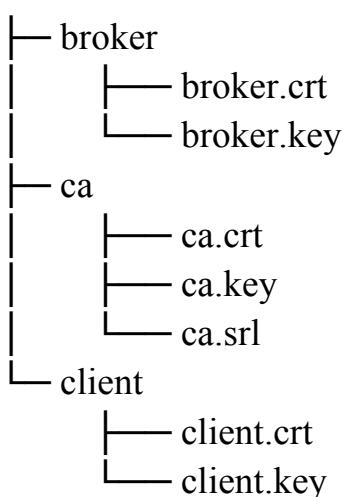
Залишимо каталог

\$ cd ..

Ви маєте отримати такі файли та каталоги

\$ sudo apt install tree

\$ tree .



Примітка:

Якщо у вас немає зазначених файлів, поверніться назад і переконайтесь, що ви виконали всі кроки.

Налаштуємо Mosquitto:

\$ sudo gedit /etc/mosquitto/mosquittotest.conf

Скопіюйте та вставте це в папку, а потім збережіть за допомогою Ctrl+S або Save.

Ви повинні змінити шлях для *ca.crt* | *broker.crt* | *broker.key*

port 8883

cafile /home/iryna/Desktop/MQTT-TLS/certs/ca/ca.crt

#capath /home/iryna/Desktop/MQTT-TLS/certs/ca

Path to the PEM encoded server certificate.

certfile /home/iryna/Desktop/MQTT-TLS/certs/broker/broker.crt

Path to the PEM encoded keyfile.

keyfile /home/iryna/Desktop/MQTT-TLS/certs/broker/broker.key

require_certificate true

Запустимо Mosquitto broker

\$ sudo mosquitto -v -c /etc/mosquitto/mosquittotest.conf

Перейдіть до каталогу клієнтів. Відкрийте новий термінал і запустіть Mosquitto

\$ cd client

```
$ mosquitto_pub -p 8883 --cafile ./ca/ca.crt --cert client.crt --key client.key  
-h
```

```
localhost -m hello -t /world
```

Якщо все було виконано вірно, ви маєте отримати:

```
1621455059: mosquitto version 1.6.12 starting  
1621455059: Config loaded from /etc/mosquitto/mosquittotest.conf.  
1621455059: Opening ipv4 listen socket on port 8883.  
1621455059: Opening ipv6 listen socket on port 8883.  
1621455059: mosquitto version 1.6.12 running  
1621455083: New connection from 127.0.0.1 on port 8883.  
1621455083: New client connected from 127.0.0.1 as mosq-  
E8T7kpb464N1lu9T46  
(p2, c1, k60).  
1621455083: No will message specified.  
1621455083: Sending CONNACK to mosq-E8T7kpb464N1lu9T46 (0, 0)  
1621455083: Received PUBLISH from mosq-E8T7kpb464N1lu9T46 (d0,  
q0, r0, m0,  
'/world', ... (5 bytes))  
1621455083: Received DISCONNECT from mosq-E8T7kpb464N1lu9T46  
1621455083: Client mosq-E8T7kpb464N1lu9T46 disconnected.
```

Виходячи із наданих вказівок, виконайте дії, щоби одержати на власному досвіді скріншоти наведені нижче.

8.3. Вимоги до звіту

Для звіту надайте відповідні скріншоти – Рис.8.1-8.5.

```
iryna@ubuntu: ~/Desktop/Lab-6-MQTT-TLS/certs$ tree .
.
├── broker
│   ├── broker.crt
│   └── broker.key
├── ca
│   ├── ca.crt
│   ├── ca.key
│   └── ca.srl
└── client
    ├── client.crt
    ├── client.key
    ├── publisher.crt
    ├── publisher.key
    ├── subscriber.crt
    └── subscriber.key

3 directories, 11 files
```

```
iryna@ubuntu:~/Desktop/Lab-6-MQTT-TLS/certs$ 
```

Рис. 8.1. Дерево сертифікатів

```
iryna@ubuntu:~/Desktop/Lab-6-MQTT-TLS/certs/client$ mosquitto_pub -p 8883 --cafile ../ca/
.crt --cert publisher.crt --key publisher.key -h localhost -t Temperature -m Temp:42
iryna@ubuntu:~/Desktop/Lab-6-MQTT-TLS/certs/client$ mosquitto_pub -p 8883 --cafile ../ca/
.crt --cert publisher.crt --key publisher.key -h localhost -t Temperature -m Temp:40
```

Рис.8.2. Publisher дає поточну температуру

```
tryna@ubuntu:~/Desktop$ sudo mosquitto -v -c /etc/mosquitto/mosquittotest.conf
[sudo] password for tryna:
1627572978: mosquitto version 1.6.12 starting
1627572978: Config loaded from /etc/mosquitto/mosquittotest.conf.
1627572978: Opening ipv4 listen socket on port 8883.
1627572978: Opening ipv6 listen socket on port 8883.
1627572978: mosquitto version 1.6.12 running
1627573002: New connection from 127.0.0.1 on port 8883.
1627573002: New client connected from 127.0.0.1 as mosq-fbsbmNvuHuVmCJIKRb (p2, c1, k60).
1627573002: No will message specified.
1627573002: Sending CONNACK to mosq-fbsbmNvuHuVmCJIKRb (0, 0)
1627573002: Received PUBLISH from mosq-fbsbmNvuHuVmCJIKRb (d0, q0, r0, m0, 'Temperature', . (7 bytes))
1627573002: Received DISCONNECT from mosq-fbsbmNvuHuVmCJIKRb
1627573002: Client mosq-fbsbmNvuHuVmCJIKRb disconnected.
1627573013: New connection from 127.0.0.1 on port 8883.
1627573013: New client connected from 127.0.0.1 as mosq-BPDvWw4BQh0tYma8UZ (p2, c1, k60).
1627573013: No will message specified.
1627573013: Sending CONNACK to mosq-BPDvWw4BQh0tYma8UZ (0, 0)
1627573013: Received SUBSCRIBE from mosq-BPDvWw4BQh0tYma8UZ
1627573013:     Temperature (QoS 0)
1627573013: mosq-BPDvWw4BQh0tYma8UZ 0 Temperature
1627573013: Sending SUBACK to mosq-BPDvWw4BQh0tYma8UZ
1627573073: Received PINGREQ from mosq-BPDvWw4BQh0tYma8UZ
1627573073: Sending PINGRESP to mosq-BPDvWw4BQh0tYma8UZ
1627573133: Received PINGREQ from mosq-BPDvWw4BQh0tYma8UZ
1627573133: Sending PINGRESP to mosq-BPDvWw4BQh0tYma8UZ
1627573137: New connection from 127.0.0.1 on port 8883.
1627573137: New client connected from 127.0.0.1 as mosq-os666wDfmVkBXR (p2, c1, k60).
1627573137: No will message specified.
1627573137: Sending CONNACK to mosq-os666wDfmVkBXR (0, 0)
1627573137: Received PUBLISH from mosq-os666wDfmVkBXR (d0, q0, r0, m0, 'Temperature', . (7 bytes))
1627573137: Sending PUBLISH to mosq-BPDvWw4BQh0tYma8UZ (d0, q0, r0, m0, 'Temperature', . (7 bytes))
1627573137: Received DISCONNECT from mosq-os666wDfmVkBXR
1627573137: Client mosq-os666wDfmVkBXR disconnected.
1627573193: Received PINGREQ from mosq-BPDvWw4BQh0tYma8UZ
1627573193: Sending PINGRESP to mosq-BPDvWw4BQh0tYma8UZ
1627573253: Received PINGREQ from mosq-BPDvWw4BQh0tYma8UZ
1627573253: Sending PINGRESP to mosq-BPDvWw4BQh0tYma8UZ
```

Рис. 8.3. Брокер одержує дані від publisher та надсилає їх до subscriber

```
tryna@ubuntu:~/Desktop/Lab-6-MQTT-TLS/certs/client$ mosquitto_sub -p 8883 --cafile ./ca.crt --cert subscriber.crt --key subscriber.key -h localhost -t Temperature
Temp:42
Temp:40
```

Рис. 8.4. Subscriber одержує опубліковану тему Temperature

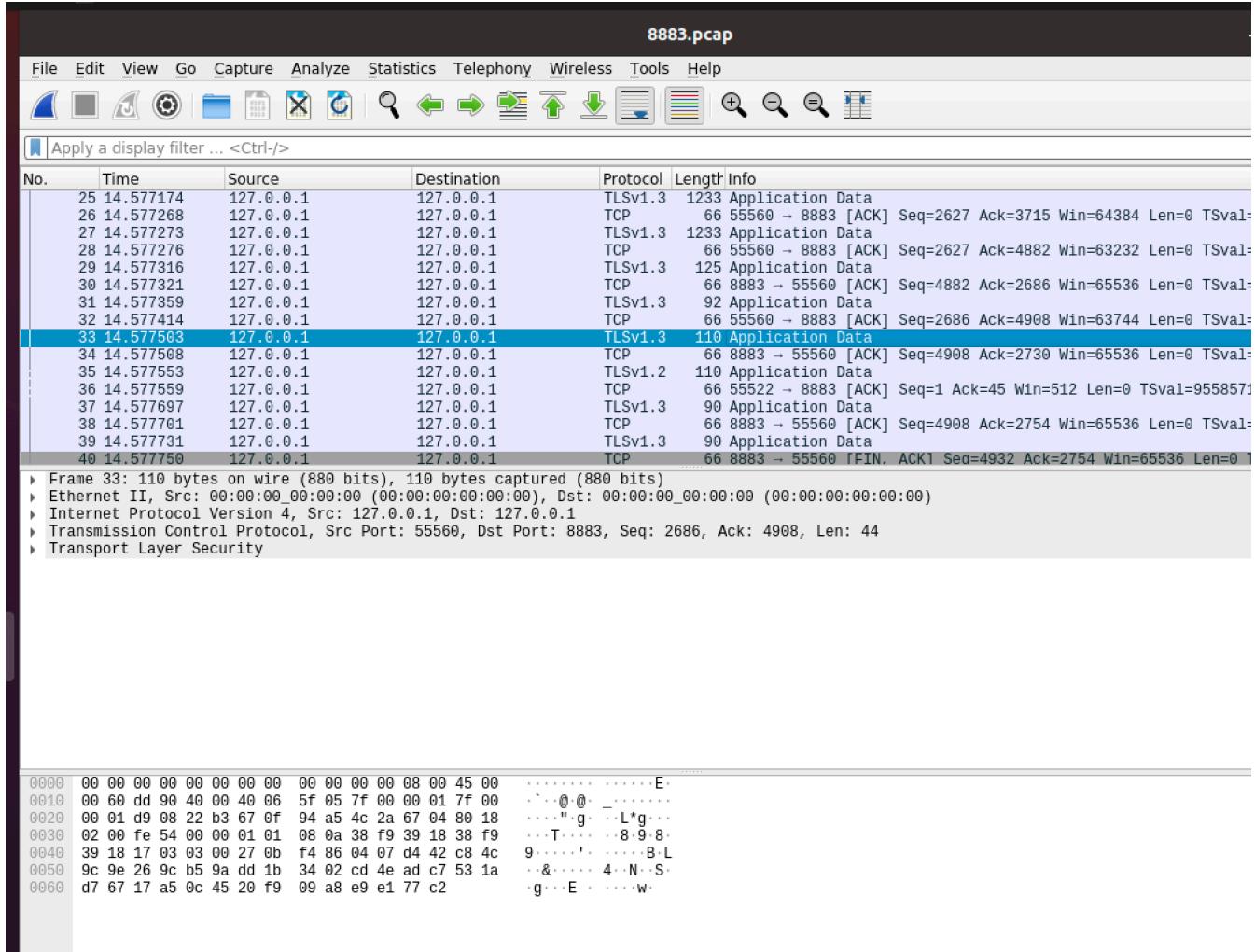


Рис. 8.5. Перехоплення

Необхідно перехопити дані між брокером та іншими учасниками спілкування за допомогою tcpdump, далі проаналізувати pcap файл в wireshark, і показати що з'єднання буде безпечним, дані шифровані.

8.4. Варіанти завдань

Завдання виконуються індивідуально, із власними даними та відповідними налаштуваннями системи.

8.5. Контрольні питання

- Як працює схема взаємодії Клієнт-Брокер-Публікатор-Підписник. В чому переваги такої схеми?
- До якого рівня моделі OSI належить MQTT?
- Які альтернативи MQTT ви можете вказати?
- Які основні кроки слід зробити, щоби налаштувати криптографічно захищенну передачу інформації?
- Для чого в роботі використовуються tcpdump та wireshark?

Література

1. Risk Management and Critical Infrastructure Protection: Assessing, Integrating, and Managing Threats, Vulnerabilities and Consequences
URL: <https://www.everycrsreport.com/reports/RL32561.html>
2. A Framework for Critical Information Infrastructure Risk Management.
URL:
<https://query.prod.cms.rt.microsoft.com/cms/api/am/binary/REVmc7>
3. K.Mali. Everything you need to Know about Linear Regression. URL:
<https://www.analyticsvidhya.com/blog/2021/10/everything-you-need-to-know-about-linear-regression/>
4. MQTT: The standard for IoT Messaging. URL: <https://mqtt.org/>
5. CVSS. URL: <https://www.first.org/cvss/calculator/3.0> .
6. Bluetooth Sniffing with Ubertooth: A Step-by-step guide. URL:
https://wiki.elvis.science/index.php?title=Bluetooth_Sniffing_with_Ubertooth:_A_Step-by-step_guide
7. Internet of Things – Contiki. URL:
https://www.tutorialspoint.com/internet_of_things/internet_of_things_contiki.htm
8. What is the Modbus Protocol & How Does It Work? URL:
<https://www.ni.com/en-us/shop/seamlessly-connect-to-third-party-devices-and-supervisory-system/the-modbus-protocol-in-depth.html>