Київський національний університет імені Тараса Шевченка
Факультет радіофізики, електроніки та комп’ютерних систем
Кафедра комп'ютерної інженерії

МЕТОДИ ВИЯВЛЕННЯ МЕРЕЖЕВИХ АНОМАЛІЙ В ІОТ СИСТЕМАХ

Дипломна робота магістра
студента 2 року навчання
спеціальність: 123 «Комп’ютерна інженерія»
Сергія МАМОТЕНКА

Науковий керівник
канд. фіз.-мат. наук Юрій БОЙКО,
доцент кафедри комп’ютерної інженерії

Рецензент
доктор. фіз.-мат. наук Євген ІВОХІН,
професор кафедри системного аналізу
та теорії прийняття рішень

До захисту допускаю: Завідувач кафедрою

Юрій БОЙКО

Ухвалено на засіданні кафедри “____” _____ 2022 р., протокол № ___

Київ - 2022

РЕФЕРАТ

Обсяг роботи 40 сторінок, 24 ілюстрацій, 2 таблиці, 11 джерел посилань.

ІНТЕРНЕТ РЕЧЕЙ, АТАКА, АНОМАЛІЯ, СТАТИСТИЧНИЙ МЕТОД, НАКОПИЧУВАЛЬНА СУМА, АВТОКОДУВАЛЬНИК.

Об’єктом роботи є аномалії в системах Internet of Things(IoT).

Метою роботи є дослідження методів виявлення мережевих аномалій та їх застосування в IoT.

Методи дослідження: аналіз статистичних методів, застосування нейронних мереж для виявлення аномалій.

Результат роботи: виконано загальний огляд архітектури IoT; проаналізовано основні проблеми безпеки та розглянуто вразливості IoT; розглянуто методи виявлення аномалій та обґрунтовано недоліки використання статистичних методів, запропоновано використання нейронних мереж.

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАЧЕННЯ

IoT – Internet of Things, Інтернет речей;

IDS - Intrusion Detection System, система виявлення вторгнень;

ПЗ – програмне забезпечення;

DoS – Denial of service – відмова обслуговування;

CUSUM - CUmulative SUM — накопичувальна сума;

ОС – операційна система;

MA – Moving Average – рухоме(ковзне) середнє.

ЗМІСТ

ВСТУП

Оцінка сучасного стану об'єкта дослідження. В поточний момент часу Інтернет речей стрімко розвивається та впроваджується в таких сферах діяльності, як догляд за дітьми та людьми похилого віку, готельний бізнес, сільське господарство, гірничодобувна промисловість, освіта і транспортні перевезення.

Технологічні експерти та аналітики прогнозують ще більше поширення пристроїв та додатків Інтернету речей. При цьому інформаційна безпека IoT не завжди відповідає необхідним вимогам. Атаки на системи Інтернету речей можуть привести до втрати конфіденційної інформації, виведення з ладу інфраструктури та фінансових збитків.

Багато компаній пропонують свої програмні рішення для аналізу та контролю даних в IoT, наприклад, хмарні сервіси Microsoft Azure Sphere, GREYCORTEX та AWS IoT Device Defender, що використовують виявлення атак, засноване на сигнатурах.

Актуальність роботи та підстави для її виконання. При використанні сигнатурних методів нерідко виникають проблеми з виявленням атак невідомих типів та постійним оновленням баз сигнатур. Ці проблеми можуть бути усунені використанням методів виявлення аномалій, серед яких найбільш популярні нейронні мережі та статистичні методи.

В роботі розглядаються як статистичні методи, так і нейронні мережі, проводиться порівняльний аналіз для дослідження їх ефективності в системах IoT.

Мета й завдання роботи. Метою кваліфікаційної роботи є дослідження методів виявлення аномалій в системах Інтернету речей.

Для досягнення цієї мети поставлено такі завдання:

- розглянути вразливості в безпеці IoT;
- розглянути методи виявлення аномалій, як підходу до виявлення мережевих атак;
- дослідити особливості методів виявлення аномалій в системах IoT.

Об’єкт, методи й засоби розроблення. Об’єктом дослідження є аномальна активність в системах Інтернету речей та методи, за допомогою яких цю активність можна виявити.

Дослідженню аномальної активності передувало виконання мережевих атак на відмову в обслуговуванні (DoS), а також MQTT flood-publish. В якості моделі IoT було використано дві віртуальні машини Debian, одна з яких виступала в якості пристрою Інтернету речей з обмеженою обчислювальною потужністю, інша в якості MQTT-брокера (сервера). Проведення та реєстрація атаки виконувались наступними програмними засобами: Wireshark, MQTT-брокер Eclipse Mosquitto, а також malaria - набір інструментів для перевірки середовища MQTT під час навантаження.

Можливі сфери застосування. Обраний метод виявлення аномалій може використовуватись в пристроях Інтернету речей з обмеженими обчислювальним потужностями.

РОЗДІЛ 1. ПРИНЦИПИ ПОБУДОВИ СИСТЕМ INTERNET OF THINGS(IOT)

1.1 Визначення IoT

У 2013 році Глобальна ініціатива по стандартизації в Інтернеті речей (IoT-GSI) визначила IoT як «глобальну інфраструктуру для інформаційного суспільства, яка надає розширені послуги шляхом об'єднання (фізичних і віртуальних) речей на основі існуючих і тих, що розвиваються, інтероперабельних інформаційно-комунікаційні технологій» і для цих цілей «річчю» є «об'єкт фізичного світу або інформаційного світу, який може бути ідентифікований та інтегрований в комунікаційні мережі» [1].

1.2 Еталонна модель IoT

На рисунку 1 показана еталонна модель IoT. Вона включає в себе чотири рівні[2]:

- прикладний рівень;
- рівень підтримки послуг і підтримки застосувань;
- мережевий рівень;
- рівень пристрою.

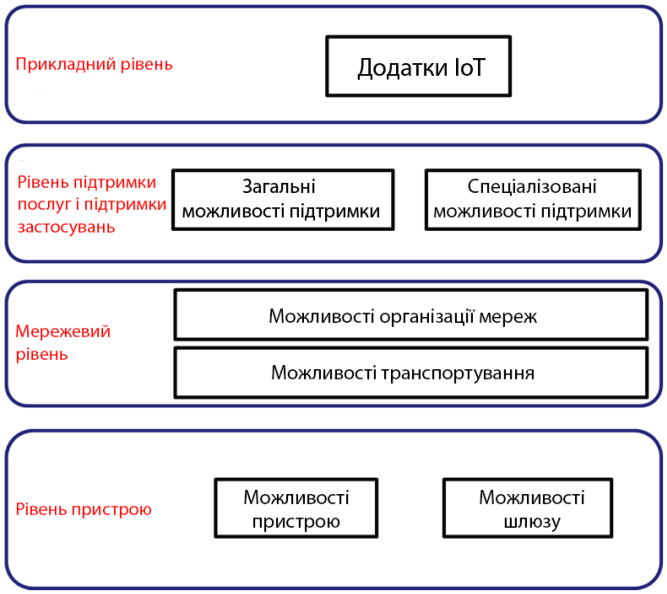

Рис.1 Еталонна модель IoT

В роботі використовується трирівнева модель, де рівень підтримки послуг і підтримки додатків об’єднаний з прикладним рівнем.

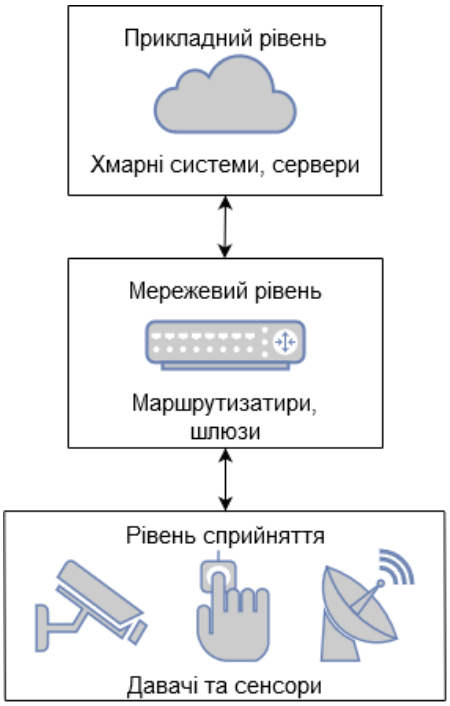

Рис. 2 Трирівнева модель IoT

1.2.1 Рівень сприйняття

Головна мета цього рівня – ідентифікація різних явищ навколишнього середовища за допомогою давачів з вбудованими сенсорами, та перетворення величини, що контролюється (тиск, температура тощо) в електричний сигнал зручний для вимірювання, передавання та перетворення.

1.2.2 Мережевий рівень

Шлюз або мережевий рівень підтримує зв’язок на рівні сприйняття, відповідає за збір, фільтрацію та передачу даних на сервер. Шлюзи полегшують зв’язок між давачами та іншою системою, перетворюючи дані з сенсорів у формати, які легко передаються та використовуються іншими компонентами IoT. Вони фільтрують та мінімізують інформацію, яка передається на хмарні сховища, що позитивно впливає на швидкість передачі даних в мережі та час реакції системи.
Шлюз та мережевий рівень також можуть підтримувати HTTP-сервер та брокер MQTT для забезпечення зв’язку між пристроями.

1.2.4 Прикладний рівень

Прикладний рівень (або хмара) призначений для зберігання та обробки великих обсягів даних для більш глибокого аналізу з використанням потужних механізмів обробки даних та механізмів машинного навчання.
Цей рівень орієнтований на користувачів, та представляє різні IoT додатки, які включають розумний будинок, розумний транспорт тощо.

1.3 Протоколи IoT

Наведена в роботі архітектура Інтернету речей передбачає наявність трьох функціональних рівнів. Оскільки нижній рівень складається з давачів та сенсорів, то необхідні спеціальні протоколи для забезпечення взаємодії цих пристроїв між собою та верхніми рівнями. Стандартні прикладні протоколи не пристосовані до умов мережі IoT. Давач, зазвичай невеликий пристрій з малим об’ємом оперативної пам'яті, вимірює та/або контролює певну величину в режимі реального часу. Результати роботи обробляються шлюзом і передаються на сервер. Кількість інформації, що надходить на шлюз від одного давача невелика, однак Інтернет речей обробляє інформацію з великої кількості вузлів.

Таким чином, варто розглянути мережеві протоколи, розроблені спеціально під вимоги Інтернету речей – CoAP і MQTT.

Constrained Application Protocol

CoAP (англ. Constrained Application Protocol) – мережевий протокол, оптимізований для мереж з обмеженими потужностями.

CoAP був розроблений на основі протоколу HTTP з урахуванням низької обчислювальної потужності пристроїв Інтернету речей. На відміну від протоколу HTTP, який використовує TCP, CoAP працює поверх UDP, завдяки чому зменшується розмір службових даних і підвищується швидкість їх передачі.

Протокол орієнтований на модель взаємодії «клієнт-сервер». Взаємодія з ресурсами відбувається за допомогою тих самих методів, які використовуються в HTTP: GET, PUT, POST і DELETE.

Застосування протоколу CoAP дозволило зменшити вимоги до пропускної здатності комунікаційного каналу, завдяки чому можна використовувати навіть низько швидкісне модемне з’єднання. Структура системи є горизонтальною, тобто такою, що дозволяє розширити конфігурацію на будь-якому рівні.

Message Queuing Telemetry Transport

MQTT (англ. Message Queue Telemetry Transport) — легковагий протокол, що працює над стеком TCP/IP.

Протокол був стандартизований в 2016 році та вже встиг набути широкого поширення в промисловості та IoT. Він був спеціально розроблений максимально компактним, для нестабільних Інтернет-каналів і малопотужних пристроїв, дозволяє гарантовано доставляти повідомлення в разі втрати пакетів або обриву каналу зв'язку.

MQTT використовує модель «видавець-підписник». Обмін повідомленнями відбувається через центральний сервер (брокер). У звичайних умовах клієнти не можуть спілкуватися безпосередньо один з одним, і весь обмін даними відбувається через брокер.

Клієнти можуть виступати в ролі постачальників даних (видавець) і в ролі одержувачів даних (підписник).

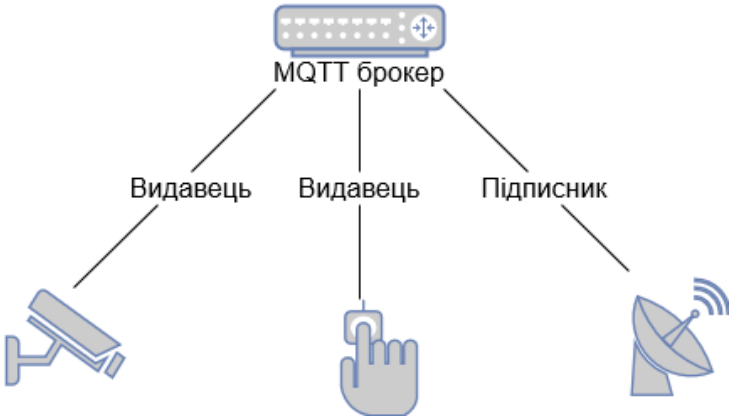

Рис. 3. Модель «видавець-підписник»

В MQTT передбачено три варіанти вибору надійності обміну повідомленнями, за що відповідають рівні QoS (англ. Quality of Service):

- QoS 0 – повідомлення передається тільки 1 раз, підтвердження не потрібне;
- QoS 1 – повідомлення передається мінімум 1 раз, підтвердження не потрібне;
- QoS 2 – для передачі використовується механізм four-part handshake між видавцем і підписником.

Модель «видавець-підписник» яку використовує протокол MQTT передбачає, що до мережі Інтернету речей буде підключена велика кількість пристроїв, що передають дані через центральний сервер – MQTT-брокер. Централізований характер робить IoT потенційно вразливим до мережевих атак, зокрема:

- підміна адреси відправника(spoofing);
- підміна даних;
- відмова в обслуговуванні(DoS та DDoS).

Для аналізу цих атак варто розглянути основні вразливості та проблеми безпеки в мережах IoT.

РОЗДІЛ 2. ВРАЗЛИВОСТІ IOT СИСТЕМ

На сьогоднішній день існують мільярди пристроїв, що з'єднують аналоговий світ з Інтернетом, і за короткий час це число збільшується, роблячи IoT найбільшим об'єктом для атак на планеті [3].

2.1 Визначення атаки

Законодавство України визначає кібератаку як цілеспрямовані дії в комп’ютерних мережах, які здійснюються за допомогою певних програмних або апаратних засобів для:

- отримання інформації, порушення її цілісності, несанкціонованого доступу до мережевих ресурсів;
- виведення з ладу комунікаційних комп’ютерних систем та мереж, та/або використання їх ресурсів для зловмисних дій [4].

Існують такі типи атак:

- Розвідка. Ці атаки включають ping sweeps, сканування та TCP або UDP портів.
- Експлойт (англ. exploit) – скрипт, що шукає і використовує вразливості в ПЗ комп’ютерної системи для отримання контролю над нею, або для виведення за ладу.

- Відмова в обслуговуванні (англ. Denial of Service) – атака комп'ютерної системи великою кількістю запитів з метою її перевантаження і виведення з ладу.

2.2 Особливості атак на IoT

Останні DDoS-атаки були спричинені, зокрема, пристроями IoT.
Безпека IoT систем і звичайних комп'ютерних систем відрізняється на основі наступних факторів [5]:

Навколишнє середовище. Одна з проблем безпеки систем IoT полягає в тому, що її компоненти розподілені в просторі та часто встановлені в публічних локаціях. Таке розташування пристроїв робить їх вразливими до несанкціонованого доступу.
Зловмисник може скопіювати дані та налаштування, а також підключити до IoT пристрій для прослуховування або зниження продуктивності мережі.

Обсяг пристроїв. Мільярди пристроїв IoT порівняно з мільйонами IT-пристроїв, підключених до корпоративних систем.
Очікується, що кількість активних пристроїв IoT зросте до 30,9 мільярдів в 2025 році [6]. Ці дані включають всі активні з'єднання і не враховують пристрої, які були придбані в минулому, але вже не використовуються.
На Землі мешкає 7,6 мільярдів людей (якщо рахувати наших східних сусідів за людей). У 4 мільярдів з них є доступ до Інтернету. Виходить, що на кожну людину припадатиме 7,725 пристроїв Інтернету речей.

Небезпечна передача і зберігання даних. Відсутність шифрування або контролю доступу до конфіденційних даних в будь-якому місці системи, в тому числі при зберіганні, передачі та під час обробки.
Пристрої Інтернету речей зберігають дані про навколишнє середовище, в тому числі персональну та конфіденційну інформацію.
IoT-пристрої можуть не тільки зберігати дані в незашифрованому вигляді, а також передавати їх по відкритим каналам зв'язку.

Небезпечні мережеві сервіси та служби. Мережеві сервіси та служби, запущені на самому пристрої, особливо ті, що відкриті для зовнішньої мережі, вразливі до атак на отримання несанкціонованого доступу та атак на відмову в обслуговуванні.

Що стосується IoT, більше всього атакам піддаються служби TCP. Популярними цілями є служби віддаленого адміністрування, такі як SSH, Telnet, VNC і RDP, а також бази даних та веб-сервери [7].

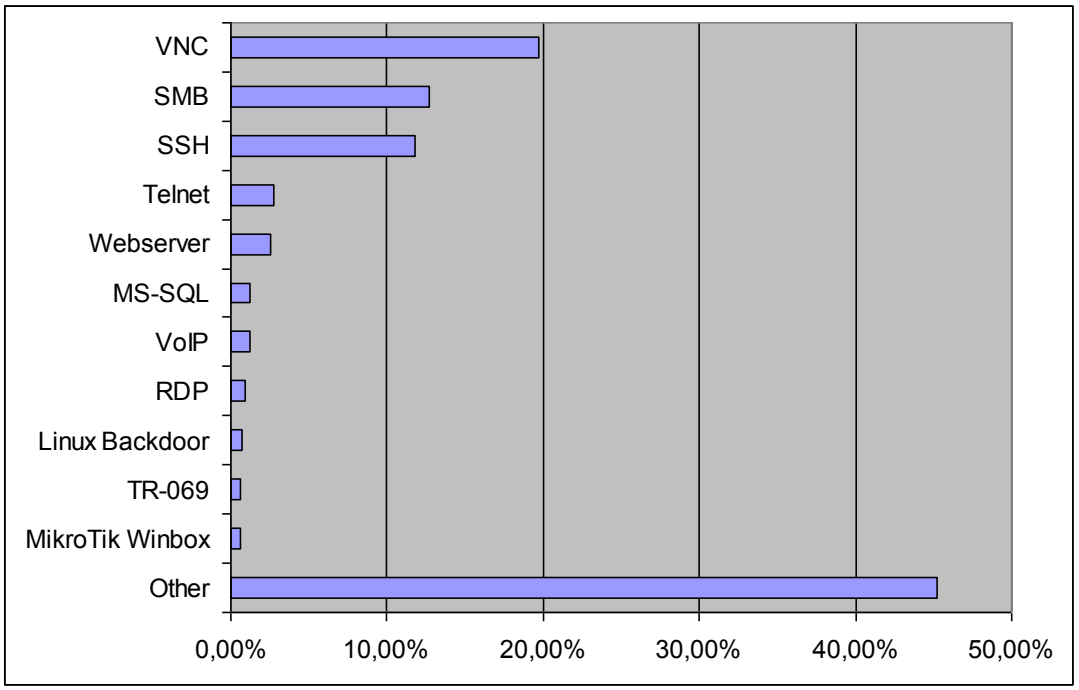

Рис.4. Найбільш атаковані служби в IoT.

Прикладом атаки, що використовувала вразливість відкритих портів SSH/Telnet є ботнет Mirai.

Ботнет – мережа підключених до інтернету пристроїв, що задіяні для виконання скоординованих DDoS-атак.

2.4 Методи виявлення атак

З існуючих підходів до виявлення мережевих атак виділяють: аналіз заголовків мережевих пакетів і аналіз їх вмісту [8].

Аналіз вмісту пакету дозволяє найбільш точно виявити атаку, однак обмежений складністю аналізу великих об’ємів даних. Тому в більшості

випадків доцільно використовувати для виявлення атак методи аналізу мережевого трафіку, в деяких випадках поєднуючи їх з аналізом контенту.

Серед основних методів виявлення атак виділяють сигнатурні методи та методи виявлення аномалій.
Сигнатура – це набір правил, за якими IDS виявляє мережеві атаки і незвичні характеристики трафіку.
Хоча система виявлення атак на основі сигнатур може легко виявляти відомі атаки, їй важко виявити нові атаки, для яких не існує шаблону. Також, треба постійно оновлювати бази даних для отримання сигнатур нових атак.

2.4.1 Програми аналізу та моніторингу мережевого трафіку

Збір і подальший аналіз різноманітної статистики (швидкість, обсяги переданих даних і т. д.) можливий за допомогою аналізаторів трафіку.

Аналізатор трафіку (інколи **сніфер –** від англійського слова *sniff* - *нюхати*) - це комп'ютерна програма або мережевий пристрій, що отримує та аналізує трафік комп’ютерної мережі [9].

Отримання трафіку здійснюється:

- прослуховуванням певного мережевого порту;
- розподіленням копії трафіку на сніфер;

Приклади аналізаторів трафіку:

tcpdump — це комп'ютерна програма для аналізу пакетів даних, що має консольний інтерфейс. Дозволяє користувачеві відображати TCP/IP та інші пакети, що передаються через мережу, до якої підключений комп'ютер. tcpdump - вільне програмне забезпечення.

Wireshark (застаріла назва Ethereal) — програма для аналізу мережевих пакетів (сніфер) з вільним вихідним кодом.

Функціонал, який Wireshark, дуже схожий з програмою tcpdump, але окрім того має ще й графічний інтерфейс з можливостями фільтрування інформації. Wireshark — це програма, яка препарує мережаві протоколи, і дозволяє розібрати мережевий пакет до байтів. Wireshark вміє і може працювати з безліччю форматів даних, наприклад файлами інших програмам аналізаторів трафіку.

2.4.2 Тестування програмного забезпечення

В якості програмного забезпечення було використано: Kali Linux - дистрибутив Linux, призначений для цифрової криміналістики і тестування на вторгнення; програми Wireshark та hping3, інтегровані в цей дистрибутив;

TCP-SYN Flood Attack

При проведенні TCP-SYN Flood атаки зловмисник повинен інтенсивно відправляти SYN-пакети з підробленими IP-адресами. Атакована система змушена реагувати, відправляючи у відповідь на кожний такий помилковий запит пакет SYN-ACK. Виділяється частина ресурсів, а порти залишаються відкритими в очікуванні численних відповідей (пакетів з встановленим прапором ACK) від хостів, яких насправді навіть не існує.

Для проведення цієї DoS-атаки, використовується дистрибутив Kali Linux, і програма hping3 — популярний TCP-інструмент тестування проникнення, що включений в набір Kali Linux.

Наступна команда розпочинає атаку на цільовий сервер (192.168.198.137):
«#hping3 -i u20000 -S -p 1883 --rand-source 192.168.198.137»

Пакети відправляються з інтервалом 20000 мікросекунд (-i u20000), вказується, прапор SYN (-S) повинен бути включений. Щоб направити атаку на MQTT-брокер, вказується порт 1883 («-p 1883»). Прапор (--rand-source) необхідний для генерації випадкових IP-адрес, завдяки чому приховується

реальне джерело, унеможливлюється відправка пакетів з прапорами SYN і ACK від жертви до зловмисника.

Для ідентифікації атаки TCP-SYN Flood використовується Wireshark. Початок атаки можна ідентифікувати по різкому збільшенню потоку TCP-трафіку.

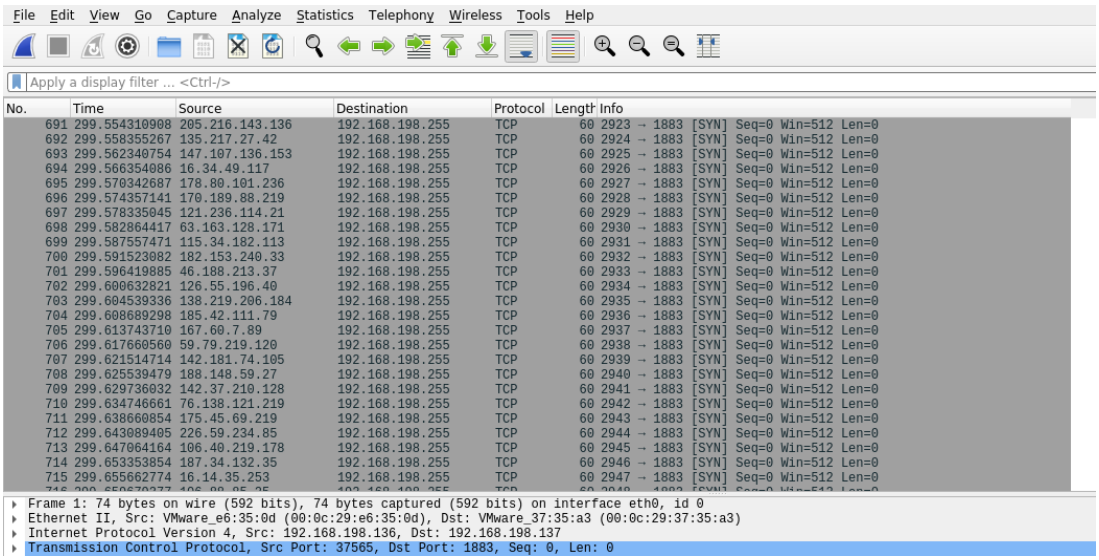

Рис. 5. Погляд на атаку TCP SYN Flood через Wireshark.

Далі трафік фільтрується за отриманими пакетами зі встановленим прапором SYN без подальшого підтвердження, наступним фільтром:
«tcp.flags.syn == 1 and tcp.flags.ack == 0»

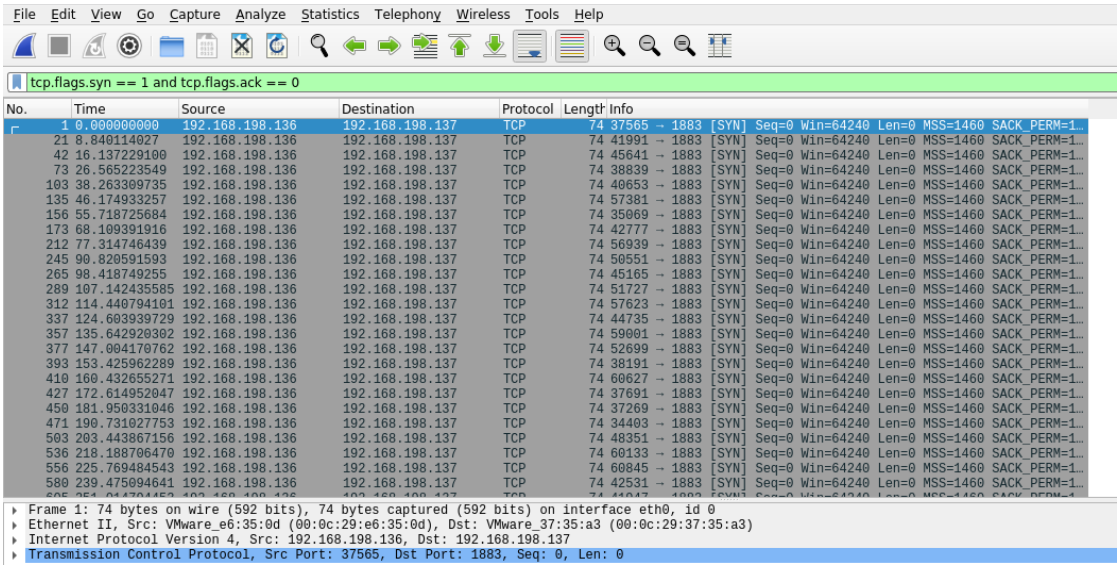

Рис. 6. з фільтрами: tcp.flags.syn == 1 та tcp.flags.ack == 0. Погляд на атаку TCP SYN Flood через Wireshark

На рисунку 6 видно велику кількість TCP-пакетів з встановленим прапором SYN без подальшого підтвердження і з різних джерел.

Якщо встановити фільтри tcp.flags.syn == 1 та tcp.flags.ack == 1, можна бачити, що кількість отриманих пар пакетів від клієнтів (відразу з встановленим прапором SYN, а потім - з прапором ACK) відносно невелика (детальніше на рисунку 7). Це ознака SYN Flood атаки.

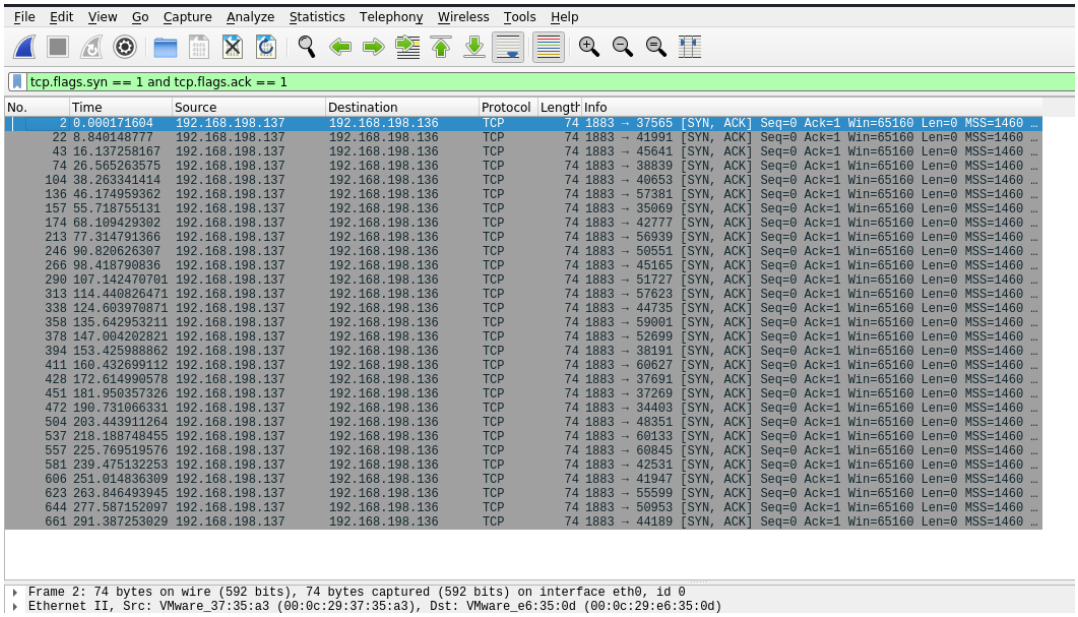

Рис. 7. Погляд на атаку TCP SYN Flood через Wireshark (фільтри tcp.flags.syn == 1 та tcp.flags.ack == 1).

Для візуального представлення зростання кількості трафіку використовується інструмент Wireshark - IO Graphs. Графік демонструє збільшення трафіку приблизно до 250 пакетів (рисунок 8).

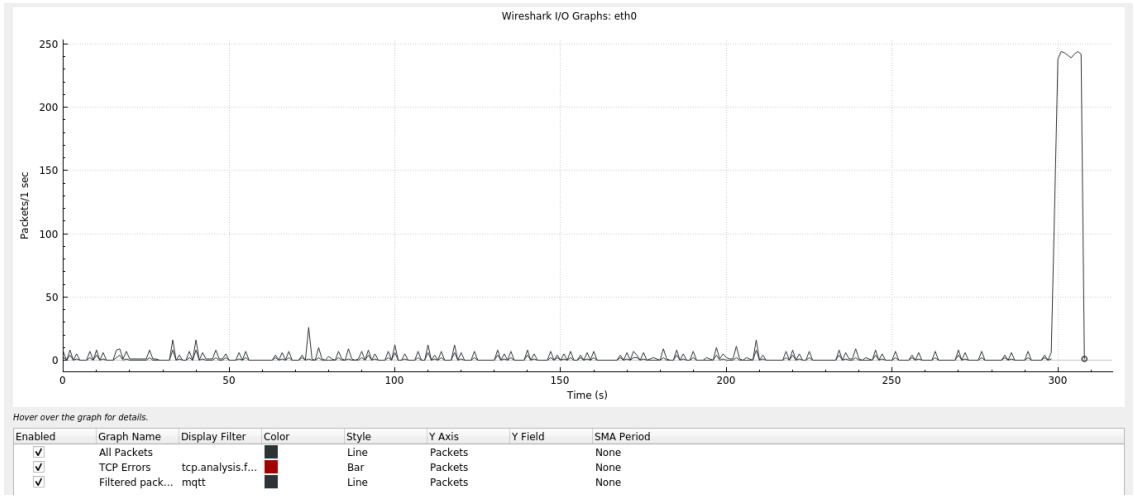

Рис. 8. Графік Wireshark.

РОЗДІЛ 3. ВИЯВЛЕННЯ АНОМАЛІЙ В IOT

Аномалія - щось, що відхиляється від того, що є стандартним, нормальним або очікуваним [10].

Методи виявлення аномалій спрямовані на виявлення невідомих атак спроб проникнення в систему (вторгнень). Для IDS формується шаблон нормальної поведінки. Сучасні IDS формують шаблон на основі:

- збору статистичної інформації;
- навчання нейронних мереж;
- формування сигнатур.

3.1 Статистичні методи виявлення аномального поведінки

Основний недолік сигнатурних методів виявлення мережевих атак, що пов'язаний з нездатністю цих методів виявляти атаки невідомих типів, може бути усунений застосуванням методів виявлення аномалій мережевої активності. Ці методи засновані на припущенні, що для обчислювальної системи існує деякий профіль нормальної поведінки і будь-які значні відхилення від нього є ознаками атаки. Основна перевага такого методу - можливість виявлення нових, невідомих раніше типів атак. Для побудови профілю нормальної поведінки системи використовується набір даних, вільний від аномалій, або статистичні методи.

Застосування методів статистичного аналізу є одним з найбільш поширених видів реалізації технології виявлення аномальної поведінки. Давачі збирають інформацію про стан і поведінку об’єкта, формуючи шаблон (профіль). Шаблон може формуватися на основі статистичних методів, таких як метод рухомого середнього (Moving Average) і метод накопичувальної суми (CUSUM). Для цього визначаються параметри, що характеризують нормальне функціонування мережі, і здійснюється динамічний контроль над її станом. В якості даних можуть використовуватись, наприклад, пакети MQTT або TCP.

Після побудови шаблону, з ним порівнюють поточний стан системи. Відхилення від цього шаблону є сигналом про наявність аномалії.

3.1.1 Контрольні карти

У контрольних картах зміна середнього значення характеризується центральною, верхньою і нижньою контрольними межами. Зміна виявляється, якщо значення перевищує одну з контрольних меж.

Зазвичай контрольні карти містять центральну лінію, що представляє середнє значення величини що спостерігається в нормальних умовах. По обидва боки від центральної лінії знаходяться верхня і нижня контрольні межі, що визначають діапазон відхилень величини.

3.1.2 Метод CUSUM.

Статистичний метод CUSUM (англ. CUmulative SUM – накопичувальна сума), заснований на виявленні точки зміни. В даному методі трафік, що піддається аналізу розділяється на параметри, наприклад, IP-адресу, номер порту або за протоколом. Після цього значення окремого параметра трафіку записується як елемент послідовності. При виконанні атаки поточний елемент послідовності суттєво відрізнятиметься від попереднього.

Перевагою методу є висока швидкодія, завдяки чому він може використовуватись в пристроях IoT в режимі реального часу.

3.2 Застосування методів CUSUM та Moving Average для виявлення мережевих аномалій в IoT

В роботі досліджуються статистичні методи виявлення аномалій за допомогою проведення DoS-атаки TCP-SYN Flood. Використовується спрощена модель IoT — дві віртуальні машини, одна в якості видавця — постачальника даних, інша в якості MQTT-брокера(сервера). Аналіз аномалій відбувався на брокері.

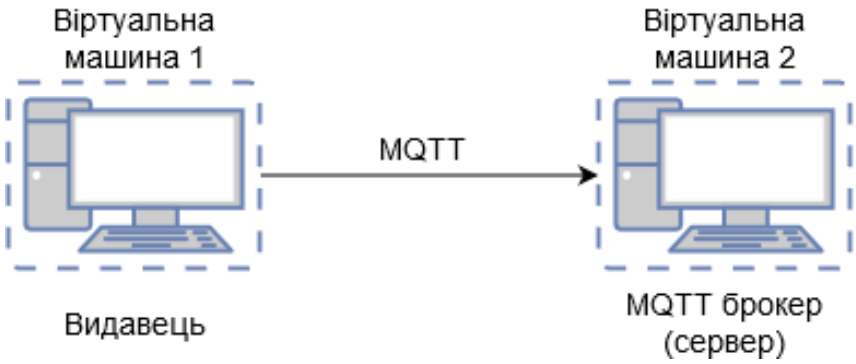

Рисунок 9. Модель IoT, використана в роботі

Всього було проведено двадцять тестів для виявлення аномалій. В перших десяти тестах встановлено інтервал між відправкою TCP-SYN пакетів – 0,35 с. В наступних десяти тестах інтервал був зменшений в п’ять разів до 0,07 с. Таким чином можна дослідити вплив інтенсивності атаки на час виявлення аномалії.

Далі розглядається перший тест з встановленим інтервалом відправки TCP-SYN пакетів 0,35 с. Для генерації TCP-SYN пакетів із заданим інтервалом виконується наступна команда:
hping3 -i u350000 -S -p 1883 --rand-source 192.168.198.137

Пакети TCP-SYN з випадковою IP-адресою джерела відправляються на порт 1883 MQTT-брокера.
Щоб зімітувати роботу системи за нормальних умов, видавець постійно відправляє невелику кількість MQTT пакетів на IP-адресу брокера.
Наступна команда вказує, що потрібно відправити 35 пакетів MQTT, по 2 пакети в секунду за вказаною адресою:
malaria publish -n 35 -T 2 -H 192.168.198.137

Під час атаки спостерігається зниження кількості MQTT трафіку, оскільки більшість ресурсів атакуючої системи використані для відправлення TCP-SYN пакетів (рисунок 10).

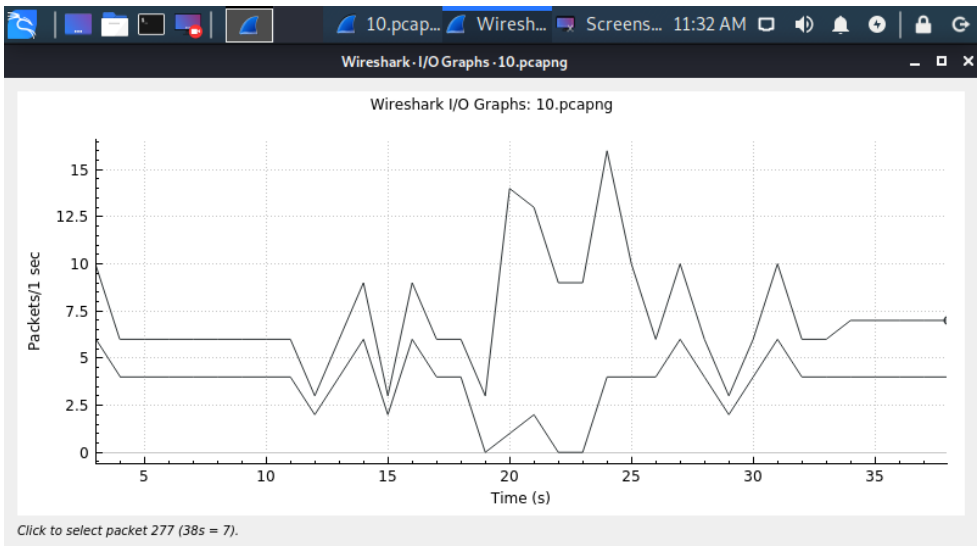

Рис. 10. Wireshark I/O Graph

Аналіз відбувався протягом 38 секунд. Атака почалася на 20 секунді і тривала 5 секунд. Дані про кількість пакетів було збережено у текстовий файл і форматовано у 3 стовбці для подальшого аналізу.

Для аналізу було обрано статистичні методи контрольних карт CUSUM та Moving Average, в силу їх швидкодії і незначних витрат оперативної пам'яті.

Графік CUSUM дозволяє спостерігати навіть незначні відхилення завдяки малим постійним відхиленням середнього значення процесу. Цей метод добре підходить для виявлення слабких атак.

В даному випадку кількість пакетів до і після атаки приблизно однакова, тому наочно виявити аномалію не вдасться(рис. 11).

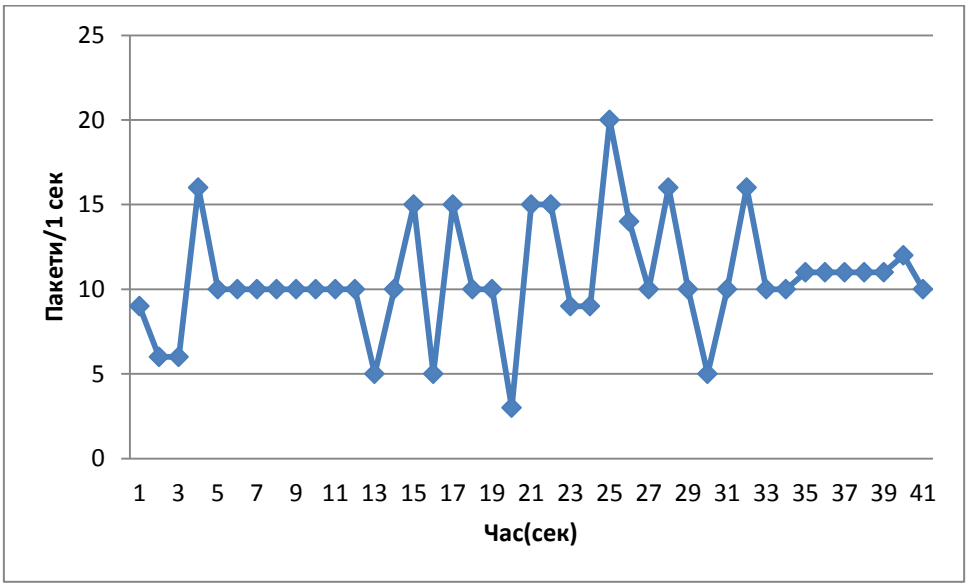

Рис. 11. Кількість пакетів

Розрахунок параметрів CUSUM для побудови графіку. Формула для виявлення позитивних змін середнього значення $u^{+}(y) = y - (\mu_0 + K)$, де K – опорне значення. Формула для виявлення негативних змін середнього значення: $u^{-}(y) = (\mu_0 - K) - y$.

В результаті отримано дві функції:

$C_i^{+} = \max [0, x_i - (T + K) + C_{i-1}^{+}]$

$C_i^{-} = \max [0, (T + K) - x + C_{i-1}^{-}]$

$K = \sigma/2$ і $h = 4\sigma$, де σ- стандартне відхилення.

Для зменшення часових витрат, аналізується не вміст мережевих пакетів, а їх кількісна характеристика - відношення TCP пакетів до загальної кількості всіх пакетів(TCP та MQTT).

Під час атаки TCP-SYN Flood IP-адреса відправника підміняється великою кількістю сторонніх адрес. Проте, атакуючий пристрій можна виявити за зменшенню на ньому вихідного MQTT-трафіку.

Результати обрахунку CUSUM наведені в додатку А. За отриманими значеннями побудовано графік.

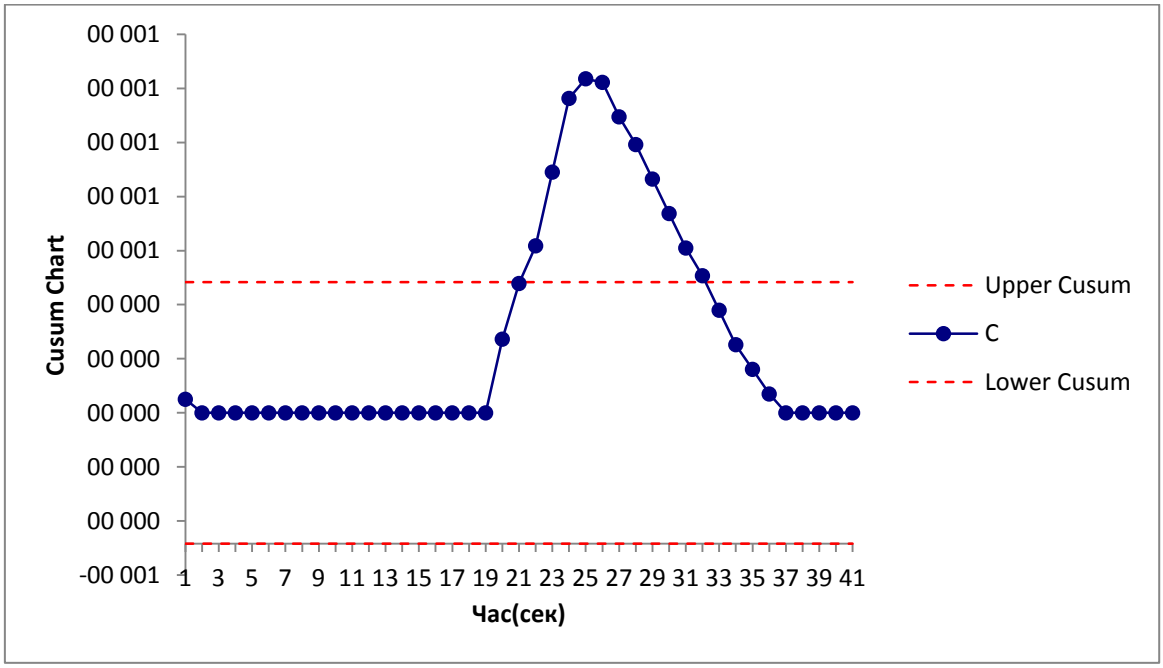

Рис. 12. Метод CUSUM

Графік функції CUSUM перетинає верхнє граничне значення на 21-й секунді. Аномалія виявлена за 1 секунду після початку атаки.

Для порівняння було застосовано метод рухомого середнього (англ. Moving Average). Це метод, який полягає в обчисленні середнього значення деякої підмножини. Застосовується в аналізі часових рядів (англ. timeseries) для згладжування випадкових відхилень та для підкреслення основних закономірностей.

Таблиця розрахованих значень для побудови графіку приведена в додатку Б.

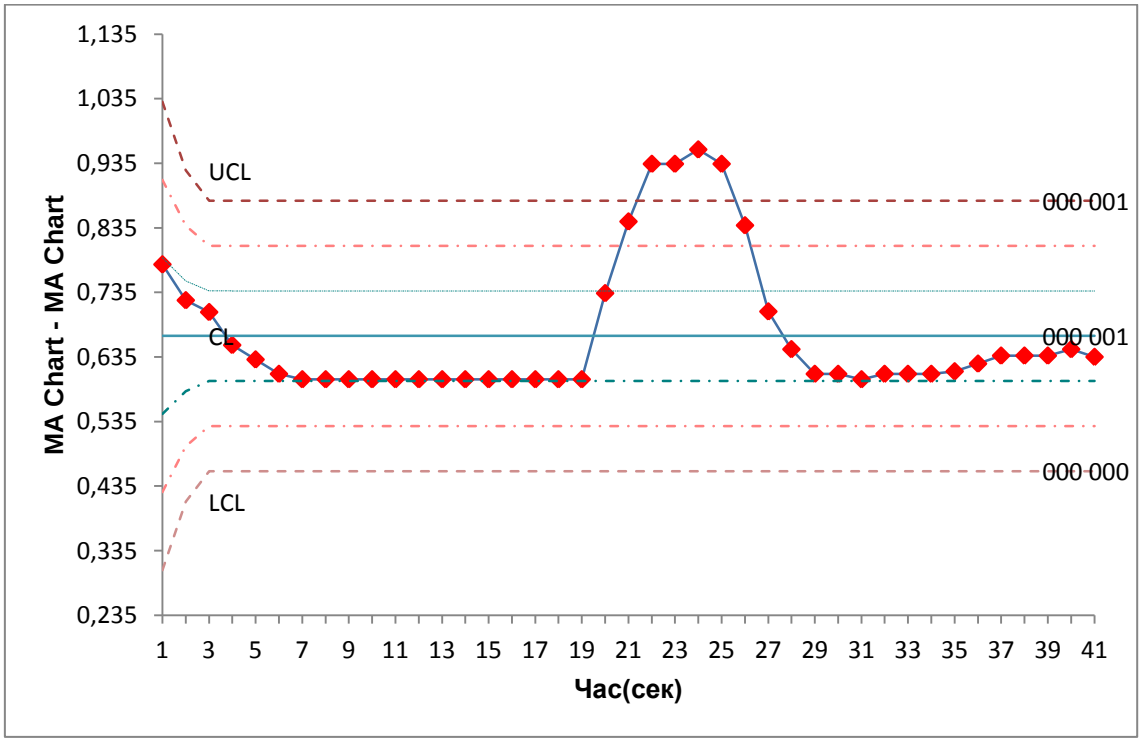

Рис 13. Метод Moving Average

Даний метод виявив аномалію на 0,4 с пізніше, ніж CUSUM. Такий результат пояснюється тим, що на графіку рухомої середньої попередні вибіркові значення змінної усереднюються разом з поточним значенням, що допомагає «згладити» вхідні дані, мінімізуючи вплив випадкових завад на процес. В цей же час, таке «згладжування» призводить до втрати частини даних і сповільнення виявлення аномалії.

Провівши наступні дев’ять тестів з аналогічними налаштуваннями, побудовано таблицю 1.

Таблиця 1 - Дані щодо перших десяти дослідів, с

Початок атаки	Кінець атаки	Час виявлення аномалії	
		CUSUM	MA
19,9	25	21	21,4
20,8	26,4	22,3	22,4
17,6	23,2	19,1	19,4
20,3	25,1	22,2	22,3
20,8	24,6	22,3	22,5
19,4	23,7	21,6	21,7
21,3	25,5	22,6	23
21	27	23,1	23,3
19,1	26,2	22	22
21,3	23,4	22,2	22,5

Розрахунок середнього часу виявлення аномалії наведено в додатку В.

Середній час виявлення аномалії після початку атаки:

- CUSUM — 1,7 с;
- Moving Average — 1,9 с.

В дев’яти з десяти випадків аномалія реєструється швидше методом CUSUM.

Зменшивши інтервал між пакетами в п’ять разів (0,07 с), проведено наступні десять дослідів.

Таблиця 2 - Дані щодо наступних десяти дослідів, с

Початок атаки	Кінець атаки	Час виявлення аномалії	
		CUSUM	MA
19,8	25,3	21,1	21,3
20,1	26,2	22,9	23,1
18,5	25,6	21	21
19	25	20,6	20,6
23,3	29,5	26	26
16,8	23,5	19,2	19,1
21,2	26,4	23,4	23,6
18,7	27,3	21,1	21,1
23,3	27,5	25,5	25,3
20	24	22,8	22,6

Середній час виявлення аномалії після початку атаки:

- CUSUM — 2,3 с;
- Moving Average — 2,3 с.

Методи CUSUM і Moving Average реєструють аномалію одночасно.
За результатами наступних десяти тестів виявлено, що метод CUSUM і метод Moving Average в виявляють аномалію однаково швидко.
Таким чином, можна зробити висновок, що метод CUSUM є більш ефективним при малих інтенсивностях атак, ніж метод Moving Average.

3.3 Застосування нейронних мереж для виявлення мережевих аномалій в IoT

Статистичні методи продемонстрували високу швидкодію у виявленні аномальної активності в IoT, однак їх недоліком є проблема (невизначеність) задання порогового значення, а також неможливість визначення точного часу початку і закінчення аномалії. Щоб позбутися цих недоліків, мережева активність додатково має аналізуватись спеціалістом по аномаліям. Ця процедура є недоцільною і трудомісткою, тому як альтернативний метод було запропоновано використання нейронних мереж.

Штучна нейронна мережа наближено імітує функції біологічного нейрона. Запропонована у 1943 році Мак-Каллокол і Піттсом.
Архітектура нейронних мереж:

- Вхідний шар (input layer) – шар вхідних сигналів
- Нейрони розташовуються по шарах (layers)
- У простому випадку нейронна мережа будується пошарово: виходи нейронів попереднього шару є входами наступного.
- Можливі складніші поєднання шарів

Нейронні мережі сучасних архітектур складаються з:

- шарів нейронів з вагами, що налаштовуються під час навчання мережі;
- шарів інших типів, які не містять ваг (параметрів), що налаштовуються під час навчання мережі, а виконують деяку операцію над сигналами (наприклад, pooling, dropout).

Функція втрат (loss function), функція помилок (error function) або функція вартості (cost function) – скалярна диференційована функція L(d, y) від вектору бажаних вихідних відгуків d і вектору їх фактичних відгуків y, що описує помилку мережі значення якої мінімізується алгоритмом навчання на кожній ітерації оновлення ваг W:

$$L(d, y) \rightarrow \min_{w}$$

Лінійна функція втрат розраховується як середня абсолютна помилка (MAE – mean absolute error):

$$L(d, y) = \frac{1}{N}\sum_{i=0}^{N}|d - y_i|$$

де y_i – прогнозоване значення

Лінійна функція втрат менше в порівнянні з квадратичною штрафує вихіний сигнал, що відрізняється від бажаного, тобто навчання мережі з використанням лінійної функції втрат є більш стійким до викидів у навчальній вибірці.

Глибоке навчання (deep learning) – такий підхід машинного навчання з вчителем, який передбачає автоматичне виділення під час навчання корисних ознак із повного набору вхідних сигналів, що описує об’єкт – навчання представлень. На відміну від класичного машинного навчання, коли людина повинна була вручну представляти ці ознаки. Після подання ознак, вони відображаються на вихідний сигнал.

Глибоке навчання передбачає використання глибоких нейронних мереж (deep neural network, DNN). Під DNN мається на увазі мережа з хоча б одним прихованим шаром. Має наступні недоліки:

- затухання градієнту;
- велика кількість ваг;
- велика чутливість до трансформації вхідного сигналу.

Мережі згортки (згорткові мережі, convolutional neural networks, CNN) - нейромережа прямого розповсюдження, в якій хоча б в одному шарі використовується математична операція дискретної згортки.

Замість того, щоб створювати набір нейронів, де в кожного нейрона свої ваги, в дискретній згортці використовується ядро (або фільтр) з автоматично налаштованими вагами. За допомогою операції дискретної згортки будується карта ознак (feature map) [11].

В дипломній роботі розглядаються деякі з основних різновидів нейронних мереж, зокрема автокодувальник.

Автокодувальник - це особливий тип нейронної мережі, який навчається відтворювати вхідні дані на вихідні. Наприклад, отримавши зображення рукописної цифри, автокодувальник спочатку кодує зображення в прихованому представленні більш низького розміру, а потім декодує це представлення, назад в зображення. Автокодувальник вчиться стискати дані, зводячи до мінімуму помилку відновлення.

Наприклад, отримавши на вхід зображення рукописної цифри, автокодувальник спочатку кодує зображення в представленні більш низького розміру, а потім декодує це представлення, назад в зображення. Автокодувальник вчиться стискати дані, зводячи до мінімуму помилку відновлення. Тому, якщо на вхід подається, наприклад буква, автокодувальник виконує відновлення значно гірше, що означає наявність аномалії.

Основний принцип роботи та навчання мережі автокодувальника - отримати на вихідному шарі набір даних, найближчий до вхідного. Щоб рішення не виявилося тривіальним, на прихований шар автокодувальника накладають обмеження: він повинен бути меншої розмірності, ніж вхідний і вихідний

шари. Завдяки цьому нейронна мережа вчиться виокремлювати істотні властивості і узагальнювати дані, виконуючи їх стиснення. Нейронна мережа автоматично навчається виділяти із вхідних даних загальні ознаки, які кодуються у значеннях ваг штучної нейронної мережі.

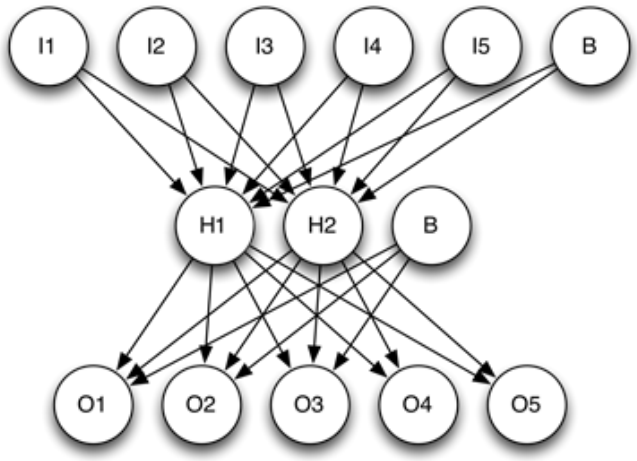

Рис 14. Архітектура автокодувальника

Автокодувальник має набір вхідних нейронів (inputs), рівний по кількості вихідним нейронам (outputs).

3.3.1 Виявлення аномалій за допомогою автокодувальника

Код нейронної мережі був написаний і викладений в Google Colaboratory[1].

Для створення набору даних була промодельована базова модель Інтернету речей, що складається з двох пристроїв. Замість реальних пристроїв використані дві віртуальні машини Debian - одна в якості MQTT-брокера, друга в якості IoT пристрою.

Був написаний bash скрипт, який запускається на брокері.

```
#!/bin/bash
mosquitto_pub -h 192.168.225.140 -m "hi bro" -t test1 --repeat 1000000 --repeat-delay 0.05 &
for (( i=1; i<=10; i++ ))
do
        sleep 40
        timeout 20s mosquitto_pub -h 192.168.225.140 -m "pls send us log ASAP" -t test2 --repeat 1000000 --repeat-delay 0.016 &
done
sleep 34
timeout 26s ./malaria publish -P 2 -n 600 -T 15 -H 192.168.225.140 -s 1 &
for (( i=1; i<=3; i++ ))
do
        sleep 40
        timeout 20s mosquitto_pub -h 192.168.225.140 -m "pls send us log ASAP" -t test2 --repeat 1000000 --repeat-delay 0.016 &
done
```

[1] https://colab.research.google.com/drive/1CasdFSS5X_srV23-PmDIh0PHr-zjhNvy#scrollTo=uMB1AC30AfW-

Рис 15. Bash-скрипт.

Він відправляє з деякою періодичністю запити на пристрій, щоб змоделювати типову поведінку системи IoT. За допомого malaria publish виконується MQTT flood-publish атака, тобто відправка великої кількості publish пакетів для перевантаження пристрою.

У виявленні аномалії за допомогою автокодувальника можна виділити такі кроки:

1. Завантаження даних

Було створено два набори даних (зразок наведений в додатку Г та Ґ):

- Тренувальний набір. Вибірка даних без аномальної активності для навчання нейронної мережі. В якості даних використовуються MQTT пакети, які були відправлені з сервера (брокера) на пристрій IoT.

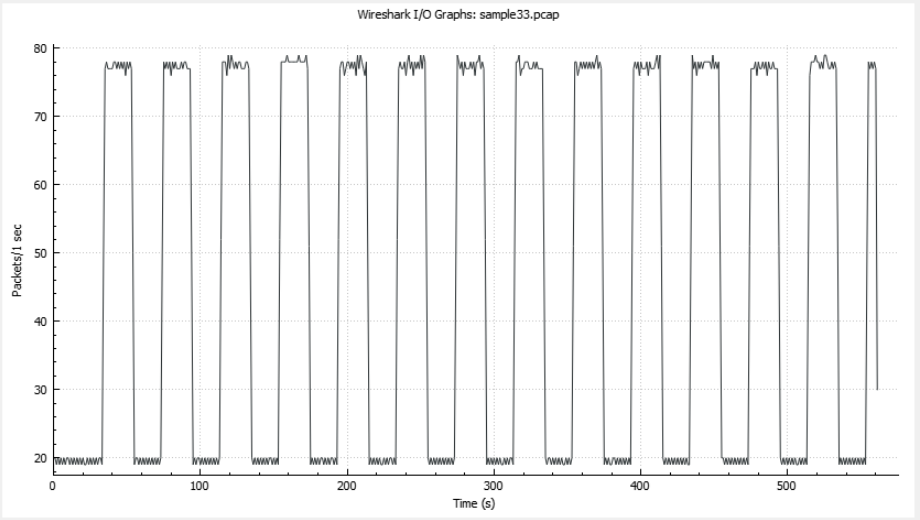

Рис 16. Тренувальний набір даних

- Тестовий набір. Вибірка даних, в якій зафіксована аномальна активність. Аномалія викликана MQTT flood-publish атакою. На графіку (рисунок 16) аномалія не помітна, однак проявляється при застосуванні фільтру mqtt.msgtype eq "Publish Ack". Це обумовлено тим, що під час цієї атаки

```
print(df_train.head())

print(df_test.head())
```

```
                     value
timestamp
2022-05-05 00:00:00     19
2022-05-05 00:00:01     20
2022-05-05 00:00:02     19
2022-05-05 00:00:03     20
2022-05-05 00:00:04     19
                     value
timestamp
2022-06-06 00:00:00   20.0
2022-06-06 00:00:01   19.0
2022-06-06 00:00:02   20.0
2022-06-06 00:00:03   19.0
2022-06-06 00:00:04   20.0
```

брокер відправляє велику кількість publish запитів, на які пристрій IoT має відповісти publish ack повідомленнями.

Рис 17. Перші п'ять рядків тренувального набору даних

Тестовий набір сформовано з MQTT пакетів, до яких додано відфільтровані MQTT publish ack пакети.

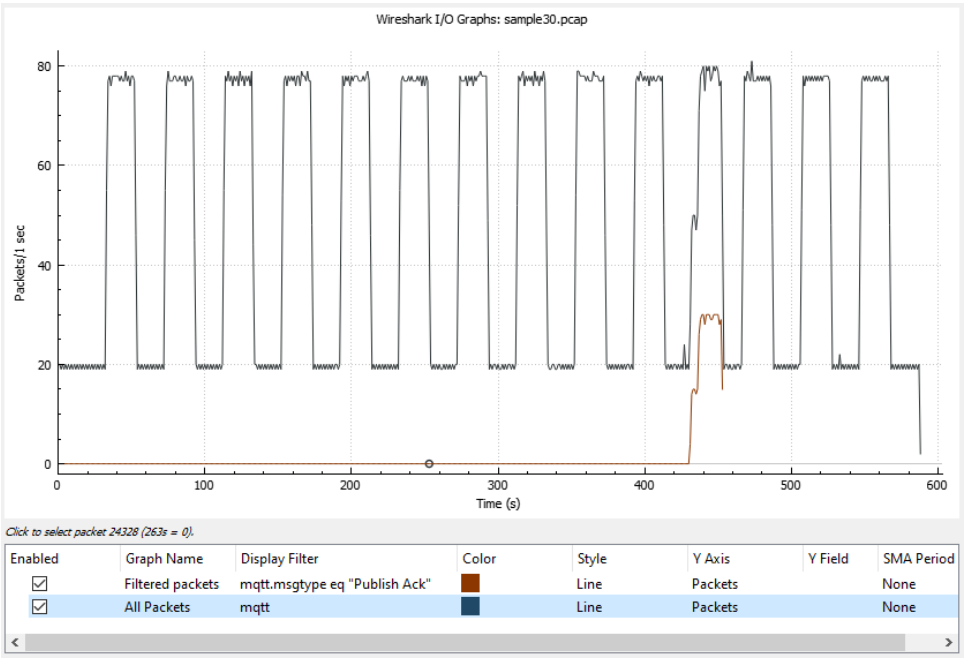

Рис 18. Тестовий набір даних

2. Попередня обробка даних

Дані тренувального набору нормалізуються шляхом використання функцій mean() і std(). Ненормалізовані вхідні дані можуть призвести до повільного або нестабільного процесу навчання мережі, що, в свою чергу, збільшить помилку узагальнення.

```
training_mean = df_train.mean()
training_std = df_train.std()
df_training_value = (df_train - training_mean) / training_std
print("Number of training samples:", len(df_training_value), df_training_value)
```

```
Number of training samples: 551                          value
timestamp
2022-05-05 00:00:00 -1.013243
2022-05-05 00:00:01 -0.977919
2022-05-05 00:00:02 -1.013243
2022-05-05 00:00:03 -0.977919
2022-05-05 00:00:04 -1.013243
...                       ...
2022-05-05 00:09:06  1.070877
2022-05-05 00:09:07  1.035553
2022-05-05 00:09:08  1.070877
2022-05-05 00:09:09  1.035553
2022-05-05 00:09:10 -0.624678
```

Рис 19. Перші п'ять рядків тренувального набору даних після нормалізації

3. Використання автокодувальника

Побудуємо модель автокодувальника згорткової реконструкції. Модель прийматиме на вхід (batch_size, sequence_length, num_features) і повертатиме на вихід дані такої ж розмірності. Довжина послідовності `sequence_length` дорівнює 40 (40 секунд триває один такт), а `num_features` дорівнює 1 (ознака – кількість пакетів).

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv1d (Conv1D)             (None, 20, 32)            256

 dropout (Dropout)           (None, 20, 32)            0

 conv1d_1 (Conv1D)           (None, 10, 16)            3600

 conv1d_transpose (Conv1DTra  (None, 20, 16)           1808
 nspose)

 dropout_1 (Dropout)         (None, 20, 16)            0

 conv1d_transpose_1 (Conv1DT  (None, 40, 32)           3616
 ranspose)

 conv1d_transpose_2 (Conv1DT  (None, 40, 1)            225
 ranspose)

=================================================================
Total params: 9,505
Trainable params: 9,505
Non-trainable params: 0
_________________________________________________________________
```

Рис 20. Модель автокодувальника

Навчання проводиться за 50 епох, в якості вхідних і цільових даних використовується одна й та сама послідовність.

4. Детектування аномалій

Аномалія виявлятиметься на основі того, як добре модель реконструює вхідні дані на вихід.

1. Знайдемо середню абсолютну помилку (MAE).
2. Знайдемо максимальне значення функції втрат MAE. Це найгірша спроба моделі відновити дані. Вона буде використовуватись для задання "порогу" аномалії.
3. Якщо функція втрат при відновленні даних перевищує порогове значення, можна стверджувати, що модель виявила незнайому закономірність. Її можна позначити як аномалію.

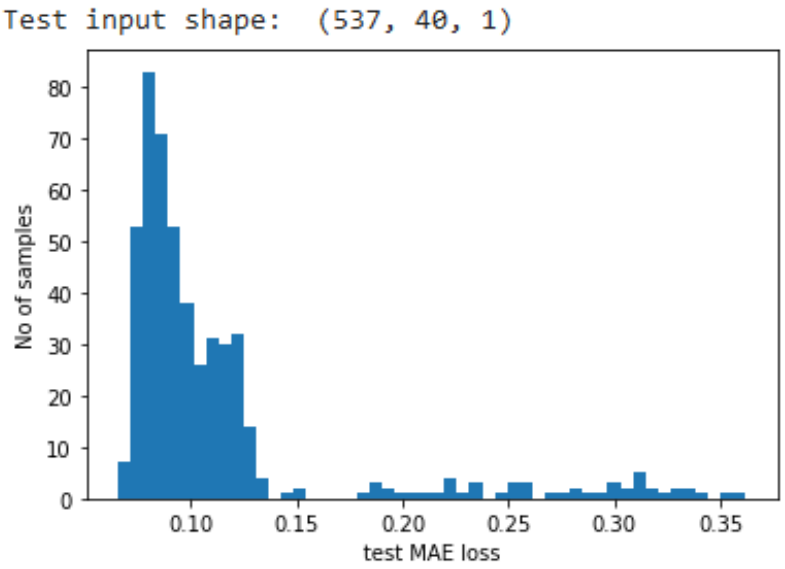

Рис 21. Відношення MAE до кількості даних

Кількість аномальних даних: 60

Індикація аномальних даних: (2, 82, 162, 242, 339, 386, … 439, 476).

Позначимо аномальні дані на графіку:

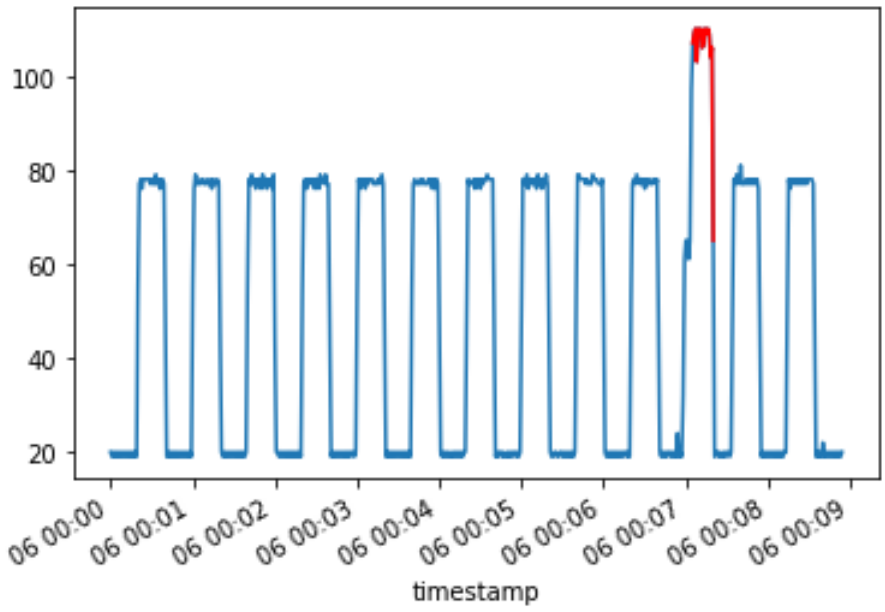

Рис 22. Аномальні дані виділені на графіку

3.3.2 Порівняння автокодувальника з методами CUSUM та Moving Average

Статистичні методи, зокрема CUSUM ефективні в безконтрольному режим лише у випадках, коли розмірність даних є низькою. Інакше, є ризик виникнення великої кількості хибних спрацювань, в підтвердження чого рисунок 22.

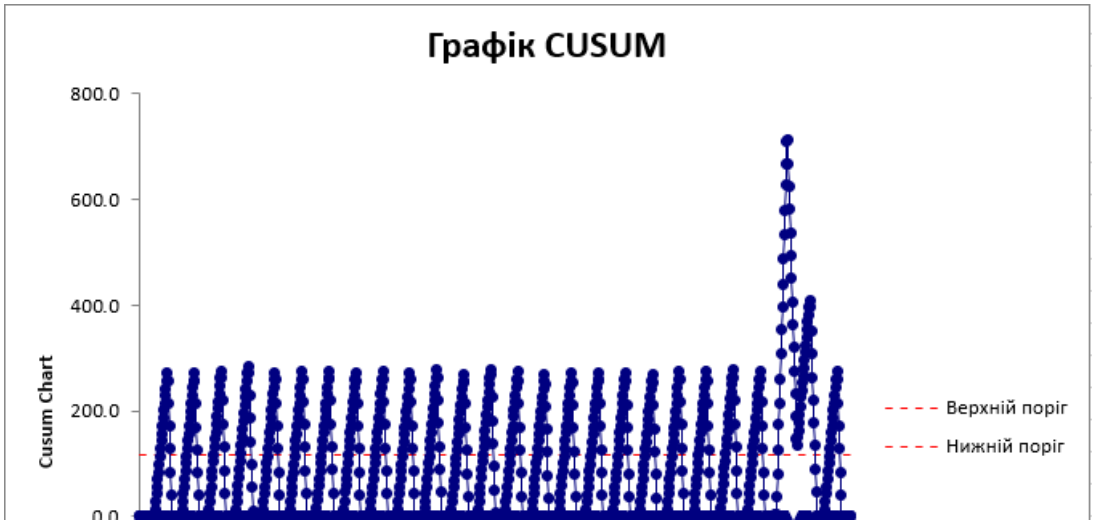

Рис 23. Графік CUSUM

Метод Moving Average зміг виявити аномалію, однак недоліком методу є неможливість визначення точного часу виникнення та закінчення аномалії. Її подальший аналіз потребує кваліфікованого спеціаліста, а також часових затрат.

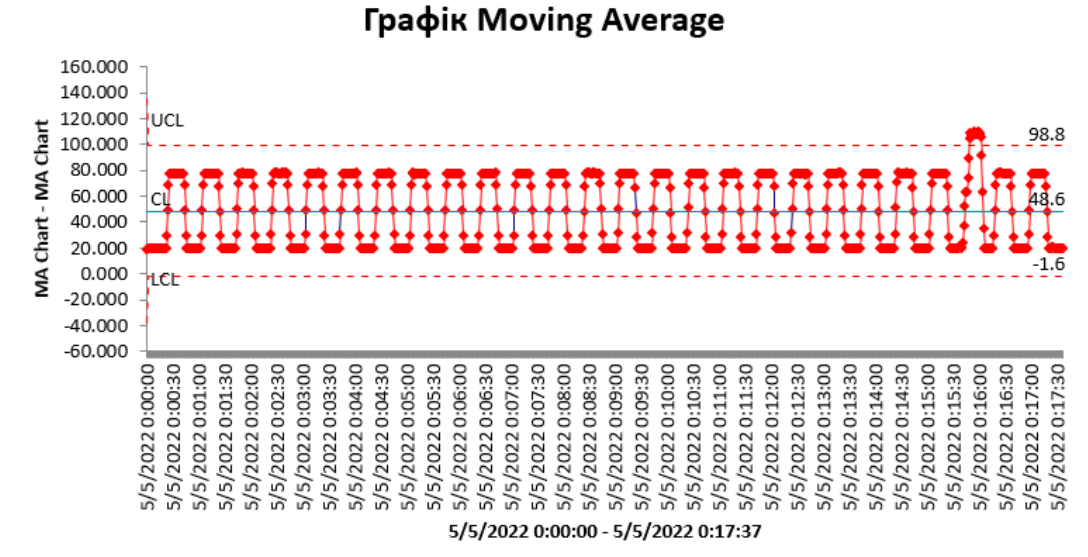

Рис 24. Графік Moving Average

ВИСНОВКИ

Аналіз архітектури систем IoT показав, що до них підключена велика кількість пристроїв, які можуть передавати персональну та конфіденційну інформацію. Для запобігання втрати цієї інформації варто передбачити механізми захисту від мережевих атак.

Запропоновано використання методів виявлення аномалій, як підходу до виявлення мережевих атак. Методи виявлення аномалій здатні виявляти атаки невідомих типів і не потребують постійного оновлення баз сигнатур.

Серед методів виявлення аномалій було обрано статистичні методи завдяки їх швидкодії та незначним витратам пам'яті. Це дозволяє використовувати їх в пристроях Інтернету речей з обмеженими обчислювальним потужностями.

Виявлено недолік статистичних методів, пов’язаний з заданням порогового значення, що ускладнює їх використання в умовах IoT. Був запропонований альтернативний метод – використання штучних нейронних мереж.

Результати випробувань показали, що нейронна мережа здатна виявляти мережеві аномалії більш достовірно, ніж статистичні методи CUSUM та Moving Average.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

[1] ITU-T. Recommendation ITU-T Y.2060 / ITU-T. – 2012. – С. 1.

[2] ITU-T. Recommendation ITU-T Y.2060 / ITU-T. – 2012. – С. 6–8.

[3] Perry Lea. Internet of Things for Architects / Perry Lea. – С. 386.

[4] Закон України «Про основні засади забезпечення кібербезпеки України» [Електронний ресурс] – Режим доступу до ресурсу: https://zakon.rada.gov.ua/laws/show/2163-19.

[5] OWASP IoT Top 10 - Exploring Vulnerability Root Causes [Електронний ресурс]. – 2018. – Режим доступу до ресурсу: https://owasp.org/www-pdf-archive/OWASP-IoT-Top-10-2018-final.pdf.

[6] Internet of Things (IoT) and non-IoT active device connections worldwide from 2010 to 2025 [Електронний ресурс] – Режим доступу до ресурсу: https://www.statista.com/statistics/1101442/iot-number-of-connected-devices-worldwide/.

[7] IoT: a malware story [Електронний ресурс] – Режим доступу до ресурсу: https://securelist.ru/iot-a-malware-story/94900/.

[8] Шелухин О. И. Обнаружение вторжений в компьютерной сети / О. И. Шелухин, Д. Ж. Сакалема, А. С. Филинова. – С. 63.

[9] Kevin J. Connolly. Law of Internet Security and Privacy / Kevin J. Connolly, 2003. – С. 131.

[10] Definition of anomaly in English: [Електронний ресурс] – Режим доступу до ресурсу: https://www.lexico.com/en/definition/anomaly.

[11] Коновалов А. М. Технології штучного інтелекту [Електронний ресурс] – Режим доступу до ресурсу: https://docs.google.com/presentation/d/1oTwhE7H7-DQi9Pm988nDOES-DBp9I8_L.

ДОДАТОК А

Розрахунок CUSUM

XAve	Upper Cusum	C+	C-	Lower Cusum	N+	N-	Target	Stdev
0,777778	0,483	0,050	0,000	-0,483	1	0	0,667046	0,120873
0,666667	0,483	0,000	0,000	-0,483	0	0	0,667046	K
0,666667	0,483	0,000	0,000	-0,483	0	0	0,667046	0,060436
0,625	0,483	0,000	0,000	-0,483	0	0	0,667046	
0,6	0,483	0,000	-0,007	-0,483	0	1	0,667046	
0,6	0,483	0,000	-0,013	-0,483	0	2	0,667046	
0,6	0,483	0,000	-0,020	-0,483	0	3	0,667046	
0,6	0,483	0,000	-0,026	-0,483	0	4	0,667046	
0,6	0,483	0,000	-0,033	-0,483	0	5	0,667046	
0,6	0,483	0,000	-0,040	-0,483	0	6	0,667046	
0,6	0,483	0,000	-0,046	-0,483	0	7	0,667046	
0,6	0,483	0,000	-0,053	-0,483	0	8	0,667046	
0,6	0,483	0,000	-0,059	-0,483	0	9	0,667046	
0,6	0,483	0,000	-0,066	-0,483	0	10	0,667046	
0,6	0,483	0,000	-0,073	-0,483	0	11	0,667046	
0,6	0,483	0,000	-0,079	-0,483	0	12	0,667046	
0,6	0,483	0,000	-0,086	-0,483	0	13	0,667046	
0,6	0,483	0,000	-0,093	-0,483	0	14	0,667046	
0,6	0,483	0,000	-0,099	-0,483	0	15	0,667046	
1	0,483	0,273	0,000	-0,483	1	0	0,667046	
0,933333	0,483	0,478	0,000	-0,483	2	0	0,667046	
0,866667	0,483	0,618	0,000	-0,483	3	0	0,667046	
1	0,483	0,890	0,000	-0,483	4	0	0,667046	
1	0,483	1,163	0,000	-0,483	5	0	0,667046	
0,8	0,483	1,235	0,000	-0,483	6	0	0,667046	
0,714286	0,483	1,222	0,000	-0,483	7	0	0,667046	
0,6	0,483	1,094	-0,007	-0,483	8	1	0,667046	
0,625	0,483	0,992	0,000	-0,483	9	0	0,667046	
0,6	0,483	0,864	-0,007	-0,483	10	1	0,667046	
0,6	0,483	0,737	-0,013	-0,483	11	2	0,667046	
0,6	0,483	0,609	-0,020	-0,483	12	3	0,667046	
0,625	0,483	0,507	-0,001	-0,483	13	4	0,667046	
0,6	0,483	0,380	-0,008	-0,483	14	5	0,667046	
0,6	0,483	0,252	-0,015	-0,483	15	6	0,667046	
0,636364	0,483	0,161	0,000	-0,483	16	0	0,667046	
0,636364	0,483	0,070	0,000	-0,483	17	0	0,667046	
0,636364	0,483	0,000	0,000	-0,483	0	0	0,667046	
0,636364	0,483	0,000	0,000	-0,483	0	0	0,667046	
0,636364	0,483	0,000	0,000	-0,483	0	0	0,667046	
0,666667	0,483	0,000	0,000	-0,483	0	0	0,667046	
0,6	0,483	0,000	-0,007	-0,483	0	1	0,667046	

ДОДАТОК Б

Розрахунок Moving Average

Data1	Data1	UCL	+2 Sigma	+1 Sigma	Average	-1 Sigma	-2 Sigma	LCL
0,777778	0,778	1,030	0,909	0,788	0,667	0,546	0,425	0,304
0,666667	0,722	0,923	0,838	0,753	0,667	0,582	0,496	0,411
0,666667	0,704	0,876	0,807	0,737	0,667	0,597	0,527	0,458
0,625	0,653	0,876	0,807	0,737	0,667	0,597	0,527	0,458
0,6	0,631	0,876	0,807	0,737	0,667	0,597	0,527	0,458
0,6	0,608	0,876	0,807	0,737	0,667	0,597	0,527	0,458
0,6	0,600	0,876	0,807	0,737	0,667	0,597	0,527	0,458
0,6	0,600	0,876	0,807	0,737	0,667	0,597	0,527	0,458
0,6	0,600	0,876	0,807	0,737	0,667	0,597	0,527	0,458
0,6	0,600	0,876	0,807	0,737	0,667	0,597	0,527	0,458
0,6	0,600	0,876	0,807	0,737	0,667	0,597	0,527	0,458
0,6	0,600	0,876	0,807	0,737	0,667	0,597	0,527	0,458
0,6	0,600	0,876	0,807	0,737	0,667	0,597	0,527	0,458
0,6	0,600	0,876	0,807	0,737	0,667	0,597	0,527	0,458
0,6	0,600	0,876	0,807	0,737	0,667	0,597	0,527	0,458
0,6	0,600	0,876	0,807	0,737	0,667	0,597	0,527	0,458
0,6	0,600	0,876	0,807	0,737	0,667	0,597	0,527	0,458
0,6	0,600	0,876	0,807	0,737	0,667	0,597	0,527	0,458
0,6	0,600	0,876	0,807	0,737	0,667	0,597	0,527	0,458
1	0,733	0,876	0,807	0,737	0,667	0,597	0,527	0,458
0,933333	0,844	0,876	0,807	0,737	0,667	0,597	0,527	0,458
0,866667	0,933	0,876	0,807	0,737	0,667	0,597	0,527	0,458
1	0,933	0,876	0,807	0,737	0,667	0,597	0,527	0,458
1	0,956	0,876	0,807	0,737	0,667	0,597	0,527	0,458
0,8	0,933	0,876	0,807	0,737	0,667	0,597	0,527	0,458
0,714286	0,838	0,876	0,807	0,737	0,667	0,597	0,527	0,458
0,6	0,705	0,876	0,807	0,737	0,667	0,597	0,527	0,458
0,625	0,646	0,876	0,807	0,737	0,667	0,597	0,527	0,458
0,6	0,608	0,876	0,807	0,737	0,667	0,597	0,527	0,458
0,6	0,608	0,876	0,807	0,737	0,667	0,597	0,527	0,458
0,6	0,600	0,876	0,807	0,737	0,667	0,597	0,527	0,458
0,625	0,608	0,876	0,807	0,737	0,667	0,597	0,527	0,458
0,6	0,608	0,876	0,807	0,737	0,667	0,597	0,527	0,458
0,6	0,608	0,876	0,807	0,737	0,667	0,597	0,527	0,458
0,636364	0,612	0,876	0,807	0,737	0,667	0,597	0,527	0,458
0,636364	0,624	0,876	0,807	0,737	0,667	0,597	0,527	0,458
0,636364	0,636	0,876	0,807	0,737	0,667	0,597	0,527	0,458
0,636364	0,636	0,876	0,807	0,737	0,667	0,597	0,527	0,458
0,636364	0,636	0,876	0,807	0,737	0,667	0,597	0,527	0,458
0,666667	0,646	0,876	0,807	0,737	0,667	0,597	0,527	0,458
0,6	0,634	0,876	0,807	0,737	0,667	0,597	0,527	0,458

ДОДАТОК В

Середній час виявлення аномалії

Дані щодо перших десяти дослідів, с

Початок атаки	Кінець атаки	Час виявлення аномалії		Час виявлення аномалії після початку атаки	
		CUSUM	MA	CUSUM	MA
19,9	25	21	21,4	1,1	1,5
20,8	26,4	22,3	22,4	1,5	1,6
17,6	23,2	19,1	19,4	1,5	1,8
20,3	25,1	22,2	22,3	1,9	2
20,8	24,6	22,3	22,5	1,5	1,7
19,4	23,7	21,6	21,7	2,2	2,3
21,3	25,5	22,6	23	1,3	1,7
21	27	23,1	23,3	2,1	2,3
19,1	26,2	22	22	2,9	2,9
21,3	23,4	22,2	22,5	0,9	1,2

Середній час розраховується як середнє значення часу виявлення аномалії після початку атаки в перших десяти тестах:

- CUSUM — 1,7 с;
- Moving Average — 1,9 с.

Дані щодо наступних десяти дослідів, с

Початок атаки	Кінець атаки	Час виявлення аномалії		Час виявлення аномалії після початку атаки	
		CUSUM	MA	CUSUM	MA
19,8	25,3	21,1	21,3	19,8	25,3
20,1	26,2	22,9	23,1	20,1	26,2
18,5	25,6	21	21	18,5	25,6
19	25	20,6	20,6	19	25
23,3	29,5	26	26	23,3	29,5
16,8	23,5	19,2	19,1	16,8	23,5
21,2	26,4	23,4	23,6	21,2	26,4
18,7	27,3	21,1	21,1	18,7	27,3
23,3	27,5	25,5	25,3	23,3	27,5
20	24	22,8	22,6	20	24

Середній час розраховується як середнє значення часу виявлення аномалії після початку атаки в наступних десяти тестах:

- CUSUM — 2,3 с;
- Moving Average — 2,3 с.

ДОДАТОК Г

Зразки тренувального набору

	Тестові дані
Timestamps	Value
6/6/2022 0:06:41	20
6/6/2022 0:06:42	19
6/6/2022 0:06:43	20
6/6/2022 0:06:44	19
6/6/2022 0:06:45	20
6/6/2022 0:06:46	19
6/6/2022 0:06:47	20
6/6/2022 0:06:48	20
6/6/2022 0:06:49	19
6/6/2022 0:06:50	20
6/6/2022 0:06:51	19
6/6/2022 0:06:52	20
6/6/2022 0:06:53	19
6/6/2022 0:06:54	24
6/6/2022 0:06:55	19
6/6/2022 0:06:56	20
6/6/2022 0:06:57	19
6/6/2022 0:06:58	32
6/6/2022 0:06:59	61
6/6/2022 0:07:00	65
6/6/2022 0:07:01	65
6/6/2022 0:07:02	61
6/6/2022 0:07:03	65
6/6/2022 0:07:04	97
6/6/2022 0:07:05	107
6/6/2022 0:07:06	109
6/6/2022 0:07:07	110
6/6/2022 0:07:08	103
6/6/2022 0:07:09	110
6/6/2022 0:07:10	109
6/6/2022 0:07:11	110
6/6/2022 0:07:12	106
6/6/2022 0:07:13	107
6/6/2022 0:07:14	110
6/6/2022 0:07:15	109
6/6/2022 0:07:16	110
6/6/2022 0:07:17	109
6/6/2022 0:07:18	104
6/6/2022 0:07:19	106
6/6/2022 0:07:20	65

ДОДАТОК Ґ
Зразки тестового набору

Тренувальні дані	
Timestamps	Value
5/5/2022 0:06:41	77
5/5/2022 0:06:42	79
5/5/2022 0:06:43	51
5/5/2022 0:06:44	20
5/5/2022 0:06:45	19
5/5/2022 0:06:46	20
5/5/2022 0:06:47	19
5/5/2022 0:06:48	20
5/5/2022 0:06:49	19
5/5/2022 0:06:50	20
5/5/2022 0:06:51	19
5/5/2022 0:06:52	20
5/5/2022 0:06:53	19
5/5/2022 0:06:54	20
5/5/2022 0:06:55	19
5/5/2022 0:06:56	20
5/5/2022 0:06:57	19
5/5/2022 0:06:58	20
5/5/2022 0:06:59	19
5/5/2022 0:07:00	20
5/5/2022 0:07:01	19
5/5/2022 0:07:02	19
5/5/2022 0:07:03	50
5/5/2022 0:07:04	79
5/5/2022 0:07:05	77
5/5/2022 0:07:06	78
5/5/2022 0:07:07	76
5/5/2022 0:07:08	78
5/5/2022 0:07:09	77
5/5/2022 0:07:10	78
5/5/2022 0:07:11	77
5/5/2022 0:07:12	78
5/5/2022 0:07:13	78
5/5/2022 0:07:14	78
5/5/2022 0:07:15	78
5/5/2022 0:07:16	78
5/5/2022 0:07:17	77
5/5/2022 0:07:18	79
5/5/2022 0:07:19	77
5/5/2022 0:07:20	78