ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ЗАХИСТУ ІНФОРМАЦІЇ

КАФЕДРА ІНФОРМАЦІЙНОЇ ТА КІБЕРНЕТИЧНОЇ БЕЗПЕКИ

Пояснювальна записка
до магістерської роботи
на тему:

«ТЕХНОЛОГІЇ ПІДВИЩЕННЯ БЕЗПЕКИ БЕЗПРОВОДОВИХ СЕНСОРНИХ МЕРЕЖ З ВИКОРИСТАННЯМ RASPBERRY PI»

Виконав: студент 6 курсу, групи БСДМ-62
Спеціальності 125 Кібербезпека
Освітньо-професійної програми «Інформаційна та кібернетична безпека»
(шифр і назва спеціальності)

Коваль В.Р.
(прізвище та ініціали)

Керівник Довженко Н.М.
(прізвище та ініціали)

Рецензент
(прізвище та ініціали)

Нормоконтролер Чумак Н.С.
(прізвище та ініціали)

КИЇВ – 2021

РЕФЕРАТ

Текстова частина магістерської роботи: 83 сторінок, 46 рисунків, 2 таблиці, 30 джерел.

Об'єкт дослідження – процес забезпечення безпеки БСМ.

Предмет дослідження – технології підвищення безпеки БСМ з використанням Raspberry Pi.

Мета роботи – підвищення безпеки та ефективності функціонування БСМ з використанням Raspberry Pi як компонента мережі.

Методи дослідження – опрацювання літератури за даною темою, теорія інформації, практичне тестування програмного забезпечення.

В роботі проаналізовано безпроводові сенсорні мережі. Представлено узагальнену таксономію атак на БСМ та приведено аналіз різних видів атак способи боротьби з ними. Проведено дослідження вразливості сенсорних датчиків та «розумних» пристроїв за допомогою пошукової системи Shodan.

Описано Raspberry Pi, операційну систему Raspbian та мову програмування Python, яка використовується сьогодні в БСМ. Підкреслено, що Raspberry Pi дає можливість не лише розширити діапазон інтелектуальної складової при використанні сенсорних датчиків, але і додати безпеку, як повноцінну складову безпроводових сенсорних мереж. Зокрема шляхом відстеження мережевого трафіку з можливістю націлення на конкретні MAC-адреси. Фільтрація MAC-адрес на Raspberry Pi обробляється кількома рядками коду і дає можливості уникнути вручань в безпроводову сенсорну мережу з боку зловмисників.

Галузь використання – кібербезпека.

БЕЗПРОВОДОВА СЕНСОРНА МЕРЕЖА, ДАТЧИК, ІОТ, БЕЗПЕКА, КОНФІДЕНЦІЙНІСТЬ, RASPBERRY PI, ОЦІНКА РИЗИКІВ, ЗАГРОЗИ, УРАЗЛИВІ МІСЦЯ, ІНФОРМАЦІЯ, MAC, АДРЕСА, СЕРВІС, КІБЕРАТАКА, МЕТОД.

ABSTRACT

Master's thesis: 83 pages, 46 figures, 2 tables, 30 sources.

Object of research – the process of ensuring the safety of WSN.

Subject of research – the security enhancement technologies of WSN using Raspberry Pi.

The aim of research – to increase the security and efficiency of WSN using the Raspberry Pi as a network component.

Research methods – elaboration of literature on the topic, information theory, practical software testing.

The paper describes the wireless sensor networks. The generalized taxonomy of attacks on WSN is presented at work. The analysis of different types of attacks and ways of struggle against them has also resulted. A study of the vulnerability of sensors and smart devices using the Shodan search is presented.

As the title implies the work describes Raspberry Pi, Raspbian operating system, and Python programming language which are used today in WSN. It is emphasized that the Raspberry Pi allows not only to expand of the range of intelligent components when using sensors but also adds security as a full-fledged component of wireless sensor networks. In particular, by tracking network traffic with the ability to target specific MAC addresses. MAC address filtering on the Raspberry Pi is handled by a few lines of code and makes it possible to avoid handing over to the wireless sensor network by attackers.

Field of use – cybersecurity.

WIRELESS SENSOR NETWORK, SENSOR, INTERNET OF THINGS, SECURITY, PRIVACY, RASPBERRY PI, RISK ASSESSMENT, THREATS, VULNERABILITIES, INFORMATION, MAC, ADDRESS, SERVICE, CYBERATRACK, METHOD.

ЗМІСТ

ВСТУП

Актуальність дослідження. Безпроводові сенсорні мережі обговорюються вже більше 30 років, але конкретизоване бачення БСМ почало втілюватися в реальність лише завдяки останнім досягненням безпроводового зв'язку та електроніки, які дозволили розробити недорогі, малопотужні та багатофункціональні датчики невеликого розміру, які здатні передавати інформацію на короткі відстані.

Сьогодні дешеві «розумні» датчики, об'єднані в мережу за допомогою безпроводових каналів і розгорнуті у великій кількості, надають безпрецедентні можливості для моніторингу та контролю за будинками, містами та навколишнім середовищем. Крім того, такі датчики мають широкий спектр застосування в оборонній сфері, створюючи нові можливості для розвідки та спостереження, а також інших тактичних застосувань.

Безпроводові сенсорні мережі (БСМ), Інтернет Речей (IoT) та промислові системи управління (ICS) вимагають особливої уваги з точки зору кібербезпеки. Це ґрунтується на відомому і документально підтвердженому факті, що протоколи та реалізації мають вразливі місця, які при експлуатації можуть завдати значної шкоди та забезпечити шлях для вилучення даних.

Крім того, при дослідженні цих середовищ через динамічний характер та/або критичні наслідки для інфраструктури активне сканування або зондування цих середовищ або не рекомендується, або є неефективним. Таким чином, пасивний моніторинг цих середовищ дає уявлення про поведінку цих пристроїв та мереж, у яких вони працюють. Однією з основних проблем є розміщення пристроїв моніторингу для забезпечення видимості та покриття як з проводової, так і з безпроводової точки зору. Сьогодні пропонуються рішення постачальників, які спираються на дорогі апаратні та програмні рішення, яким може не вистачати гнучкості.

Будь-яке радіообладнання може брати участь, пасивно або активно, у будь-якій формі радіозв’язку, налаштовуючи обладнання на ту саму смугу частот, що й вузли зв’язку.

З впровадженням технології мініатюризації попит на БСМ зростає ще швидше. Отже, з’явилося масове виробництво сенсорних вузлів із посереднім апаратним забезпеченням, яке зазвичай не захищене від втручання. Отже, вузли можуть бути легко скомпрометовані зловмисниками, що призводить до абсолютно нового набору атак.

Рішення впроваджвати для широкого використання Raspberry Pi із програмним забезпеченням Python з відкритим вихідним кодом для пасивного моніторингу та виявлення аномальної поведінки, спочатку, викликало сумніви. Однак, Raspberry Pi з її багатоядерним процесором та інтегрованими проводовими та безпроводовими мережевими компонентами забезпечує базову основу, необхідну для впровадження як окремих БСМ, так і датчиків IoT/ICS. Поєднання програмних рішень Python з відкритим вихідним кодом, певну кількість сенсорів, безпроводове середовище, і ось виходить недорогий, гнучкий та спритний PiSensor для середовищ IoT та ICS.

Об’єкт дослідження – процес забезпечення безпеки БСМ.

Предмет дослідження – технології підвищення безпеки БСМ з використанням Raspberry Pi.

Мета роботи – підвищення безпеки та ефективності функціонування БСМ з використанням Raspberry Pi як компонента мережі.

Відповідно до мети наукового дослідження були поставлені та розв’язані наступні завдання:

- проаналізовано безпроводові сенсорні мережі;
- представлено узагальнену таксономію атак на БСМ;
- описано Raspberry Pi, операційну систему Raspbian та мову програмування Python, яка використовується сьогодні в БСМ;
- представлено детальний огляд елементів коду, який використовується для здійснення управління Raspberry Pi;

- визначено, що можливість моніторингу, запису та активації датчика Raspberry Pi для націлення на певні MAC-адреси є важливою в межах промислового контролю або розділених середовищ БСМ чи Інтернету речей;
- визначено, що фільтрація MAC-адрес на Raspberry Pi обробляється лише кількома рядками коду і дає можливості уникнути вручань в безпроводову сенсорну мережу з боку зловмисників.

Результати, досягнуті в процесі роботи дозволяють зробити висновок, що впровадження в безпроводову сенсорну мережу Raspberry Pi як окремого датчика дає можливість не лише розширити діапазон інтелектуальної складової при використанні сенсорних датчиків, але і додати безпеку, як повноцінну складову мережі.

Методи дослідження – опрацювання літератури за даною темою, теорія інформації, практичне тестування програмного забезпечення.

1 АНАЛІЗ АКТУАЛЬНИХ ВРАЗЛИВОСТЕЙ БСМ ТА ШЛЯХИ ЇХ НЕЙТРАЛІЗАЦІЇ

1.1. Особливості розгортання архітектури БСМ

Сенсорна мережа, як правило, складається з набору недорогих, малопотужних сенсорних вузлів, розподілених певним, тимчасовим способом. Кожна з цих мереж має один або кілька вузлів-приймачів або базових станцій, які зазвичай представлені комп'ютером із значними ресурсами з достатніми можливостями зберігання, обчислення та зв'язку (рис.1.1).

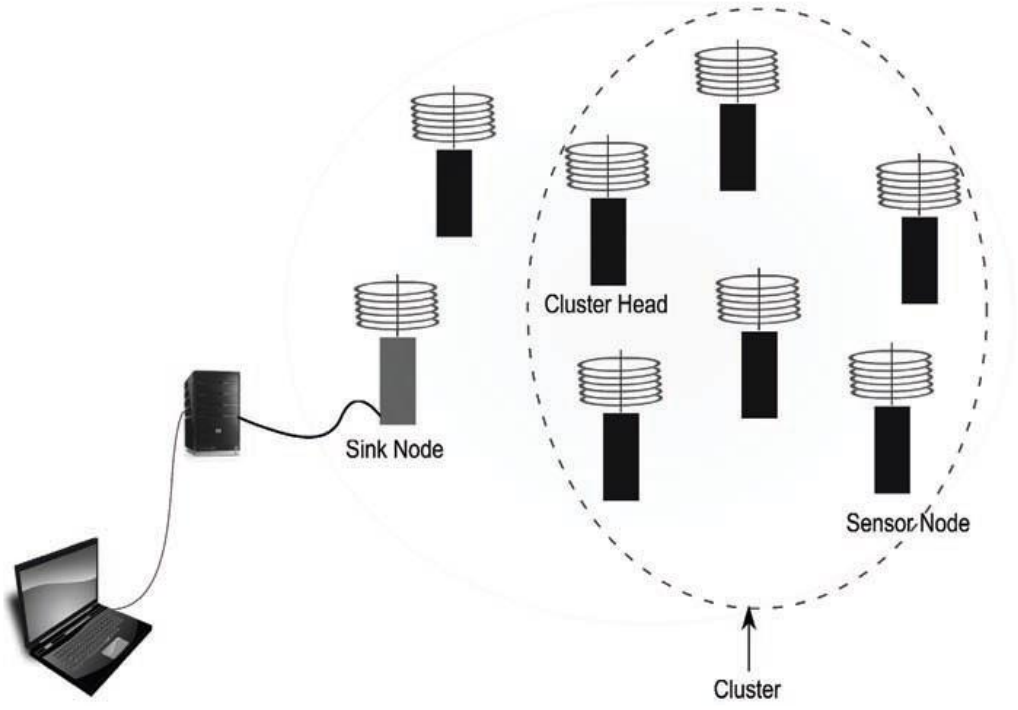

Рис.1.1. Безпроводові сенсорні мережі

Ці вузли-приймачі надсилають запити або команди своїм відповідним сенсорним вузлам, які потім співпрацюють один з одним для виконання заданого завдання. Крім того, вузли-приймачі діють як шлюз між сенсорною мережею і зовнішньою мережею (Інтернет або стільникова мережа). Типовий сенсорний вузол оснащений власне сенсорним обладнанням для агрегації даних, вбудованим мікроконтролером для миттєвої обробки даних і радіоприймачем для цифрового

зв'язку. Зазвичай вузли спілкуються на невеликій відстані і використовують безпроводові засоби для комунікації. Кожен сенсорний вузол сприймає дані з навколишнього середовища і відправляє ці отримані дані до свого вузла-приймача [1].

Як правило, дані або керуючі пакети від вузла-відправника проходять поетапно через кожен проміжний сенсорний вузол, щоб досягти вузла-приймача (рис.1.2).

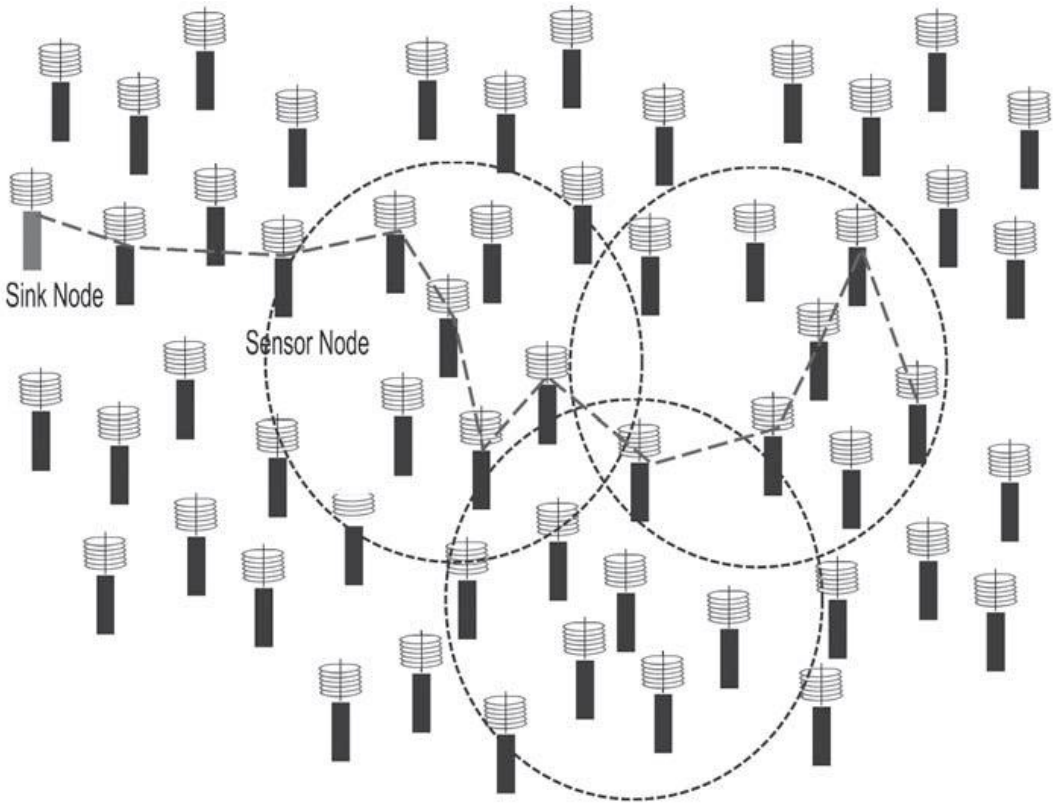

Рис.1.2. Поетапна маршрутизація

Однак у певних ситуаціях для кожного вузла неефективно пересилати захоплені вихідні дані безпосередньо на вузол-приймач, оскільки це може призвести до раннього вичерпання цінних ресурсів (батареї, пам’яті та обчислювальної здатності) кожного проміжного вузла маршруту. Таким чином, щоб подолати цю проблему, мережа додатково поділяється на кластери, і кожен кластер має голову кластера. Кожна голова кластера відповідає за локальне агрегування даних зі своїх вузлів кластера, а потім пересилає їх до свого вузла-

приймача. Крім того, голова кластеру може попередньо обробити вихідні дані перед відправкою їх на вузол-приймач.

Залежно від того, як дані агрегуються сенсорними вузлами, архітектури БСМ можна класифікувати на однорідні, гетерогенні та гібридні сенсорні мережі [2].

Однорідні сенсорні мережі. У однорідній мережевій архітектурі голови кластерів і звичайні сенсорні вузли мають такі ж можливості обчислення, зберігання та зв'язку, що й вузол-приймач. У цій архітектурі топологія мережі визначає метод агрегації даних. Зазвичай плоска та ієрархічна архітектури (рис. 1.3 та 1.4) використовуються в однорідній архітектурі [3].

Рис.1.3. Плоска архітектура в БСМ

Плоска архітектура. У плоскій архітектурі кожен об'єкт мережі (приймальний вузол і вузли датчиків) має однакові обчислювальні та комунікаційні можливості. Отже, його також можна сприймати як однорангову архітектуру. Однак із збільшенням масштабованості сенсорних вузлів досить складно мати глобальну схему адресації. З цієї причини схема маршрутизації, орієнтована на дані, шукається як прагматичне рішення проблеми збільшення масштабованості. У цій схемі вузол-приймач передає запит або команду всім вузлам в мережі, і лише відповідні вузли, які відповідають запиту, відповідають на вузол-приймач.

Ієрархічна архітектура. На відміну від плоскої архітектури, ієрархічна архітектура розділяє мережу на кластери і підвищує загальну енергоефективність мережі [4]. У цій архітектурі вузли працюють у безпосередній близькості від відповідних голів кластерів. Таким чином, вузли з нижчими рівнями енергії просто захоплюють необхідні вихідні дані та пересилають їх у відповідні голови кластерів.

Зазвичай голови кластерів володіють більшою ємністю обробки та зберігання, ніж будь-який звичайний сенсорний вузол.

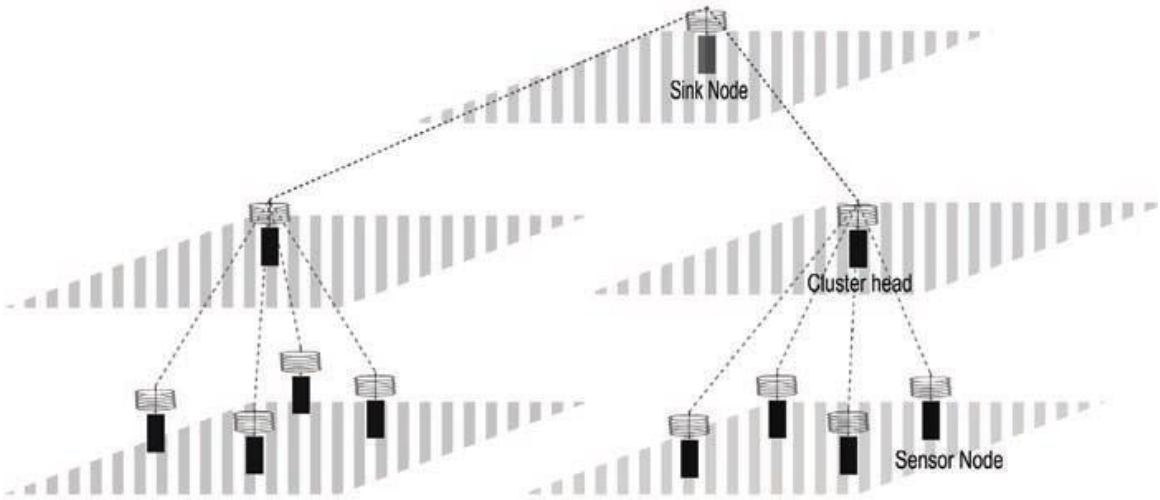

Рис.1.4. Ієрархічна архітектура в БСМ

У випадку високодинамічного та децентралізованого середовища вузли з вищими енергетичними рівнями можуть взяти на себе відповідальність за голову кластера, а вузли з нижчими енергетичними рівнями можуть бути вузлами кластера. Крім того, застосування цієї архітектури не тільки підвищить енергоефективність мережі, але й збалансує навантаження на трафік із збільшенням розміру мережі. У певних ситуаціях агрегація даних може виконуватися в головах кластера замість вузла-приймача, щоб підвищити ефективність усієї мережі.

Неоднорідні сенсорні мережі. У гетерогенній сенсорній мережі вузол-приймач або базова станція можуть бути мобільними [5]. Таким чином, мобільний вузол-приймач може випадковим чином переміщатися в будь-яку з зон сприйняття та збирати дані, тісно взаємодіючи з сенсорними вузлами. Крім того, мобільність кластера підвищить загальний рівень енергії системи.

1.2. Огляд поширених атак на БСМ

Безпроводові сенсорні мережі загалом є підкласом безпроводових мереж, тому більшість видів атак схожі із атаками на безпроводові канали та середовище в цілому.

Однак через додаткові проблеми, БСМ створили передумови до нових варіацій атак, які можна класифікувати на «атаки дрібного класу» та «атаки класу ноутбуків» [6].

Під час атаки класу «атаки дрібного класу» зловмисник отримує контроль над кількома скомпрометованими вузлами і має потужність не більше, ніж у цих звичайних сенсорних вузлів. Отже, він повинен запускати атаки, не вичерпуючи свої ресурси (сховище, обчислення та пропускна здатність). Крім того, зловмисник обмежений у охопленні через обмежену потужність передачі. Навпаки, зловмисник «класу ноутбуків» має більше ресурсів (висока потужність передачі, тривалий термін служби батареї, високошвидкісні процесори та високочутливі всеспрямовані антени). Таким чином, має більший охоплення та різноманітність щодо атак, які він може здійснити. На рис.1.5. представлена таксономія атак для БСМ, а в табл.1.1 показана класифікація атак за допомогою багаторівневого підходу.

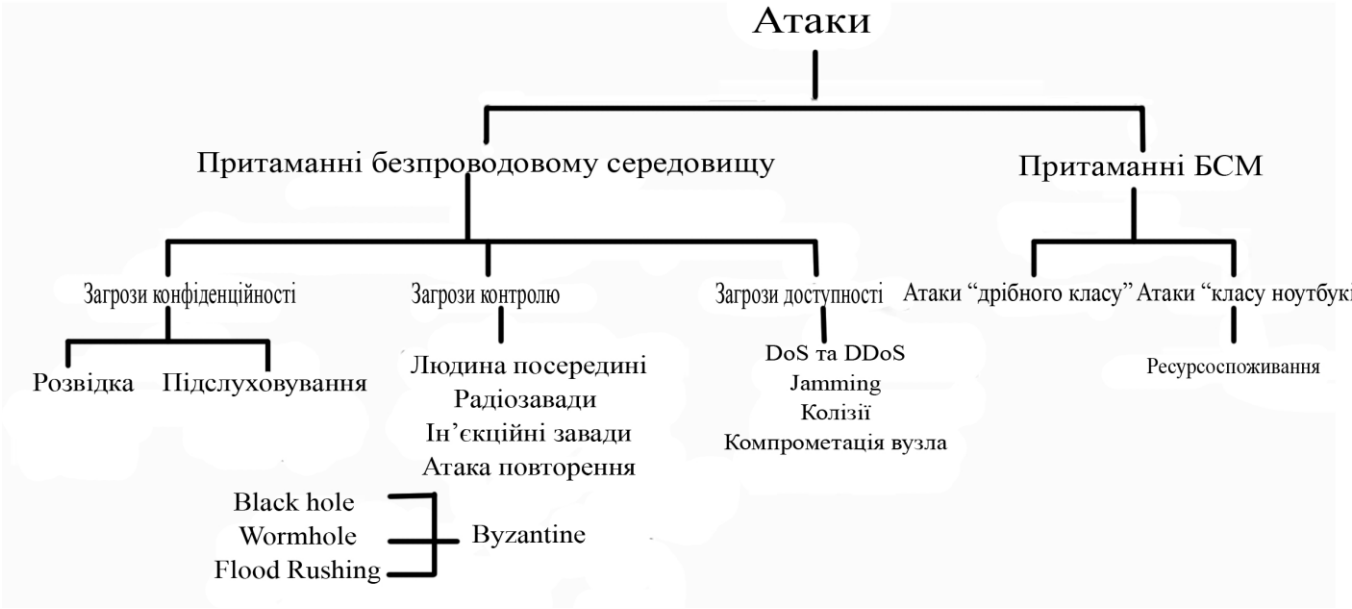

Рис.1.5. Таксономія атак для БСМ

Загалом атаки можна розділити на активні та пасивні.

Пасивна атака. У цьому типі атаки зловмисник може перехоплювати та відстежувати дані між вузлами, що спілкуються, але не підробляє та не модифікує пакети, побоюючись викликати підозру щодо шкідливої діяльності серед вузлів.

Наприклад, під час аналізу трафіку зловмисник може не мати змоги декодувати зашифровані дані, але можна знайти корисну інформацію, аналізуючи заголовки пакетів, їх розміри та частоту передачі. У БСМ також можна виконувати розвідку, щоб зрозуміти обмін інформацією між вузлами, що спілкуються, особливо в точках агрегації даних. Крім того, можна використовувати інформацію про маршрути за допомогою аналізу трафіку.

Активна атака. В цьому типі атаки зловмисник бере активну участь у всіх формах зв'язку (контроль і дані) і може видаляти, змінювати порядок та відтворювати повідомлення або навіть надсилати підроблені повідомлення вузлам мережі. Деякі інші активні атаки включають захоплення вузлів, підробку інформації про маршрутизацію, і атаки виснаження ресурсів.

1.2.1. Загрози конфіденційності

У БСМ загрози конфіденційності можна далі класифікувати на розвідку та підслуховування.

Таблиця 1.1.

Класифікація атак у відповідність рівням реалізації

Рівень	Вектор атаки
Прикладний	Пошкодження даних, переривання та відміна програм
Транспортний	Захоплення сеансу та SYN-flood
IP рівень	Атака Byzantine, ресурсоспоживання, wormhole та black hole, розкриття розташування
Канальний	Аналіз трафіку
Фізичний	Фізичні завади та перешкоди
Багаторівнева атака	DoS, відтворення та людина посередині

1) Розвідка. Це інтелектуальний збір або зондування для доступу до вразливостей мережі, щоб пізніше запустити повномасштабну атаку. Розвідувальні атаки можна розділити на активні та пасивні. Пасивні розвідувальні атаки

включають збір мережевої інформації непрямими або прямими методами, але без зондування цілі; активні розвідувальні атаки включають процес збору трафіку з наміром отримати відповідь від цілі.

2) Підслуховування – це акт таємного прослуховування приватної розмови. Однак у парадигмі БСМ підслуховування — це операція з вивчення «сукупних даних», які збирає вся мережа. Таким чином, підслуховування між двома конкретними сенсорними вузлами може не допомогти зловмиснику досконало зрозуміти всю мережу. Його можна розділити на активне та пасивне прослуховування.

– Активне підслуховування – у цьому випадку зловмисник активно надсилає запити іншим вузлам, намагаючись спонукати їх відповісти на його запити, і в обмін зможе зрозуміти точне завдання, призначене вузлам у мережі. Зазвичай зловмисник запускає атаку «людина посередині», щоб проникнути в мережу та закріпити себе у маршруті;

– Пасивне підслуховування. Зловмисник підключається в активний маршрут, не знаючи інших вузлів мережі. Потім він пасивно прослуховує весь трафік, надісланий через канали зв’язку. Може бути важко виявити пасивну атаку підслуховування, оскільки зловмисник може діяти в режимі прихованості.

1.2.2. Загрози для контролю

1) Атака «Людина посередині» є однією з класичних, яка може бути виконана в БСМ. У цьому типі атаки зловмисник проникає в мережу і намагається встановити незалежне з’єднання між вузлом та вузлом-приймачем. Вузли в мережі не знають, що весь контроль за трафіком обробляється зловмисником. Він може перебувати як в пасивному, так і в активному стані. У пасивному стані він просто передає кожне повідомлення між вузлами з наміром здійснити атаку підслуховування. У активному стані він може змінювати перехоплені дані, намагаючись зламати аутентифікацію. Атака може бути здійснена на фізичному канальному, мережевому та прикладному рівнях [6].

2) Радіоперешкоди. Зі збільшенням кількості безпроводових технологій, що використовують ту саму смугу відкритого спектру (2,4 ГГц, 5 ГГц або 900 МГц), неминуче виникнуть радіоперешкоди. Наприклад, у щільному міському середовищі, де безпроводові телефони мають однаковий спектр, відбувається різке погіршення продуктивності окремих вузлів через радіоперешкоди. Подібні проблеми можна передбачити для сенсорних мереж із збільшенням кількості сенсорних вузлів на мережу. Результат такої перешкоди може призвести до зміни бітів інформації, що передаються і це робить біти нерозбірливими і в кінцевому підсумку їх відкидає приймач [7]. Отже, радіоперешкоди можуть призвести до атаки відмови в обслуговуванні. Найгірший сценарій радіоперешкод – це глушіння.

3) Ін'єкційна атака. Після того, як зловмисник таємно проник в мережу БСМ, він може видавати себе за кілька сенсорних вузлів (або навіть вузлів-приймачів) і може впровадити шкідливі дані в мережу. Шкідливі дані можуть бути фальшивою рекламою інформації про сусідні вузли іншим вузлам, що призведе до видавання себе за приймальні вузли та агрегації всіх даних.

4) Повторна атака. Атака відтворення — це поширена атака в БСМ, за допомогою якої зловмисник може перехопити дані користувача та повторно передати дані користувача пізніше. Ця атака особливо корисна для зламу слабких схем аутентифікації, які не враховують позначку часу під час аутентифікації вузлів. Ця атака також корисна під час процесів розповсюдження спільного ключа.

5) Byzantine. Під час візантійської атаки зовнішній зловмисник може отримати повний контроль над підмножиною аутентифікованих вузлів, які можна використовувати для атаки на мережу зсередини. Такі атаки зловмисної поведінки відомі як візантійські атаки. Деякі приклади атаки – це black hole, wormhole та flood rushing.

- Атаки black hole. В цьому типі атаки (рис.1.6) зловмисник вибірково скидає пакети або всі пакети керування та даних, які маршрутизуються через нього. Таким чином, будь-який пакет, маршрутизований через цей проміжний шкідливий вузол, буде страждати від часткової або повної втрати даних.

- Flood rushing атака. Цей тип атаки (рис.1.7) є загальним для безпроводових мереж. У цій атаці зловмисник намагається порушити існуючий шлях маршрутизації, посилаючи потоки пакетів через альтернативний маршрут, що призведе до відкидання перевіреного маршруту і прийняття шкідливого маршруту. Звичайні схеми аутентифікації не можуть запобігти цій атаці, оскільки зловмисники є аутентифікованими вузлами.

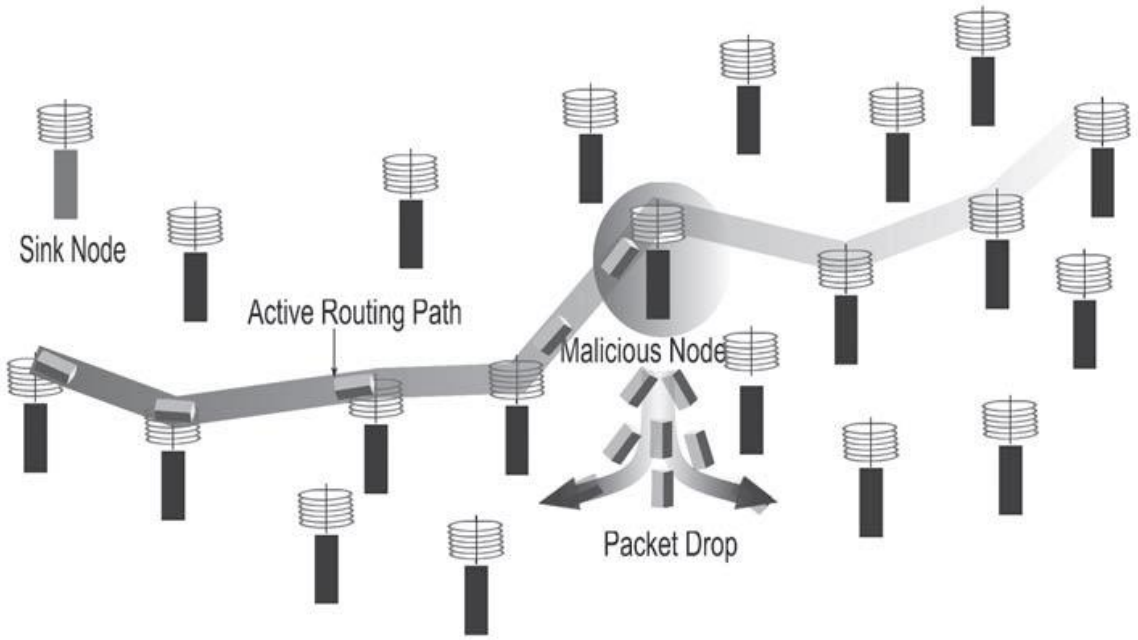

Рис.1.6. Атака black hole

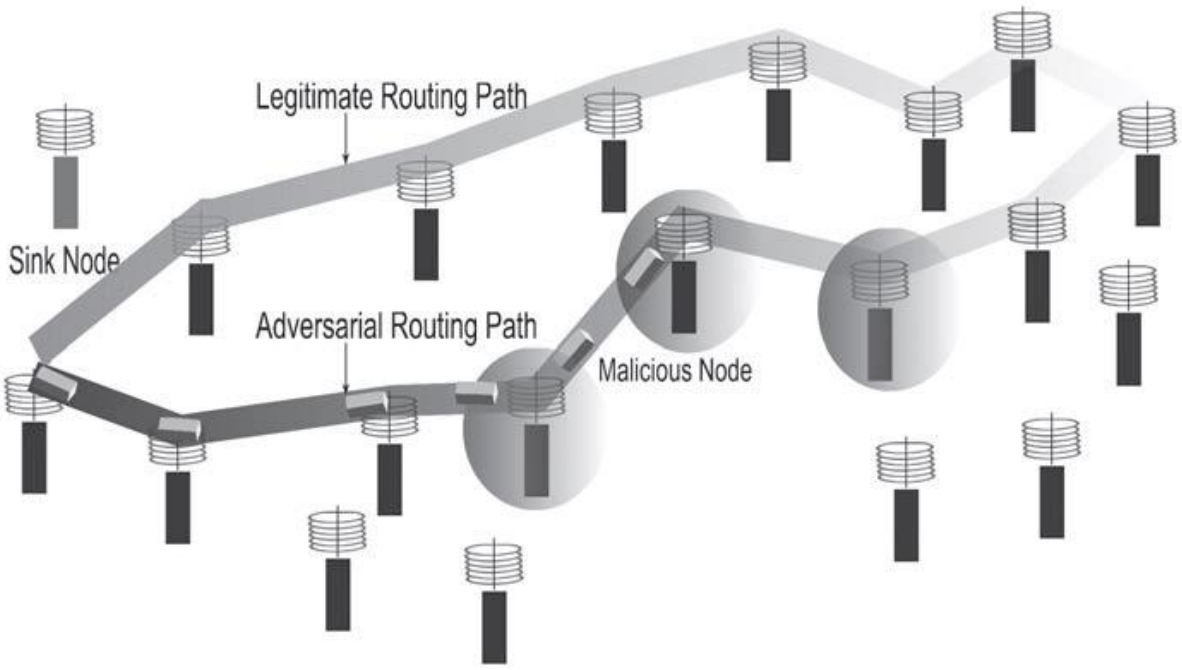

Рис.1.7. Атака Flood rushing

- Атака wormhole. В цьому типі атаки два підступних сенсорних вузла, або ноутбуки, контролюють тунель і пакети даних між собою, з метою створення ярлика в БСМ. Такий тунель із низькою затримкою між двома підступними вузлами, ймовірно, збільшить ймовірність того, що він буде обраний як активний шлях. Цей тип атаки дуже тісно пов'язаний з атакою скважин, оскільки один із підступних вузлів може помилково оголосити себе як вузол-приймач і тим самим залучити більше трафіку, ніж зазвичай. Однією з основних відмінностей Byzantine wormhole від традиційної є те, що у першої тунель існує між двома скомпрометованими вузлами, тоді як у традиційній два легітимні вузли обманом повірили, що між ними існує безпечний тунель [8].

6) Атака Sybil. Цей тип атаки становить серйозну загрозу для механізмів маршрутизації в БСМ, адже під час цієї атаки зловмисний вузол маскується під набір вузлів, заявляючи про фальшиві ідентичності або створюючи нові ідентичності в гіршому випадку. Такі атаки можна легко здійснити в середовищі БСМ, оскільки вузли завжди розгорнуті в неструктурованому та розподіленому середовищі та спілкуються за допомогою радіопередачі. Вони особливо шкідливі в таких програмах, як агрегація даних, системи голосування, оцінка репутації та географічна маршрутизація. Використовуючи атаку Sybil у маршрутизації з урахуванням місцезнаходження, вузол-зловмисник може ідентифікуватися одночасно в кількох локаціях.

1.2.3. Загрози доступності

1) Відмова в обслуговуванні (DoS або DDoS). Атака відмови в обслуговуванні відбувається, коли зловмисник надсилає жертві фіктивні або підроблені пакети з наміром знизити швидкість їх опрацювання. У гіршому випадку це робить жертву абсолютно невідповідною. Наприклад, у середовищі БСМ, де вузли мають обмежену обчислювальну потужність, DoS-атака від зловмисника, який має більшу потужність, призведе до перевантаження вузла шляхом зменшення пропускної здатності, пам'яті і потужності обробки запитів. З

точки зору зловмисника, ця атака також ефективна в мережах, де вузли зобов’язані передавати критично важливі дані. Заглушення безпроводових каналів також може призвести до DoS-атаки. Розширена DoS-атака — це розподілена DoS-атака, коли зловмисник бере контроль над кількома вузлами в мережі, що призводить до розподіленої атаки flood на жертву.

2) HELLO flood Attack. Одним із поширених методів виявлення сусідів є відправка пакетів HELLO. Якщо вузол отримує пакет HELLO, це означає, що він знаходиться в межах діапазону зв'язку. Однак зловмисник атаки «класу ноутбуку» може легко відправити пакети HELLO з достатньою потужністю, щоб переконати сенсорні вузли, що він знаходиться поблизу зв’язку і може бути потенційним сусідом. Зловмисник може видавати себе за вузол-приймач або вузол кластера.

3) Jamming. Заглушення є одним з найбільш смертоносних типів атак у БСМ і на меті має пряму ціль скомпрометувати всю безпроводову мережу. У цьому типі атаки зловмисник перекриває діапазон спектру потужним передавачем і не дає будь-якому члену мережі у враженій зоні передавати або отримувати будь-який пакет. Атаки заглушення можна розділити на постійні та часткові заглушення. Часткове може бути дуже ефективним у моменти, коли зміна одного біта кадру даних змусить приймач скинути його. Під час такого роду атаки жертві важко визначити, чи його смуга заглушена навмисно чи через завади каналу, і його негайною реакцією зазвичай є збільшення потужності передачі, тим самим виснажуючи ресурси з більшою швидкістю. Атаки спрямовані на фізичний і канальний рівні.

4) Колізія. Дані атаки націлені на канальний рівень (MAC), щоб створити дороговартісний експоненційний відхід. Всякий раз, коли відбувається колізія, вузли повинні повторно передавати пакети, які постраждали, що призводить до множинних повторних передач. Кількість енергії, витраченої зловмисником, становить набагато менше, ніж енергії, що витрачається (вичерпання батареї) вузлами датчика. Атаку колізії можна віднести до атак на вичерпання ресурсів.

5) Компрометація вузлів є однією з найпоширеніших і шкідливих атак у БСМ. Оскільки датчики можуть бути розгорнуті в нетипових умовах, вони легко

піддаються захопленню стороннім агентом. У разі такого сценарію зловмисник може витягти корисні дані (наприклад, закриті ключі в сенсорних вузлах). Крім того, його можна було перепрограмувати і запустити знову, щоб діяти від імені зловмисника [9].

1.2.4. Специфічні атаки на БСМ

1) *Атаки на протокол TinyOS Beaconing Protocol.* Протокол TinyOS Beaconing Protocol використовує алгоритм охоплюючого дерева для трансляції оновлень маршрутизації. Вузол-приймач періодично передає оновлену інформацію про маршрутизацію своїм безпосереднім сусіднім вузлам. Ці сусідні вузли потім повторно передають цю інформацію своїм безпосереднім сусідам, і процес продовжується рекурсивно. Під час цього процесу кожен проміжний вузол записує свій батьківський вузол (батьківський вузол є першим вузлом, який зміг встановити контакт зі своїм підлеглим вузлом і передати інформацію про маршрутизацію). Коли всі активні вузли працюють, вони повинні надіслати всі отримані дані на свій батьківський вузол. Однак цей протокол вразливий до багатьох атак. Наприклад, проста атака видавання себе за іншу особу, що призведе до атаки впадин, може повністю скомпрометувати всю мережу.

Аутентифікацію можна використовувати, щоб запобігти атакам імітації, але вона не заважає зловмиснику «класу ноутбука» запустити атаку вибіркового пересилання (рис.1.8.), атаку підслуховування або атаку Black Hole. Зловмисник створює Wormhole між двома зловмисниками «класу ноутбуків». Обидва ноутбуки розміщені біля зацікавленого вузла. Ноутбук біля вузла-приймача залучає весь трафік свого сусіда і просто тунелює ці аутентифіковані повідомлення своєму змовнику.

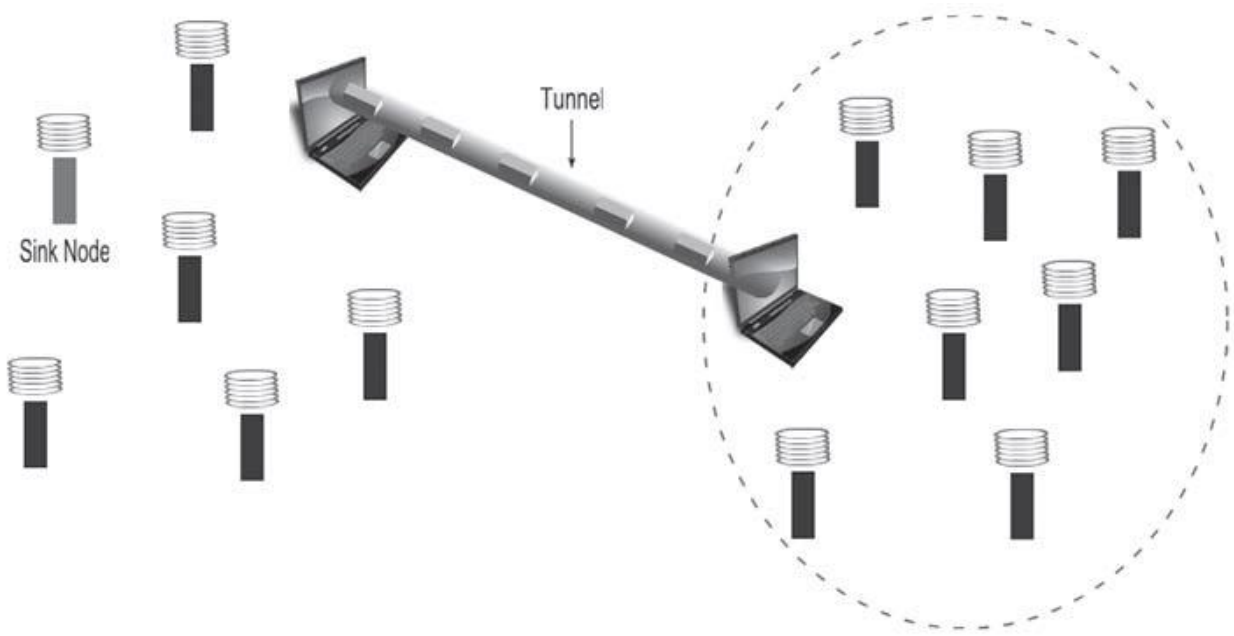

Рис.1.8. Вибіркова атака переадресації в TinyOS Beaconing

Ноутбук зловмисника, розташований поблизу вузла-приймача, відіграє пасивну роль у пересиланні повідомлень. Через прихований характер його сусідам важко виявити, чи він являється зловмисним. Після того, як аутентифіковані повідомлення потрапляють до віддаленого зловмисника, він може почати атаку через Black Hole або Wormhole.

При ситуації, коли для аутентифікації використовуються цифрові підписи, під час оновлення маршрутизації приватний ключ вузла-приймача витікає. Як тільки вузол-приймач усвідомлює, що його закритий ключ скомпрометований, він негайно передає новий відкритий ключ. Усі вузли в безпосередній близькості від вузла-приймача оновлять свою локальну копію відкритого ключа вузла-приймача. Ноутбук поруч із вузлом буде виконувати ту саму операцію та передавати цю інформацію своєму ноутбуку, який у змові. Віддалений ноутбук тепер може легко видати себе за вузол-приймач і запустити штурмову атаку. Крім того, він може додатково створювати цикли маршрутизації, що є атакою на вичерпання ресурсів.

2) Атаки на географічну та енергозалежну маршрутизацію (GEAR). GEAR пропонує рекурсивний алгоритм маршрутизації з урахуванням розташування та енергії для вирішення проблеми нерівномірного споживання енергії при маршрутизації в БСМ. У GEAR кожен вузол вимірює рівень енергії своїх сусідів

разом із відстанню від цілі, перш ніж приймати рішення про маршрутизацію. У таких ситуаціях зловмисник класу ноутбука може оголосити, що він має більший рівень енергії, ніж його сусідній вузол, і залучити до себе весь трафік. Відтепер він може здійснити атаку Sybil, Black Hole, Wormhole.

1.3. Заходи протидії атакам у БСМ

Оскільки атаки на БСМ стають все більш складними, попит на нові рішення безпеки постійно зростає. Таким чином, за останнє десятиліття було розроблено та впроваджено низку нових схем безпеки. Більшість із цих схем було розроблено для надання рішень на основі кожного окремого рівня, а не на основі кожної атаки; при цьому вони залишають проміжок між рівнями, що може призвести до міжрівневих атак.

Загалом, будь-який підхід до забезпечення безпеки повинен забезпечувати аутентифікацію, цілісність, конфіденційність, доступність, контроль доступу та невідмовність. Крім того, фізична безпека абсолютно необхідна, щоб уникнути втручання або руйнування вузлів. Тому конструкція стійких до несанкціонованого доступу сенсорних вузлів є абсолютно необхідною. Однак такі схеми, захищені від несанкціонованого доступу, мають більш високу вартість виробництва і обмежуються додатками, які не тільки є критичними, але й використовують меншу кількість вузлів [10].

Аутентифікація. Головною метою аутентифікації є запобігання атак від зловмисника, що видає себе за інший вузол. Отже, аутентифікацію можна визначити як процес забезпечення того, що ідентичність комунікаційного об’єкта є тим, на що він претендує.

Цілісність. Мета цілісності полягає в тому, щоб підтвердити, що отримані дані не змінюються перехоплювачем під час спілкування (шляхом вставки, видалення або відтворення даних) і є саме такими, як їх надіслав авторизований відправник. Зазвичай для забезпечення цілісності даних використовуються криптографічні методи, такі як цифрові підписи та хеш-значення.

Конфіденційність. Метою конфіденційності є захист даних від несанкціонованого розголошення. Поширеним підходом до досягнення конфіденційності є шифрування даних користувача.

Доступність. Метою доступності є забезпечення доступності ресурсів системи (мережі) та їх використання уповноваженим органом на його запит.

Контроль доступу. Метою контролю доступу є забезпечення прав доступу до всіх ресурсів системи. Цей етап намагається запобігти несанкціонованому використанню системних і мережевих ресурсів. Контроль доступу тісно пов'язаний з атрибутами аутентифікації. Він відіграє важливу роль у запобіганні витоку інформації під час компрометації вузла. Одним із традиційних підходів до контролю доступу є використання порогової криптографії. Цей підхід приховує дані, розділяючи їх на кілька часток. Щоб отримати остаточні дані, кожен спільний ресурс має бути отриманий за допомогою процесу аутентифікації.

Невідмовність. Для пояснення цього етапу, необхідно розглянути приклад на прикладі. Нехай Аліса і Боб будуть двома вузлами, які бажають спілкуватися один з одним. Аліса відправляє повідомлення (M) Бобу. Пізніше Аліса стверджує, що не надсилала Бобу жодного повідомлення. Отже, виникає питання, як захистити Боба якщо Аліса заперечує будь-яку причетність до будь-якої форми спілкування з Бобом. Невідречення спрямоване на захист від тих, хто спілкується, які заперечують, що вони коли-небудь брали участь у будь-якому спілкуванні з жертвою.

1.3.1. Заходи безпеки на фізичному рівні

Щоб запобігти радіоперешкодам або завадам, використовуються дві поширені методики: розширений спектр зі стрибковою зміною частоти (FHSS) і розширений спектр прямої послідовності (DSSS). У FHSS сигнал модулюється на таких частотах, що він стрибає з однієї частоти до іншої випадковим чином через фіксований інтервал часу. Передавач і відповідний приймач переключаються між частотами, використовуючи один і той же псевдовипадковий код для модуляції та

демодуляції. Якщо підслуховувач перехоплює сигнал FHSS, якщо він не має попереднього знання коду розповсюджуваного сигналу, він не зможе демодулювати сигнал. Крім того, поширення сигналу на кількох частотах значно зменшить перешкоди.

У DSSS код розширення використовується для відображення кожного біта даних у вихідному сигналі на кілька біт у переданому сигналі. Псевдовипадковий код (розширювальний код) розповсюджує вхідні дані в більш широкому діапазоні частот порівняно з вхідною частотою. В частотної області вихідні сигнали виглядають як шум. Оскільки псевдовипадковий код забезпечує широку смугу пропускання для вхідних даних, він дозволяє потужності сигналу опускатися нижче порога шуму без втрати будь-якої інформації. Тому ця техніка є важко виявити підслуховувачем через нижчі рівні енергії на частоту та більшу стійкість до перешкод.

Вищезгадані схеми можуть забезпечити безпеку лише до тих пір, поки шаблон стрибків або код розповсюдження не будуть розкриті жодному зловмиснику.

1.3.2. Заходи безпеки на канальному рівні

Безпека канального рівня відіграє важливу роль у забезпеченні загальної безпеки. Протоколи цього рівня корисні при обробці доступу до каналу, виявлення сусідніх вузлів і контролю помилок кадрів. Застарілі протоколи безпеки, такі як SSL або IPSec, не можна застосовувати безпосередньо до БСМ, оскільки вони не забезпечують агрегацію даних та не дозволяють обробку в мережі, що є основними вимогами при розробці протоколів безпеки.

Щоб запобігти атакам відмови в обслуговуванні (DoS) на БСМ, пропонується, щоб кожен проміжний вузол в активному маршруті маршрутизації виконував аутентифікацію та перевірку цілісності [11].

Однак, якщо кілька проміжних вузлів на активному маршруті мають дуже низькі рівні енергії, і якщо вони змушені виконувати перевірки автентифікації,

вони витрачатимуть всю свою енергію і порушують активний маршрут. З іншого боку, якщо розглянути наскрізну аутентифікацію у БСМ, то вона є більш енергоефективною, оскільки вузол-приймач (з багатим ресурсом) є єдиним вузлом, який виконує аутентифікацію та перевірку цілісності. Тим не менш, ця схема вразлива до багатьох типів атак безпеки (Black Hole, Jamming, і т.д.). Отже, існує потреба в адаптивних схемах, які враховують рівні енергії кожного вузла під час прийняття рішення про схеми аутентифікації.

Ранні підходи до безпеки були зосереджені на техніках симетричного введення ключів, а аутентифікація була досягнута за допомогою коду аутентифікації повідомлення (MAC). Однією з поширених схем MAC є код аутентифікації повідомлення ланцюжка шифру. Однак ця схема не є безпечною для вхідних повідомлень змінної довжини. Таким чином, кінцевий користувач (сенсорні вузли) повинен доповнювати вхідні повідомлення так, щоб вони були кратними блочного шифру.

Тому кожен вузол повинен витрачати вхідні дані для доповнення енергії. Щоб подолати цю проблему, були запропоновані інші моделі блочного шифру, такі як CTR і OCB. Що стосується конфіденційності, то для захисту БСМ використовуються симетричні схеми шифрування DES, AES, RC5 і Skipjack (блокові шифри) і RC4 (потоковий шифр). Зазвичай блочним шифрам переважають потокові шифри, оскільки вони дозволяють аутентифікацію та шифрування. Кілька запропонованих структур безпеки канального рівня включають TinySec, Sensec, SNEP, MiniSec, SecureSense і ZigBee Alliance [12]. Однак ці схеми мають обмеження. Наприклад, у Tinysec один ключ вручну програмується на всіх вузлах датчиків мережі. Проста атака захоплення вузлів на будь-який із цих вузлів може призвести до витоку секретного ключа та скомпрометації всієї мережі. Для захисту TinySec потрібен більш потужний механізм ключів. Крім того, TinySec вимагає заповнення для вхідних повідомлень, розмір яких менше 8 байт. Він використовує блочний шифр для шифрування повідомлень, а для повідомлень, розмір яких менше 8 байтів, вузол повинен буде використовувати додаткову енергію для заповнення повідомлення перед шифруванням.

1.4. Аналіз захищеності підключених сенсорних датчиків та пристроїв

Для прикладу аналізу підключення можна запропонувати використання пошукової системи Shodan, щоб зрозуміти, чому безпека повинна бути центром будь-якої реалізації безпроводових сенсорних мереж.

У Shodan є сервери, розташовані по всьому світу, які постійно сканують мережу Інтернет у пошуках підключених пристроїв. Він може знайти певні пристрої та типи пристроїв. Потім ці дані можна більш детально розглянути та проаналізувати. Деякі з найбільш популярних пошуків включають: «веб-камера», «паролі за умовчанням», «маршрутизатори», «відеоігри» тощо.

Shodan - улюблений інструмент, який використовують дослідники, фахівці з безпеки, великі підприємства та комп'ютерні групи реагування на надзвичайні ситуації (CERT), адже:

- Дослідники можуть використовувати Shodan для отримання інформації про те, які пристрої підключені, де вони підключені та які послуги відкриті;
- Фахівці з безпеки можуть використовувати Shodan як частину плану тестування на проникнення для виявлення пристроїв, які необхідно загартувати, щоб запобігти потенційним атакам;
- На великих підприємствах працюють фахівці з безпеки, які повинні знати такі інструменти, як Shodan для визначення поточного профілю ризику підключених пристроїв підприємства;
- CERT можуть використовувати Shodan для швидкого створення звітів про нову атаку на підключені пристрої. Shodan також є інструментом, яким користуються зловмисники, яких зазвичай називають суб’єктами загрози. Shodan може прискорити розвідку суб’єкта загрози щодо підключених до мережі Інтернет [13].

1.4.1. Алгоритм підключення до пошукової мережі Shodan

Для доступу до пошукової системи необхідно відкрити веб-браузер та перейти на веб-сайт Shodan за адресою https://www.shodan.io/.

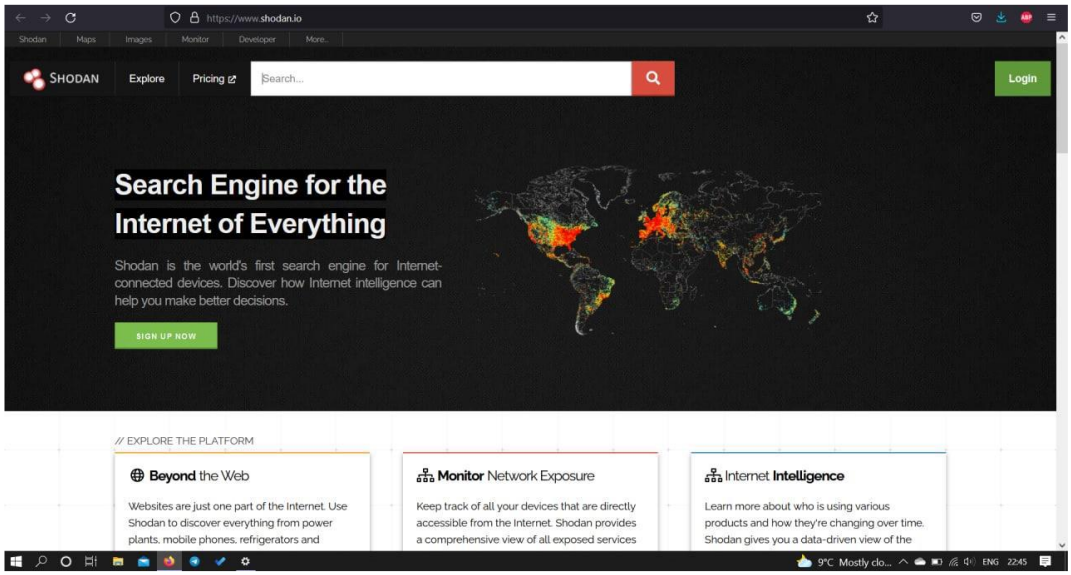

Рис.1.9. Сторінка пошукової системи Shodan

Наступним кроком є крок створення облікового запису [14]. Він може бути реалізований одним із двох способів:

1) Створення безкоштовного облікового запису (рис.1.10), шляхом заповлення форми для створення облікового запису.

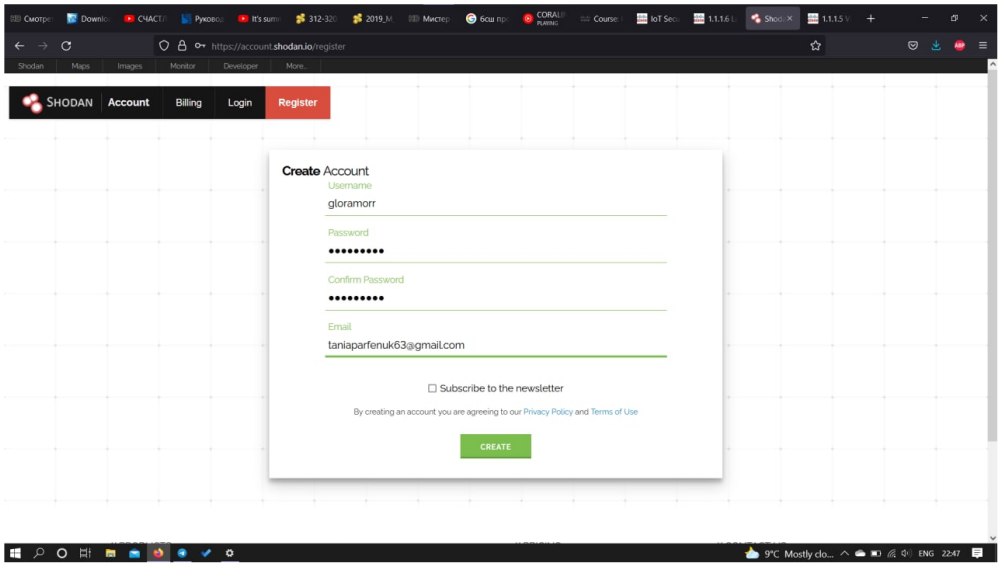

Рис.1.10. Реєстрація в пошуковій системі Shodan

2) «Увійти» або «Зареєструватись». Таким чином, будь-який користувач буде перенаправлений на сторінку, де зможе увійти за допомогою одного з кількох інших облікових записів, які вже є наявними, включаючи Google або Facebook.

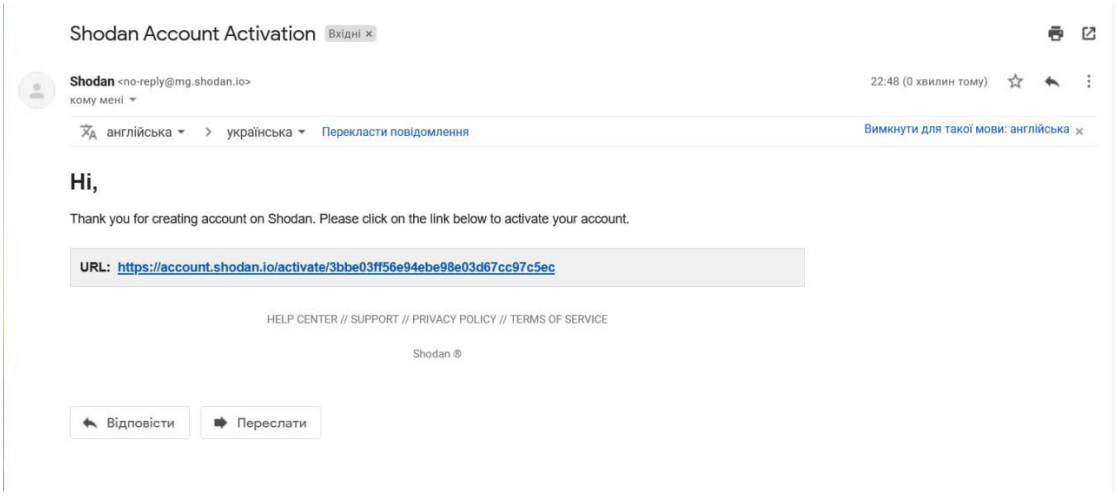

Рис.1.11. Сторінка облікового запису в пошуковій системі Shodan

1.4.2. Дослідження підключених сенсорних пристроїв

Так як одним із найбільш перспективних напрямків БСМ являється реалізація IoT, то подальше дослідження базується саме на тестуванні сенсорних датчиків та пристроїв. Які мають відношення (підключенні) до мережі Інтернет Речей [15].

Крок 1. Використання основних можливостей пошукової системи Shodan. На головній сторінці можна ввести ключові слова у поле пошуку, щоб отримати список результатів. Після введення ключового слова «cisco», користувач отримає порядку 6 502 291 результатів (рис.1.12).

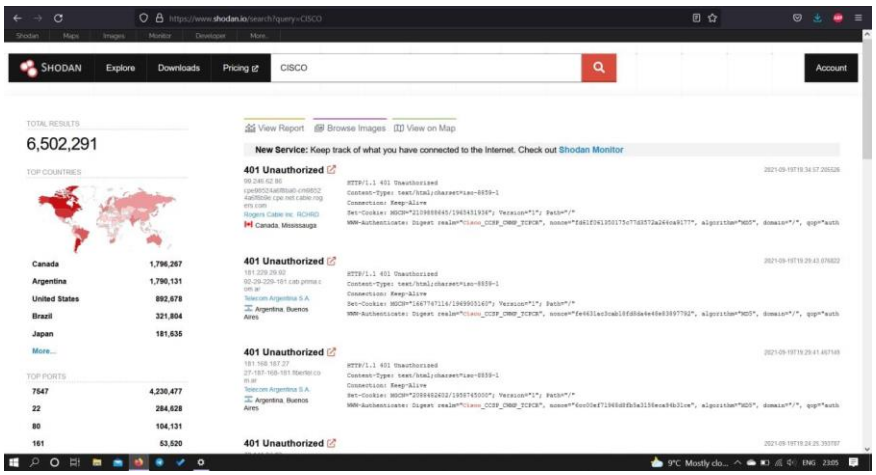

Рис.1.12. Результати перевірки ключового слова «cisco»

Крок 2. Виокремлення результатів пошуку. Результати пошуку поділено на різні категорії. Кожен запис у категорії є посиланням, на яке можна натиснути, що уточнить ціль пошук. У правому боці головного розділу пошуку відображаються пристрої, які відповідають запиту пошуку. Наприклад можна розглянути наступний запис:

IP -адреса: 68.169.131.159

Ім'я хосту: host-68-169-131-159.CMGOLT1.epbfi.com

ISP: Волоконна оптика EPB

Дата додавання запису: 19.09.2021 (останнє оновлення)

Країна: США

При натисканні на банер, можна отримати додаткову інформацію, в якій будуть вказані і деталі щодо безпеки.

Місто та країна: Чаттануга, Сполучені Штати

Відкриті порти: 22, 23, 80, 443

Послуги працюють: -

Ключові типи: ssh-rsa

Крок 3. Дослідження результатів пошуку. Повернувшись на домашню сторінку Shodan, і натиснувши кнопку «Дослідити», можна перевірити найпопулярніші результати, такі як: «веб-камера», «камери», «мережева камера», «пароль за замовчуванням», «ufanet». Одним із найцікавіших результатів для дослідження є запит «пароль за замовчуванням». Результати запиту приведені на рис.1.13

Саме стільки безпроводових пристроїв, включених до мережі Інтернет речей, функціонують із паролем за замовчуванням. При виборі запиту «веб-камера», отримується результат 9173, що свідчить про те, що саме стільки елементів підключені до IoT мережі на момент дослідження [16].

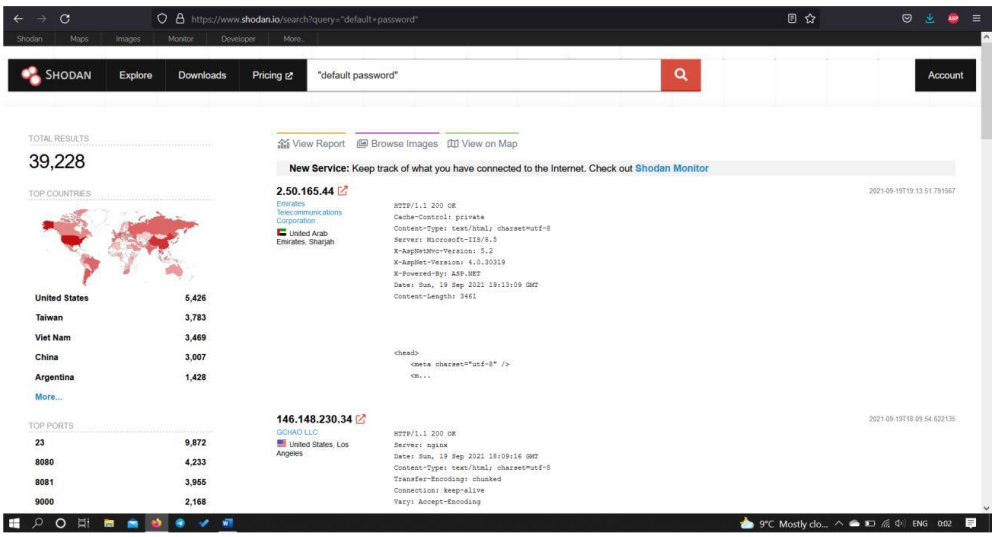

Рис.1.13. Результати запиту«пароль за замовчуванням»

Крок 4. Використання ключових слів разом з пошуковими операторами. Цей крок дає змогу відфільтрувати пошук. Shodan шукає служби, що працюють на пристрої. Потім він збирає банерну інформацію для кожної послуги. Наприклад, ось інформація про банер для служби SNMP, що працює на пристрої Cisco, знайденої за допомогою пошуку Shodan:

Програмне забезпечення операційної системи Cisco Internetwork IOS (tm) 7200

Програмне забезпечення (UBR7200-IK9SU2-M), Версія 12.3 (23) BC10, ВИПУСК ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ (fc1)

Технічна підтримка: http://www.cisco.com/techsupport

Авторське право (c) 1986-2011 рр. Компанією cisco Systems, Inc.

Пошук просто запиту «cisco», швидше за все, виявить від одного до двох мільйонів результатів. Ця інформація може бути корисною. Однак, якщо користувача цікавить більш конкретна інформація, то необхідно відфільтрувати пошук, використовуючи назви фільтрів та значення з інформації банера [17].

Наприклад, якщо подивитися, скільки маршрутизаторів Cisco 7200 у Сполучених Штатах використовують службу SNMP, то необхідно ввести наступну послідовність:

країна: США

продукт: "Маршрутизатор Cisco 7200"

порт: 161

Пошук Shodan використовує дволітерний (альфа-2) код країни на основі публікації 3166 Міжнародної організації з стандартизації (ISO 3166-1993).

Shodan є безкоштовним у користуванні сервісом, тому потенційно кожен може отримати доступ до цієї інформації, аби використати її у своїх цілях.

Будь-який користувач може перевірити, чи є вразливі місця у його IP-адреси, скориставшись Сканером Інтернету речей за такою адресою: https://iotscanner.bullguard.com. Натиснувши «Перевірити», чи «я на Shodan», щоб дозволити «Сканер Інтернету речей» для сканування IP-адреси. Завершення цього процесу може зайняти деякий час.

Висновки до першого розділу

Проаналізовано безпроводові сенсорні мережі, як підклас безпроводових мереж, та зазначено, що через обмеженість ресурсів БСМ вони породжують цілий новий набір атак.

Представлено узагальнену таксономію атак на БСМ, та приведено аналіз різних видів атак на такі мережі та способи боротьби з ними.

Зазначено, що на сьогодні одним із найбільш перспективних напрямків БСМ являється реалізація IoT, а також проведено дослідження вразливості сенсорних датчиків, сенсорів та «розумних» пристроїв, що мають відношення до мережі IoT за допомогою пошукової системи Shodan.

Зроблено висновок, що більшість пристроїв БСМ та IoT вразливі та незахищені, адже: або використовують паролі за замовчуванням, або не підключають мережеві фільтри, тому їх легко можна відшукати в глобальному просторі Інтернет.

2 ДОСЛІДЖЕННЯ ОСОБЛИВОСТЕЙ ВИКОРИСТАННЯ RASPBERRY PI ДЛЯ ЗАБЕЗПЕЧЕННЯ БЕЗПЕКИ БСМ

Значна кількість датчиків та сенсорів обєднанні між собою та підключенні до Raspberry Pi. Крім того, ядро Windows IoT також працює на Raspberry Pi, що пропонує розробникам як Linux (Raspbian та інші версії) разом з Windows як вибір для розробки [18].

2.1. Особливості обладнання Raspberry Pi

Основні особливості одноплатного пристрою Raspberry Pi, важливі для побудови захищеного функціонування, включають:

- Процесор: 4-х ядерний ARM Cortex A53 з частотою 1,2 ГГц (набір інструкцій ARMv8). Використання кожного ядра для певних функцій буде мати вирішальне значення для збору та ідентифікації поведінки пристроїв IoT;
- Пам'ять: 1 ГБ LPDDR2-900 SDRAM. Використання розширеної пам'яті Pi 3 допоможе зменшити введення-виведення на повільніший пристрій SD;
- Мережа: Ethernet 10/100 MBPS, безпроводова локальна мережа 802.11n, Bluetooth 4.0. Вбудована мережева опція дозволяє використовувати основні функції Pi для основних інтерфейсів моніторингу мережі, таких як: проводовий Ethernet, Wi-Fi або пристрої Bluetooth .
- USB -порти: 4. Надає необхідні можливості розширення для підтримки інших безпроводових технологій, таких як ZigBee.

ОС Raspbian. На додаток до самого Pi, необхідно розглянути ОС Raspbian на Pi. Зокрема, в роботі буде розглянуто використання Raspbian GNU/Linux 8.

Python. Python - це мова вибору для всіх компонентів програмного забезпечення, що розробляється. В роботі буде приведено скрипт коду на Python 2.7.9, однак, з незначними змінами, код також працюватиме на Python 3.x [19].

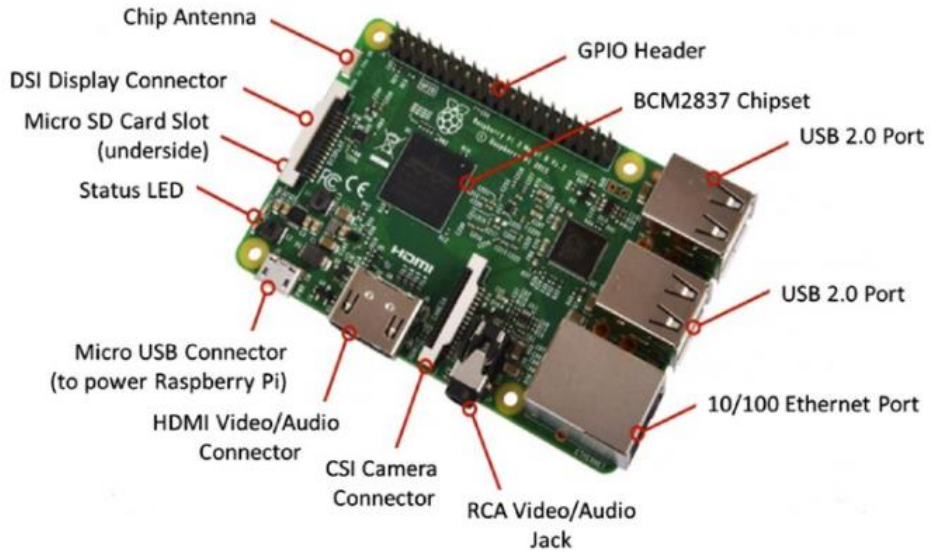

Рис.2.1. Raspberry Pi 3 Модель B

За кількома винятками, будуть використовуватися лише модулі стандартної бібліотеки Python, тим самим усуваючи необхідність встановлювати або найголовніше розуміти основи, продуктивність та ризики, пов'язані з сторонніми бібліотеками, що також можна вважати вразливістю. Це в основному рішення щодо продуктивності та безпеки, яке збереже Pi настільки мінімальним і безпечним, наскільки це можливо.

2.2. Моделювання поведінки Raspberry Pi

Пасивний моніторинг поведінки IoT. У порівнянні з активним моніторингом, пасивний моніторинг забезпечує краще уявлення про діяльність мережі, що контролюється. Використання таких інструментів, як NMAP для ідентифікації пристроїв, що працюють у БСМ, забезпечує миттєвий перегляд тих пристроїв, які належним чином реагують. У багатьох випадках пристрої IoT чи датчики БСМ є перехідними і тому можуть і будуть пропущені активними методами або методами на основі моніторингу. Зміна поведінки цих пристроїв протягом тривалого періоду часу має вирішальне значення для розуміння потенційних загроз, які вони несуть, поряд з підключеннями до інших пристроїв [20].

Моделювання нормальної поведінки. До ключових елементів, за якими можна вести спостереження, відносяться: Ethernet, джерело MAC-адреси, MAC-адреса призначення, тип кадру (IPv4, IPv6, ARP), IP-рівень, IP джерела, IP призначення, протокол, транспортний рівень, джерело порту, порт призначення.

Оскільки ймовірно, що під час моніторингу можна зустріти багато пакетів з однаковими значеннями MAC, IP-адреси джерела, IP-адреси призначення, протоколу, потрібно скоротити дані, які будуть зберігатися щодо спостережень.

Вбудований тип даних словника в Python забезпечує ідеальне рішення для зберігання спостережень. Словники Python, подібно до традиційних словників у стилі Вебстера, мають ключ і значення, яке зазвичай називають парою ключ-значення. У Python і ключ, і значення можуть бути складними, єдине правило полягає в тому, що ключ повинен мати хеш-тип, такий як ціле число, рядок або кортеж. Частиною значення пари ключ/значення може бути список або інший тип даних, які не можна розділити.

Далі буде використано наступний кортеж як ключ: SRC-MAC, DST-MAC, SRC-IP, DST-IP, протокол, порт. Не було включено порт SRC та DST. Причина в тому, що коли клієнт здійснює з'єднання з сервером, обраний порт є динамічним і зазвичай надходить з ефемерного набору портів. Таким чином, порт, який буде включений до ключа, буде нефемерним. Якщо обидва порти є нефемерними, то в словнику буде зроблено два записи: один із використанням порту SRC, а інший - порту DST. Якщо обидва порти є ефемерними, знову будуть внесені обидва записи.

2.3. Особливості захоплення пакетів у Python на Raspberry Pi

1. Як показано на рис.2.1, потрібно приєднати Pi до моніторингового або SPAN-порту комутатора Ethernet разом із Wi-Fi 802.11 [21].

2. Далі потрібно перевести порт Pi Ethernet у безладний режим.

3. Нарешті, потрібно захопити пакети за допомогою стандартної бібліотеки сокетів Python.

Простий фрагмент коду для виконання цих операцій (захоплення одного пакета за допомогою Python) приведено нижче.

```
Capture a single packet in promiscuous mode
Note: you must run this script as super user i.e. sudo python sniff.py
import os # Python operating system standard library
import socket # Python low level socket standard library
import sys # Python system standard library from binascii
import hexlify # Python binary ascii conversions standard library
# configure Raspberry Pi eth0 in promiscuous mode
# using a system command
try:
ret = os.system("ifconfig eth0 promisc")
except Exception as err:
print "System Command Failed: ", str(err)
sys.exit(0)
if ret == 0:
# If the command was successful print 'Promiscuous Mode Set Correctly'
# create a new socket using the python socket module
# PF_PACKET : Specifies Protocol Family Packet Level
# SOCK_RAW : Specifies A raw protocol at the network layer
# htons(0x0800) : Specifies all headers and packets
# : Ethernet and IP, including TCP/UDP etc
try:
# attempt to open the socket for capturing raw packets
rawSocket=socket.socket(socket.PF_PACKET,socket.
SOCK_RAW,
socket.htons(0x0800))
except Exception as err:
# catch any exceptions and report the error print "Socket Error", str(err)
sys.exit(0)
```

```
# If socket is established and we have established promiscuous mode
print "Network : Promiscuous Mode"
print "Sniffer : Ready: \n"
# attempt to receive a packet
# Note: this function call is synchronous, thus it will wait)
try:
recvPacket=rawSocket.recv(65535)
print "Packet Received:"
print hexlify(recvPacket)
print "\nEnd"
except Exception as err:
# Catch any exceptions and report the error
print "Receive Socket Error: ", str(err)
sys.exit(0)
else:
print "System Command Failed to set promiscuous mode"
```

1. Код переводить стандартний порт Ethernet Pi в безладний режим. Це дозволяє переглядати будь-який трафік, що надходить до мережі, навіть якщо він не призначений або походить від самого Pi.

2. Код активує роз’єм, пов'язаний з портом Ethernet, для прослуховування трафіку, що проходить через мережу.

3. Код фіксує один пакет і відображає результати в шістнадцятковій системі.

Визначення та вилучення пакетів. Наступним кроком у процесі є захоплення, а потім аналіз вмісту пакета. Це включає вилучення компонентів Ethernet, IP, ARP, TCP, ICMP, IGMP та UDP у першому прикладі. Протокол IGMP використовується для встановлення членства в групах багатоадресної розсилки. Протоколи багатоадресної розсилки зазвичай використовуються пристроями IoT для виявлення пристроїв поблизу разом із послугами, які вони пропонують. Щоб впоратися з цим, було створено новий сценарій Python під назвою “PacketRecorder.py”, який постійно фіксує пакети, витягує ключову інформацію та

записує входження кожної унікальної комбінації у словник Python. Огляд сценарію PacketRecorder.py представлено далі:

```
# Main Script Starts Here
#=====================================
if __name__ == '__main__':
"Python Packet Recorder v.50"
"Python Forensics, Inc. July 2017 \n"
# create a packet processing object packetObj = PacketProcessor()
# Python Packet Capture
# configure the eth0 in promiscuous mode
try:
ret = os.system("ifconfig eth0 promisc")
except Exception as err:
print "System Command Failed: ", str(err)
sys.exit(0)
if ret == 0:
print 'Promiscuous Mode Enabled for eth0'
# create a new socket using the python socket module
# PF_PACKET : Specifies Protocol Family Packet Level
# SOCK_RAW : Specifies A raw protocol at the network layer
# htons(0x0800) : Specifies all headers and packets
# : Ethernet and IP, including TCP/UDP etc
# attempt to open the socket for capturing raw packets
try:
rawSocket=socket.socket(socket.PF_PACKET,socket.
SOCK_RAW,
socket.htons(0x0800))
except Exception as err:
print "Socket Error", str(err)
sys.exit(0)
```

```
print "Packet Processor : Ready: \n"
# Set signal to 1 hour
signal.signal(signal.SIGALRM, handler)
signal.alarm(3600)
try:
while True:
# attempt to receive (synchronous call)
try:
recvPacket=rawSocket.recv(65535)
packetObj.PacketExtractor(recvPacket)
except Exception as err:
packetObj.printMap()
print "Receive Socket Error: ", str(err)
sys.exit(0)
except myTimeout:
packetObj.printMap()
packetObj.SaveOb("observations.pickle")
print "\nEnd Packet Processor"
sys.exit(0)
else:
print "System Command Failed to set promiscuous mode"
```

Сценарій виконує такі операції:

1. Створює об'єкт PacketProcessor, який буде використовуватися для вилучення та запису ключової інформації з кожного пакета.

2. Налаштовує порт Ethernet 0 на Raspberry Pi у безладному режимі [23].

3. Створює необроблений сокет за допомогою цього безладного порту.

4. Встановлює таймер сигналу для збору пакетів протягом 1 години (3600 секунд).

5. Створює цикл для прийому пакетів.

6. Потім кожен прийнятий пакет передається методу PacketExtractor об'єкта PacketProcessing.

7. Нарешті, як тільки таймер закінчиться, метод PrintMap об'єкта PacketProcessing викликається, щоб роздрукувати результати.

Під час перевірки спостережуваних пакетів важливо не перевантажувати цей процес значним кодом, базами даних тощо. Завданнямпрограми PacketRecorder.py - створити базову лінію мережі «нормальної роботи». Це фактично створить детальну карту активів на рівні мережевого пристрою для середовища, яке контролюється. Цю карту слід порівнювати з іншими доступними картами пристроїв (наприклад, згенерованими NMAP, адміністративною документацією тощо). Таким чином, підхід полягає у попередній обробці списків відомих хороших/поганих портів, кодів країн та індексів виробника, а також у швидкому пошуку тих значень, які можна легко додати до словника спостережень. Звичайно, можна генерувати аномалії, виявлені під час процесу базового аналізу.

Кожен із пошукових запитів обробляється у порівнянному порядку, який починається з перетворення онлайн-даних у об’єкти словника. Залежно від складності онлайн-джерела даних, розбір та підготовка цих словників можуть бути простими або досить складними. Після завершення етапу попередньої обробки перетворюються отримані об’єкти словника в серіалізовані дані (файли збірки Python), які завантажуються на Pi. Таким чином, Pi не вимагає доступу до мережі Інтернет під час етапів базового аналізу чи оперативного зондування [23].

Щоб ефективно це зробити, можна отримати інформацію з надійних джерел. А саме:

1. Виробник IEEE;

2. IANA для відомого перекладу номера порту/імені;

3. База даних розташування країни Maxmind.

2.4. Використання файлів Pickle у PacketRecorder.py

Інтеграція файлів pickle досягається шляхом створення класу для кожного типу пошуку. Ініціалізація (або конструктор класу) завантажує пов'язаний файл .pickle у словник, пов'язаний з об'єктом. Поб тім включається метод пошуку, що дозволяє швидко шукати потрібну конверсію: тип пакета Ethernet, MAC-адреса для пошуку виробника, пошук транспортного протоколу, пошук імені порту, пошук IP-адреси країни

Наступні фрагменти коду надають код для кожного з класу, пов'язані з пошуком.

```
class ETH:
def __init__(self):
''' FrameTypes Supported'''
self.ethTypes = {}
with open("ethTypes.pickle2",'rb') as fp:
self.ethTypes = pickle.load(fp)
def lookup(self, ethType):
''' Returns the FrameType associated with the lookup or
not=supported'''
try:
result = self.ethTypes[ethType]
except:
result = "not-supported"
return result.strip()
# MAC Address Lookup Class
class MAC:
def __init__(self):
''' constructor'''
# Open the MAC Address OUI Dictionary
try:
```

with open('oui.pickle', 'rb') as pickleFile:

self.macDict = pickle.load(pickleFile)

except Exception as err:

print str(err)

def lookup(self, macAddress):

try:

result = self.macDict[macAddress]

if len(result) >= 2:

result = ": ".join(result[0:2])

else:

result = result[0]

return result

except:

return "unknown"

 # Transport Lookup Class

class TRANSPORT:

def __init__(self):

 # Open the Transport protocol Dictionary

with open('protocol.pickle', 'rb') as pickleFile:

self.proDict = pickle.load(pickleFile)

def lookup(self, protocol):

try:

result = self.proDict[protocol]

return result

except:

return ["unknown", "unknown", "unknown"]

#PORTS Lookup Class

class PORTS:

def __init__(self):

 # Open the Transport protocol Dictionary

```
with open('ports.pickle', 'rb') as pickleFile:
self.portsDict = pickle.load(pickleFile)
def lookup(self, port, portType):
try:
lookupValue = (str(port).strip(),portType)
result = self.portsDict[lookupValue]
return result
except:
return "unknown"
#
# Country Lookup
#
class COUNTRY:
def __init__(self):
# download from http://dev.maxmind.com/geoip/legacy/
geolite/
self.giv4 = pygeoip.GeoIP('geoIPv4.dat')
self.giv6 = pygeoip.GeoIP('geoIPv6.dat')
def lookup(self, ipAddr, kind):
try:
if kind == 'IPv4':
return self.giv4.country_name_by_addr(ipAddr)
elif kind == 'IPv6':
return self.giv6.country_name_by_addr(ipAddr)
else:
return ''
except:
return ''
```

2.5. Визначення та доступ до методів пошуку

Наступним кроком є створення екземплярів кожного з класів у об’єктах, що можуть використовуватися локально, а потім використання відповідних функцій пошуку під час обробки спостережуваного пакета. Фрагмент коду для створення екземплярів класів в об’єкти

```
class PacketProcessor:
"""
Packet Processor Class Methods
__init__ Constructor
PacketProcessor(self, packet) : processes a single packet
PrintMap(self) : prints out the content of the map
"""
def __init__(self):
"""Constructor"""
'''
Create Lookup Objects
These Object provide lookups for:
Ethernet Frame Types
MAC Addresses
Transport Protocol Types
TCP/UDP Port Names
Country
'''
self.traOBJ = TRANSPORT()
self.ethOBJ = ETH()
self.portOBJ = PORTS()
self.ouiOBJ = MAC()
self.cc = COUNTRY()
```

2.6. Використання пошуку під час обробки пакетів

Тепер, коли об’єкти self.traOBJ, self.ethOB, self.portOBJ, self.ouiOBJ та self.cc створено, можна використовувати їх під час звичайної обробки пакетів.

Зразок перетворення обробки IPv4. Цей уривок зображує перетворення вихідної та цільової IP-адрес у місцезнаходження країни та перетворює номер протоколу пакета IPv4 у назву відповідної країни. Приклад звіту представлено на рис.2.2, і він включає: назва порту, виробник, середній розмір пакета. Крім того, знизу можна побачити, що тепер включено захоплення пакетів IPv6 [24]

```
# covert the source and destination address to typical dotted notation strings
self.packetSize = packetLength
self.srcIP = socket.inet_ntoa(sourceIP);
self.dstIP = socket.inet_ntoa(destIP);
self.srcCC = self.cc.lookup(self.srcIP, 'IPv4')
self.dstCC = self.cc.lookup(self.dstIP, 'IPv4')
translate = self.traOBJ.lookup(str(protocol))
transProtocol = translate[0]
Convert the Port Numbers into Port Names (Excerpt)
# unpack the TCP Header to obtain the
# source and destination port
tcpHeaderBuffer = struct.unpack('!HHLLBBHHH', stripTCPHeader)
self.srcPort = tcpHeaderBuffer[0]
self.dstPort = tcpHeaderBuffer[1]
self.srcPortName = self.portOBJ.lookup(self.srcPort, 'TCP')
self.dstPortName = self.portOBJ.lookup(self.dstPort, 'TCP')
```

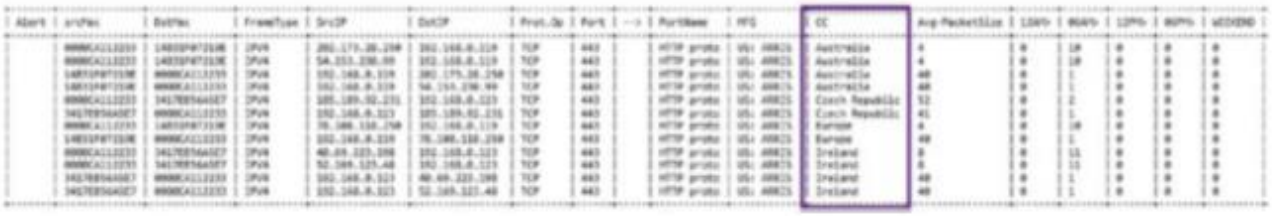

Рис.2.2. Приклад звіту захоплення пакетів

Висновки до другого розділу

Описано Raspberry Pi, операційну систему Raspbian та мову програмування Python, яка використовується сьогодні в БСМ.

Розглянуто Raspberry Pi за допомогою декількох спеціальних інструментів командного рядка Raspbian Pi, а також розглянуто як переваги, так і деякі потенційні обмеження Pi на основі наявної пам'яті та простору файлової системи.

Досліджено процес збору, аналізу та відображення мережевого трафіку. Основні методи фіксації та запису спостережень будуть використані для створення базової лінії «нормальних» операцій у середовищі IoT.

Зазначено, що ці спостереження будуть використовуватися для моніторингу та виявлення аномальної поведінки та навчання методів машинного навчання.

Представлено сенсорний скрипт, який буде використовувати попередньо записану базову лінію (створену PacketRecorder) та повідомлятиме про аномалії між базовою лінією та оточуючим середовищем.

3 ТЕХНОЛОГІЯ МОНІТОРИНГУ ЗА АНОМАЛІЯМИ ТА ЗАХОПЛЕННЯ МЕРЕЖЕВОГО ТРАФІКУ В БСМ ЗА ДОПОМОГОЮ RASPBERRY PI

3.1. Перетворення пакетного реєстратора в датчик

Оскільки PacketRecorder перетворюється на повну сенсорну платформу, яка може контролювати мережу в режимі реального часу та повідомляти про аномалії, необхідно внести кілька серйозних удосконалень. Ці вдосконалення покликані полегшити роботу:

1. Управління диктофоном і датчиком за допомогою інтерфейсу.

2. Створення HTML-звітів, які охоплюють наступне: загальний основний звіт, спостереження за MAC-адресами, спостереження за зв'язками з країнами, спостереження за використанням порту, виявлення можливих з'єднань БСМ, спостереження за можливим підключенням промислової системи управління (ІКС), сповіщення, що генеруються в режимі датчика.

3. Надання основної інформації про стан безпосередньо на Raspberry Pi.

4. Створення реєстратора/датчика як єдиного виконуючого файлу.

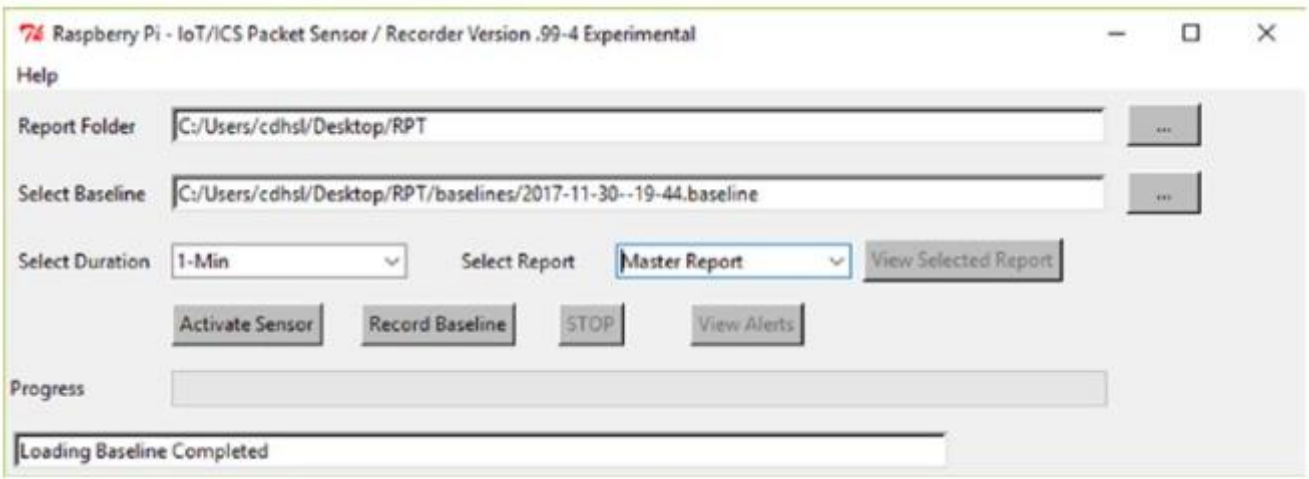

Рис.3.1. Знімок графічного інтерфейсу датчика/реєстратора Raspberry Pi

Основні операційні елементи проектування включають керований подіями графічний інтерфейс (графічний інтерфейс користувача), повністю розроблений на

Python, із використанням бібліотеки TKinter (рис.3.1). Це забезпечує сумісність з новими пристроями Raspberry Pi у міру їх просування.

Крім того, до самого Raspberry Pi додано дисплей ePaper у режимі реального часу (як додатковий елемент) (рис.3.2). Це забезпечує зворотний зв'язок безпосередньо з Raspberry Pi в режимах запису та моніторингу.

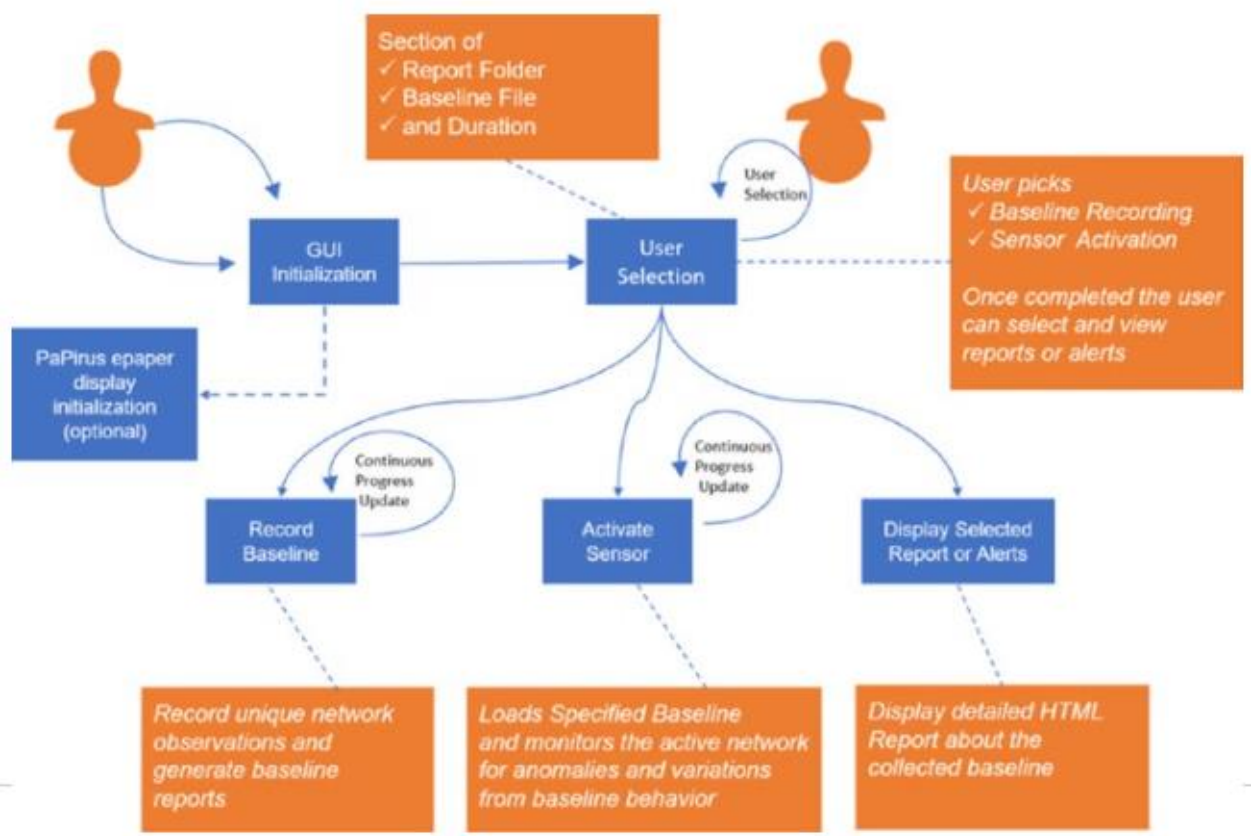

Рис.3.2. Raspberry Pi налаштований за допомогою дисплея PaPirus в режимі реального часу

На рис.3.3. зображено загальну функціональну конструкцію датчика/реєстратора Pi.

Рис.3.3. Датчик/реєстратор Raspberry Pi

Датчик/реєстратор Pi налаштований на виконання в межах циклу програми, керованого подіями, що підтримується Python та TKinter. Розділ основного коду виглядає так:

```
# Script Constants
# M1
NAME = "Raspberry Pi - IoT/ICS Packet Sensor / Recorder"
VERSION = " Version .99-4 Experimental "
TITLE = NAME+'\t'+VERSION
# Initialize the root TK Window
# M2from Tkinter import *
root = Tk()
def main():
# Set the Title for the Main Window
# M3
root.title(TITLE)
# Instantiate the GUI Application Object
app = Application(root)
# Start App MainLoop
app.mainloop()
# Main Script Starts Here
```

 # M3

if __name__ == '__main__':

main()

Вивчаючи основні розділи класу додатків, виокремлюються три основні розділи:

A1. Встановлює атрибути об'єктів, які будуть пов'язані з кожним екземпляром класу програми. Наприклад, тут ініціалізуються змінні, які містять інформацію про стан щодо базової лінії та вибраних звітів, разом із словниками, які будуть використовуватися датчиком під час діяльності з моніторингу.

A2. Створює модель, яка буде використовуватися програмою. Найголовніше, викликається метод initUI (): - встановлює всі віджети графічного інтерфейсу у вікні, такі як мітки, кнопки, текстові поля, випадаючі списки, смуги виконання, дисплеї стану та параметри меню.

A3. Нарешті, якщо дисплей PaPirus був приєднаний і виявлений, створюється об'єкт для управління інтерфейсом з дисплеєм.

```
    def btnSelectFile(self):
    # Handle Folder Browse Button Click

    def btnSelectFolder(self):
    def btnSelectBaseLine(self):
    def btnViewSelectedReport(self):
    def btnViewAlerts(self):
    def btnActivateSensor(self):
    def btnPerformCapture(self):
    def SaveOb(self, d):
    def GenCSV(self, d):
    def GenHTML(self, d):
    def GenCOUNTRY(self, d):
    def GenICS(self, d):
    def GenIOT(self, d):
    def GenMFG(self, d):
    def GenPortUsage(self, d):
    def translateAlertCodes(self, alerts):
    def GenAlerts(self, d):
    '''
    def btnSTOPCapture(self):
    def menuToolsExit(self, event=True):
    '''
    def menuAbout(self, event=True):
```

Рис.3.4. Об'єктні методи застосування

Короткий опис методів застосування додатків наведено в табл.3.1.

Таблиця 3.1.

Визначення методів функціонування та застосування додатків

Метод	Опис
initUi	Створює та ініціалізує всі віджети відображення, знайдені на фреймі програми. Як тільки всі віджети створені, завантажуються таблиці пошуку, що використовуються програмою, і оновляється рядок стану
btnselectFolder	Надає користувачеві можливість перегляду папок. Користувач повинен вибрати існуючу папку або створити нову папку для збереження результатів базового сценарію запису або активувати вибір датчиків
Btnselectbaseline	Обробляє вибір існуючої базової лінії, яка буде використовуватися в режимі датчика для виявлення аномалій із записаної базової лінії, наприклад, нових з'єднань, пристроїв, використання портів та країн, з якими ви контактуєте
btnperformCapture	Розпочинає процес запису базового сценарію. На основі обраної тривалості цей метод буде працювати до кінця, якщо його не перерве кнопка зупинки
Btnactivatesensor	Використовує вибрану базову лінію та розпочинає процес моніторингу активності мережі та порівняння цих результатів із базовою. Як і метод btnperformCapture, буде працювати протягом вибраної тривалості, якщо не буде перерваний кнопкою зупинки
btnstopCapture	Активується після натискання кнопки зупинки під час базового запису або виконання датчика. Це перерве запис або датчик, але збереже проміжні результати
GenhtML	Створює основний звіт htML
GenCountry	Створює звіт htML країни

Продовження таблиці 3.1.

btnViewselectedreport	Відображає поточний звіт, вибраний користувачем у спадному меню вибору звіту. Можливі звіти включають наступне: основний звіт, звіт виробника (назва пристрою oUi), звіт про країну, звіт про використання порту, звіт iot, звіт iCs
btnViewalerts	Активується користувачем натисканням кнопки перегляду сповіщень. Ця кнопка активна лише після завершення роботи датчика. метод відображатиме поточний звіт попередження, створений за останньої операції датчика
Saveob	Зберігає інформацію про базову лінію
GenCsV	Створює файл значень (CsV) у папці звітів. CsV містить усі унікальні спостереження під час запису базового процесу
GenMFG	Створює htML -звіт про спостережуваних виробників
GenportUsage	Створює звіт portUsage htML
translatealertCodes	Перетворює коди попереджень, створені датчиком pi, у змістовні повідомлення

3.2. Методи btnSelectBaseLine для елементів графічного інтерфейсу

Цей метод використовує вбудовану назву tkFileDialog.askopenfilename для вибору базової лінії. Оскільки користувач може вибрати будь-який файл із розширенням .baseline, потрібно перевірити, чи це дійсна базова лінія, створена методом запису базової лінії. Після того, як це буде перевірено, створюється набір локальних словників для зберігання вилучених значень з базового рівня, включаючи країни, що спостерігалися раніше, та MAC-адреси; вони будуть використовуватися під час процесу моніторингу для створення сповіщень від невідомих країн та нових спостережуваних MAC-адрес [25].

 # F1 Report Folder Selection

def btnSelectFolder(self):

```
try:
self.folderSelection = tkFileDialog.
askdirectory(initialdir="./", title='Select Report Folder')
self.ReportFolder['text'] = self.folderSelection
if os.path.isdir(self.folderSelection) and os.access(self.folderSelection, os.W_OK):
self.reportFolderGood = True
self.statusText['text'] = "Report Folder
Selected"
self.update()
''' Ok to enable Record Baseline Button '''
self.CapturePackets['state']=NORMAL
if self.baselineGood:
self.ActivateSensor['state']=NORMAL
else:
self.reportFolderGood = False
self.statusText['text'] = "Invalid Folder
Selection ... Folder
must exist and be writable"
self.update()
except Exception as err:
self.reportFolderGood = False
self.update()
# Baseline Selection
def btnSelectBaseLine(self):
self.fileSelection = tkFileDialog.askopenfilename(initi
aldir="./",
self.fileSelection =
tkFileDialog.askopenfilename(initiald ir="./",
filetypes=[("Sensor Baseline Files","*.baseline")],
title='Select Baseline File') title='Select Baseline File')
```

```
self.fileBaseline['text'] = self.fileSelection
if self.fileBaseline:
try:
with open(self.fileSelection, 'rb') as base:
try:
''' Make sure we loaded a dictionary '''
self.baselineDictionary = pickle.
load(base)
''' Make sure the elements match our
structure'''
if type(self.baselineDictionary) is dict:
value = self.baselineDictionary.
values()[0]
if value[POV] == 'S' or value[POV] == 'D':
self.baselineGood = True
else:
self.baselineGood = False
self.statusText['text'] = "Baseline Load Failed"
if self.baselineGood:
''' Create Quick Lookups for Country, MFG'''
self.statusText['text'] = "Loading Baseline Contents"
self.update()
for key, value in self.
baselineDictionary.
iteritems():
try:
srcCC = value[SRCCC]
dstCC = value[DSTCC]
srcMAC = value[SRCMAC]
dstMAC = value[DSTMAC]
```

```
if srcCC != '' and srcCC.lower() != 'unknown': self.
baselineCC[srcCC] = 1
if dstCC != '' and dstCC.lower() != 'unknown': self.
baselineCC[dstCC] = 1
if srcMAC != '' and srcMAC.lower() != 'unknown': self.
baselineMAC[srcMAC] = 1
if dstMAC != '' and dstMAC.lower() != 'unknown': self.
baselineMAC[dstMAC] = 1
except:
''' ignore errors in baseline loading'''
continue
self.statusText['text'] = "Loading Baseline Completed"
''' Ok to enable Activate Sensor Button '''
if self.reportFolderGood:
self.ActivateSensor['state' ]=NORMAL
except Exception as err:
self.statusText['text'] = "Baseline Load Failed"
except Exception as err:
self.statusText['text'] = "Baseline Load Failed: "+str(err)
self.update()
```

3.3. Метод Activate Sensor (btnActivateSensor, PacketMonitor)

Даний метод імітує пакувальний диктофон. Різниця полягає в обробці кожного прийнятого пакета. Метод PacketMonitor досліджує прийнятий пакет і визначає, чи існує «однакова конструкція пакета» в поточній базовій лінії. Якщо ні, то генерується пункт звіту з попередженням, що вказує на «Нове спостереження». Крім того, ключові елементи пакета, такі як місцезнаходження країни IP, MAC - адреса, розмір пакета та час спостереження, порівнюються з відомими або очікуваними значеннями. У разі виявлення аномалій реєструються додаткові статті

звіту. Нижче наведений фрагмент коду містить btnActivateSensor, PacketMonitor та допоміжні методи.

```
def btnActivateSensor(self):
# Handle Active Sensor Button Click
self.ActivateSensor['state']=DISABLED
saveCaptureState = self.CapturePackets['state']
self.CapturePackets['state']=DISABLED
self.StopCapture['state']=NORMAL
self.update()
# create a packet processing object
self.statusText['text'] = "Loading Lookups ..."
self.update()
self.packetObj = PacketProcessor(self.lookupList, self.baselineDictionary)
self.statusText['text'] = "Monitoring Packets ..."
if PA_ON:
self.statusText['text'] = "Resetting PaPirus Display
... Please Wait"
self.update()
self.paObj.ResetDisplay()
self.paObj.UpdateMode("Monitor")
self.paObj.UpdateStatus("Operation Started ")
self.statusText['text'] = "Monitoring Packets ..."
self.update()
durationValue = self.duration.get()
durSec = CONVERT[durationValue]
startTime = time.time()
curProgress = 0
self.progressBar['value'] = curProgress
self.alertDict = {}
# Python Packet Capture
```

```
# configure the eth0 in promiscuous mode
try:
ret = os.system("ifconfig eth0 promisc")
if ret == 0:
LogEvent(LOG_INFO, 'Promiscuous Mode Enabled for eth0')
# create a new socket using the python socket module
# PF_PACKET : Specifies Protocol Family Packet Level
# SOCK_RAW : Specifies A raw protocol at the network layer
# htons(0x0003) : Specifies all headers and packets
# : Ethernet and IP, including TCP/ UDP etc
# attempt to open the socket for capturing raw packets
rawSocket=socket.socket(socket.PF_PACKET,socket.
SOCK_RAW,
socket.htons(0x0003))
else:
self.statusText['text'] = "Monitoring Failed ... Cannot Open Socket"
self.progressBar['value'] = 0
self.update()
self.CapturePackets['state']=NORMAL
self.StopCapture['state']=DISABLED
self.update()
return
except Exception as err:
self.statusText['text'] = "Socket Exception ...
"+str(err)
self.progressBar['value'] = 0
self.CapturePackets['state']=NORMAL
self.StopCapture['state']=DISABLED
self.update()
return
```

```
pkCnt = 0
upTime = time.time()
paTime = time.time()
while True:
curTime = time.time()
elapsedTime = curTime - startTime
if elapsedTime > durSec:
break
if self.abortFlag:
''' User Aborted '''
''' Reset the Flag for next use '''
self.abortFlag = False break
''' Update the Progress Bar on Change vs Total Time'''
newProgress = int(round((elapsedTime/durSec * 100)))
if newProgress > curProgress:
self.progressBar['value'] = newProgress
curProgress = newProgress
self.update()
''' Update the Status Window every two seconds'''
newTime = time.time()
if (newTime - upTime) >= 2:
upTime = newTime
cntStr = '{:,}'.format(pkCnt)
self.statusText['text'] = "Pck Cnt: " + cntStr
self.update()
''' Update the PA Display if available '''
if PA_ON:
newPATime = time.time()
if (newPATime - paTime) >= 20:
paTime = newPATime
```

```
cntStr = '{:,}'.format(pkCnt)
self.paObj.UpdatePacketCnt(cntStr)
# attempt to receive (this call is synchronous, thus it will wait)
try:
recvPacket=rawSocket.recv(65535)
self.packetObj.PacketMonitor(recvPacket, self.alertDict, self.baselineCC, self.baselineMAC)
pkCnt += 1
except Exception as err:
LogEvent(LOG_INFO,'Recv Packet Failed: '+str(err))
continue
# Generate Sensor Reports
self.GenAlerts(self.alertDict)
''' Enable Report Button '''
self.ViewAlerts['state']=NORMAL
''' Reset Progress Bar and Post Completed status'''
self.progressBar['value'] = 0
cntStr = '{:,}'.format(pkCnt)
alertsGenerated = '{:,}'.format(len(self.alertDict))
self.statusText['text'] = "Done: Total Connections Processed
: "+cntStr+" Alerts: "+alertsGenerated
self.CapturePackets['state'] = saveCaptureState
self.ActivateSensor['state']=NORMAL
self.StopCapture['state']=DISABLED
self.update()
if PA_ON:
self.paObj.UpdateAlertCnt(alertsGenerated)
self.paObj.UpdatePacketCnt(cntStr)
self.paObj.UpdateStatus("Operation Completed")
self.paObj.UpdateMode(" ")
```

def PacketMonitor (self, packet, alertDict, baseCC, baseMAC):

''' Extract Packet Data input: string packet, dictionary d result is to update dictionary d '''

ETH_LEN = 14 # ETHERNET HDR LENGTH

IP_LEN = 20 # IP HEADER LENGTH

IPv6_LEN = 40 # IPv6 HEADER LENGTH

ARP_HDR = 8 # ARP HEADER

UDP_LEN = 8 # UPD HEADER LENGTH

TCP_LEN = 20 # TCP HEADER LENGTH

''' Elements of the key '''

self.srcMac = ''

self.dstMac = ''

self.frType = ''

self.srcIP = ''

self.dstIP = ''

self.proto = ''

self.opcode = ''

self.port = ''

self.srcPort = ''

self.dstPort = ''

self.srcPortName = ''

self.dstPortName = ''

self.packetSize = 0

self.srcMFG = ''

self.dstMFG = ''

self.dstMacOui ="

self.srcMacOui = ''

self.srcCC = ''

self.dstCC = ''

self.alert = ''

self.lastObservationTime = time.ctime(time.time())

ethernetHeader=packet[0:ETH_LEN]

ethFields =struct.unpack("!6s6sH",ethernetHeader)

self.dstMac = hexlify(ethFields[0]).upper()

self.dstMacOui = self.dstMac[0:6]

self.dstMFG = self.ouiOBJ.lookup(self.dstMacOui)

self.alert = 'Normal'

self.srcMac = hexlify(ethFields[1]).upper()

self.srcMacOui = self.srcMac[0:6]

self.srcMFG = self.ouiOBJ.lookup(self.srcMacOui)

self.fType = ethFields[2]

frameType = self.ethOBJ.lookup(self.fType)

self.frType = frameType

if frameType == "IPV4":

 # Process as IPv4 Packet

ipHeader = packet[ETH_LEN:ETH_LEN+IP_LEN]

 # unpack the ip header fields

ipHeaderTuple = struct.unpack('!BBHHHBBH4s4s' , ipHeader)

 # extract the key ip header fields of interest

 # Field Contents

verLen = ipHeaderTuple[0] # Field 0: Version & Length

TOS = ipHeaderTuple[1] # Field 1: Type of Service

packetLength = ipHeaderTuple[2] # Field 2: Packet Length

protocol = ipHeaderTuple[6] # Field 6: Protocol Number

sourceIP = ipHeaderTuple[8] # Field 8: Source IP

destIP = ipHeaderTuple[9] # Field 9: Destination IP

timeToLive = ipHeaderTuple[5] # Field 5: Time to Live

 # Calculate / Convert extracted values version = verLen >> 4 # Upper Nibble is the version Number

length = verLen & 0x0F # Lower Nibble represents the size

```
ipHdrLength = length * 4 # Calculate the header size in bytes
# covert the srcIP/dstIP to typical dotted notation strings
self.packetSize = packetLength
self.srcIP = socket.inet_ntoa(sourceIP);
self.dstIP = socket.inet_ntoa(destIP);
self.srcCC = self.cc.lookup(self.srcIP, 'IPv4')
self.dstCC = self.cc.lookup(self.dstIP, 'IPv4')
translate = self.traOBJ.lookup(str(protocol))
transProtocol = translate[0]
if transProtocol == 'TCP':
self.proto = "TCP"
stripTCPHeader = packet[ETH_
LEN+ipHdrLength:ipHdrLength+
ETH_LEN+TCP_LEN]
# unpack the TCP Header to obtain the
# source and destination port
tcpHeaderBuffer = struct.unpack('!HHLLBBHHH' , stripTCPHeader)
self.srcPort = tcpHeaderBuffer[0]
self.dstPort = tcpHeaderBuffer[1]
self.srcPortName = self.portOBJ.lookup(self.srcPort, 'TCP')
self.dstPortName = self.portOBJ.lookup(self.dstPort, 'TCP')
elif transProtocol == 'UDP':
self.proto = "UDP"
stripUDPHeader = packet[ETH_LEN+ipHdrLength:ETH_LEN+
ipHdrLength+UDP_LEN]
# unpack the UDP packet and obtain the
# source and destination port
udpHeaderBuffer = struct.unpack('!HHHH' , stripUDPHeader)
self.srcPort = udpHeaderBuffer[0]
self.dstPort = udpHeaderBuffer[1]
```

```
self.srcPortName = self.portOBJ.lookup(self.srcPort, 'UDP')
self.dstPortName = self.portOBJ.lookup(self.dstPort, 'UDP')
elif transProtocol == 'ICMP':
self.proto = "ICMP"
elif transProtocol == 'IGMP':
self.proto = "IGMP"
else:
self.proto = transProtocol
elif frameType == 'ARP':
# Process as IPv4 Packet
arpHeader = packet[ETH_LEN:ETH_LEN+ARP_HDR]
# unpack the ARP header fields
arpHeaderTuple = struct.unpack('!HHBBH' , arpHeader)
ht = arpHeaderTuple[0]
pt = arpHeaderTuple[1]
hal = arpHeaderTuple[2]
pal = arpHeaderTuple[3]
op = arpHeaderTuple[4]
# set packetSize for ARP to zero
self.packetSize = 0
base = ETH_LEN+ARP_HDR
shAddr = hexlify(packet[base:base+hal])
base = base+hal
spAddr = hexlify(packet[base:base+pal])
base = base+pal
thAddr = hexlify(packet[base:base+hal])
base = base+hal
tpAddr = hexlify(packet[base:base+pal])
self.srcIP = shAddr
self.dstIP = thAddr
```

```
        self.proto = str(op)
        elif frameType == "IPV6":
        # Process as IPv6 Packet
        ipHeader = packet[ETH_LEN:ETH_LEN+IPv6_LEN]
        # unpack the ip header fields
        ipv6HeaderTuple = struct.unpack('!IHBBQQQQ' , ipHeader)
        flush = ipv6HeaderTuple[0]
        pLength = ipv6HeaderTuple[1]
        nextHdr = ipv6HeaderTuple[2]
        hopLmt = ipv6HeaderTuple[3]
        srcIP = (ipv6HeaderTuple[4] << 64) | ipv6HeaderTuple[5]
        dstIP = (ipv6HeaderTuple[6] << 64) | ipv6HeaderTuple[7]
        self.packetSize = pLength
        self.srcIP = str(netaddr.IPAddress(srcIP))
        self.dstIP = str(netaddr.IPAddress(dstIP))
        self.srcCC = self.cc.lookup(self.srcIP, 'IPv6')
        self.dstCC = self.cc.lookup(self.dstIP, 'IPv6')
        translate = self.traOBJ.lookup(str(nextHdr))
        transProtocol = translate[0]
        if transProtocol == 'TCP':
        self.proto = "TCP"
        stripTCPHeader = packet[ETH_LEN+IPv6_LEN:ETH_LEN+
IPv6_LEN+TCP_LEN]
        # unpack the TCP Header to obtain the
        # source and destination port
        tcpHeaderBuffer = struct.unpack('!HHLLBBHHH', stripTCPHeader)
        self.srcPort = tcpHeaderBuffer[0]
        self.dstPort = tcpHeaderBuffer[1]
        self.srcPortName = self.portOBJ.lookup(self.srcPort, 'TCP')
        self.dstPortName = self.portOBJ.lookup(self.dstPort, 'TCP')
```

```
elif transProtocol == 'UDP':
self.proto = "UDP"
stripUDPHeader = packet[ETH_LEN+IPv6_LEN:ETH_LEN+
IPv6_LEN+UDP_LEN]
# unpack the UDP packet and obtain the
# source and destination port
udpHeaderBuffer = struct.unpack('!HHHH' , stripUDPHeader)
self.srcPort = udpHeaderBuffer[0]
self.dstPort = udpHeaderBuffer[1]
self.srcPortName = self.portOBJ.lookup(self.srcPort, 'UDP')
self.dstPortName = self.portOBJ.lookup(self.dstPort, 'UDP')
elif transProtocol == 'ICMP':
self.proto = "ICMP"
elif transProtocol == 'IGMP':
self.proto = "IGMP"
else:
self.proto = transProtocol
else:
self.proto = frameType
valueNdx = getOccurrenceValue()
if self.srcIP == '127.0.0.1' and self.dstIP == '127.0.0.1':
''' Ignore this packet '''
return
if self.srcPort <= CORE_PORTS:
''' if srcPort is definately a service port'''
key = (self.srcIP, self.dstIP, self.srcPort,
self.frType, self.proto)
elif self.dstPort <= CORE_PORTS:
''' if dstPort is definately a service port'''
key = (self.srcIP, self.dstIP, self.dstPort, self.frType, self.proto)
```

elif self.srcPort < self.dstPort:

''' Guess that srcPort is server '''

key = (self.srcIP, self.dstIP, self.srcPort, self.frType, self.proto)

else:

''' guess destination port is server'''

key = (self.srcIP, self.dstIP, self.dstPort, self.frType, self.proto)

''' Check Baseline for previously observed key '''

try:

''' if match found, snag the time entries and avg packet size'''

value = self.b[key]

avgPckSize = value[AVGPCKSIZE]

timeList = [value[AM12], value[AM6], value[PM12], value[PM6], value[WKEND]]

newEntry = False

except:

''' Then this is a new observation'''

self.CreateAlertEntry(key, alertDict, "New Observation")

newEntry = True

chk, value = self.ouiOBJ.chkHotlist(self.dstMacOui)

if chk:

self.CreateAlertEntry(key, alertDict, "HotList: "+value)

if self.isNewMAC(self.srcMac, baseMAC):

self.CreateAlertEntry(key, alertDict, "New MAC Address")

if self.isNewMAC(self.dstMac, baseMAC):

self.CreateAlertEntry(key, alertDict, "New MAC Address")

if self.isNewCC(self.srcCC, baseCC):

self.CreateAlertEntry(key, alertDict, "New Country Code")

if self.isNewCC(self.dstCC, baseCC):

self.CreateAlertEntry(key, alertDict, "New Country Code")

''' If this is not a new entry the safe to check pckSize and Times'''

if not newEntry:

if self.isUnusualPckSize(self.packetSize, avgPckSize):

self.CreateAlertEntry(key, alertDict, "Unusual Packet Size")

if self.isUnusualTime(timeList):

self.CreateAlertEntry(key, alertDict, "Unusual Packet Time")

Packet Monitor Supporting Methods

def isUnusualPckSize(self, pSize, avgSize):

if float(pSize) < float(avgSize.70):*

return True

*if float(pSize) < float(avgSize*1.30):*

return True

return False

def isNewMAC(self, mac, b):

if mac == 'Unknown' or mac == '':

return False

if not mac in b:

return True

else:

return False

def isNewCC(self,cc, b):

if cc == 'Unknown' or cc == '':

return False

if not cc in b:

return True

else:

return False

def isUnusualTime(self, occList):

occ = getOccurrenceValue()

if occList[occ] == 0:

return True

else:

return False

def CreateAlertEntry(self, key, alertDict, alertType):

try:

''' See if the alert already exists '''

value = alertDict[key]

''' if yes, then bump the occurrence count'''

cnt = value[1] + 1

alertDict[key] = [alertType, cnt, self.lastObservationTime, self.packetSize, self.srcCC, self.dstCC, self. srcMac, self.dstMac, self.srcMFG, self. dstMFG, self.srcPort, self.dstPort, self. srcPortName, self.dstPortName]

except:

''' Othewise create a new alert entry'''

alertDict[key] = [alertType, 1, self. lastObservationTime, self.packetSize, self.srcCC, self.dstCC, self.srcMac, self.dstMac, self. srcMFG, self.dstMFG,self.srcPort, self. dstPort, self.srcPortName, self. dstPortName]

3.4. Налаштування Raspberry Pi для захоплення мережевого трафіку

Налаштувавши Raspberry Pi, який включає в себе базовий реєстратор, датчик та звіти, можна перейти до технології захоплення мережевого трафіку. Для початку, необхідно оновити версію Python до останньої версії. Далі, необхідно скопіювати інсталяційні файли, доступні у python-forensics.org/piSensor до вибраної папки на Pi. Для даної тестової установки файл було переміщено в папку під назвою TEST прямо на робочому столі Pi. Також відкритий доступ ще до декількох папок та елементів. А саме:

- папка RPT - Звіти та базові показники записуються до цієї папки датчиком Raspberry Pi;
- piSensorV3 - це компільований додаток датчика Python;

- lookup.db містить різні таблиці пошуку портів, протоколів, виробників MAC -адрес та типів Ethernet;
- файли geoIPv6 та geoIPv4 використовуються для зіставлення IP -адрес із країнами;
- hotlist.txt містить список цікавих портів.

Можна додати дисплей PaPirus ePaper до свого Pi, як показано на рис.3.5. Це відображатиме інформацію в режимі реального часу безпосередньо на Pi. Якщо PaPirus не встановлено, датчик буде працювати нормально, і весь дисплей буде надаватися лише через графічний інтерфейс.

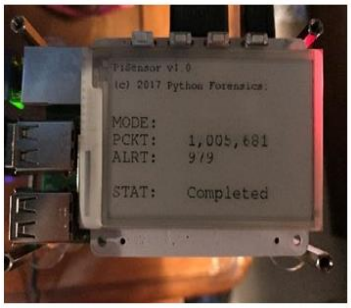

Рис.3.5. Raspberry Pi з дисплеєм PaPirus ePaper

Підключення Raspberry Pi. Наступним кроком є підключення Raspberry Pi до мережі, яку необхідно моніторити та контролювати.

Конфігурація комутатора для захоплення пакетів. Більшість сучасних мережевих інфраструктур та комутаторів підтримують дзеркальне відображення портів за допомогою аналізатора комутованих портів (SPAN) або віддаленого комутованого порту

Аналізатор (RSPAN). Можна використовувати 8-портовий гігабітний комутатор TP-LINK. Він порівняно недорогий, надійний та простий в налаштуванні пристрій (рис.3.6).

Рис.3.6. 8-портовий гігабітний «розумний комутатор» TP-LINK

Простота комутатора базується на програмному застосунку «Easy Smart Configuration Utility», що показана на рис.3.7., який входить до складу комутатора. Утиліта конфігурації дозволяє налаштовувати всі функції, доступні на TL-SG108E. З метою захоплення всього мережевого трафіку, що проходить через комутатор, необхідно налаштувати вибір моніторингу.

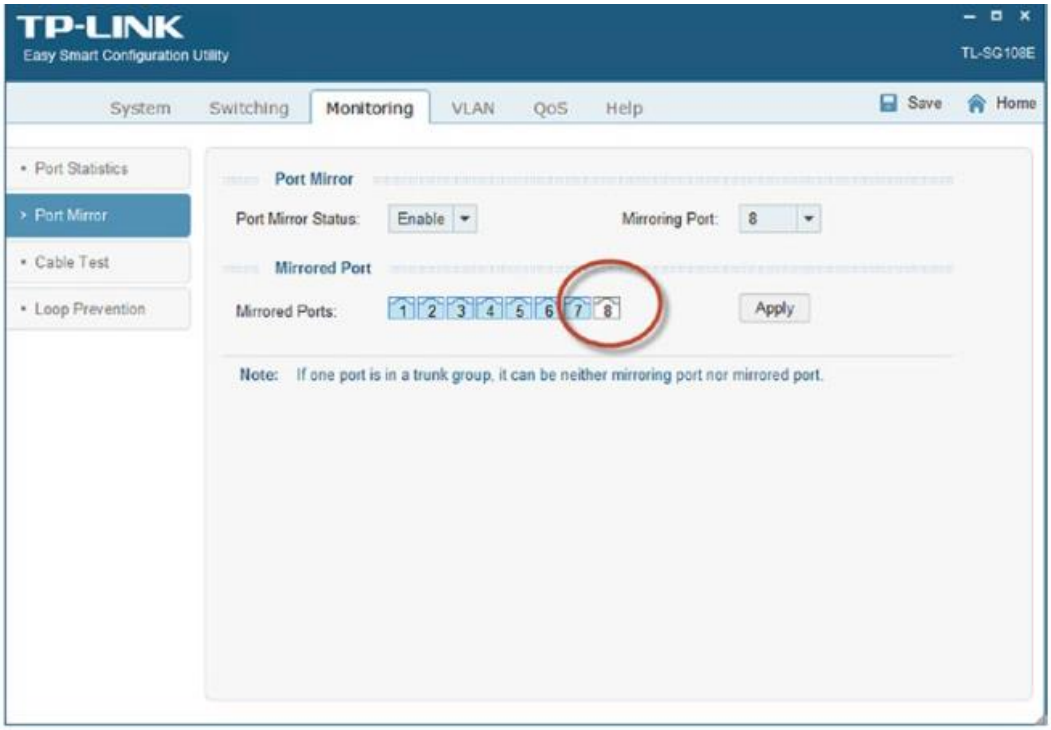

Рис.3.7. Утиліта конфігурації на комутаторі

На рис.3.7. зображено екран конфігурації для моніторингу портів. У цьому прикладі було встановлено порт 8 як порт моніторингу, а порти 1–7 для моніторингу. Це означає, що весь трафік, що надходить у порти 1-7 або з них, буде доступний для моніторингу на порту 8. Далі просто підключається порт Ethernet на Raspberry Pi до порту 8 на комутаторі, як показано на рис.3.8.

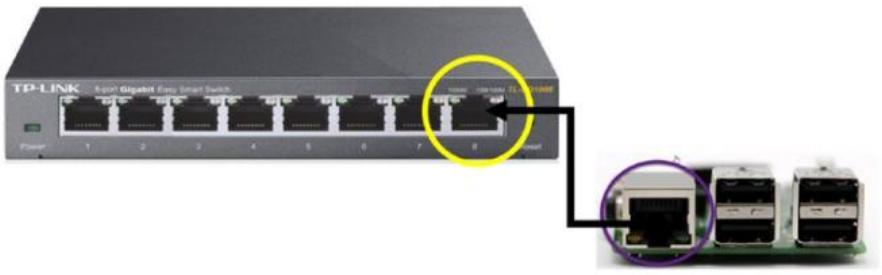

Рис.3.8. Підключення датчика Pi до порту моніторингу TP-LINK

Запуск програми Python. Тепер, коли Raspberry Pi налаштований і підключений до відповідного мережевого комутатора з монітором або портом SPAN, можна почати запуск програми-датчика.

PiSensorV3-це компільована версія сенсорної програми на основі Python. Можна запустити інтерпретатор Python і вказати основний сценарій Python piSensorV3.py. Для цього потрібно завантажити сценарії Python. piSensorV3.py - це сценарій Python 2.7 і він не працюватиме в середовищах Python 3.x. Однак програма piSensorV3 не покладається на базову інсталяцію Python. sudo python piSensorV3.py

Щоб зробити сценарій Python виконуваним, можна використати один із існуючих методів перетворення сценаріїв Python у більш традиційні виконувані файли [26].

Pyinstaller - це чудовий продукт для перетворення сценаріїв Python у виконувані файли. Щоб виконати piSensorV3, необхідно відкрити вікно терміналу на Raspberry Pi. Простий спосіб зробити це-натиснути значок на верхній панелі інструментів, як показано на рис.3.9.

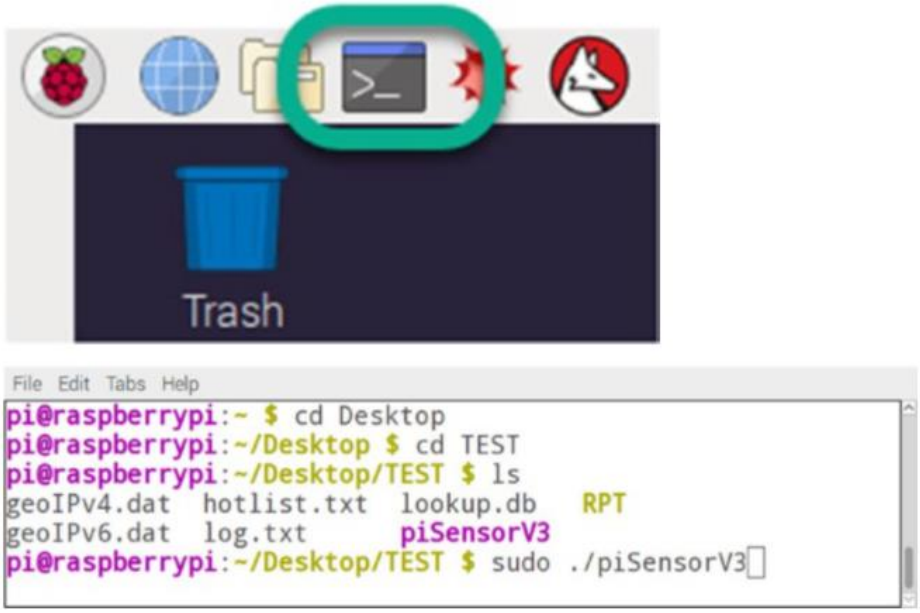

Рис.3.9. Відкриття вікна терміналу

Це запустить термінальну програму, що дозволяє вводити команди командного рядка. Щоб запустити piSensorV3, необхідно перейти до папки, де знаходиться скопіовані файли.

Запуск виконувального файлу. Даний файл запускається з поточного робочого каталогу, і як sudo. Це потрібно, оскільки piSensorV3 потребує привілеїв для переходу мережевого адаптера в безладний режим. Після цього буде запущено додаток piSensorV3 з графічним інтерфейсом, як показано на рис.3.10.

Рис.3.10. Запуск додатку piSensorV3

Створення базової лінії. Наступним кроком в операції є створення базової лінії мережі, на якій буде здійснюватися контроль та моніторинг. Далі це буде використовуватися датчиком для моніторингу поведінки пристрою в режимі датчика. Тим не менш, багато чого можна дізнатися про мережу, створивши також базову лінію.

Перший крок у створенні базового сценарію - це вказати папку, де будуть записані результати, що спостерігаються, разом із налаштуванням звітів. Для цього обирається папку RPT для зберігання результатів, як показано на рис.3.11. та рис.3.12.

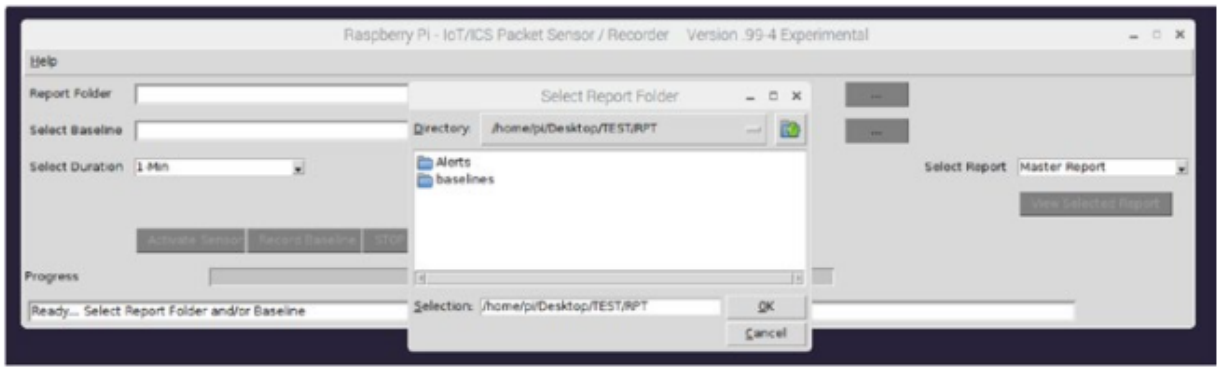

Рис.3.11. Повідомити про вибір папки

Обрано тривалість 1 день. Тривалість запису залежить від поведінки, яку необхідно контролювати. У більшості випадків, краще встановлювати тривалість в один повний тиждень, щоб охоплювати всі операції.

Рис.3.12. Звіт та тривалість вибрано

Можна помітити, що кнопка запису базової лінії тепер доступна, оскільки було успішно вказано папку звіту та тривалість. Рис.3.13. показує запис базового прогресу, тоді як на рис.3.14. зображено індикації прогресу на дисплеї PaPirus.

Рис.3.13. Базовий прогрес запису

Кнопка запису базової лінії більше не доступна, але з’явилася кнопка СТОП. У будь-який момент її можна натиснути, а далі - продовжити запис або скасувати його [27]. При скасуванні, записані результати будуть збережені у базовій лінії, а отримані проміжні звіти будуть сформовані.

Рис.3.14. Відображення прогресу запису PaPirus

Після завершення запису повідомлення про стан змінюється на «Завершено» та відображає загальну кількість оброблених зв’язків разом із кількістю унікальних спостережень (рис.3.15.).

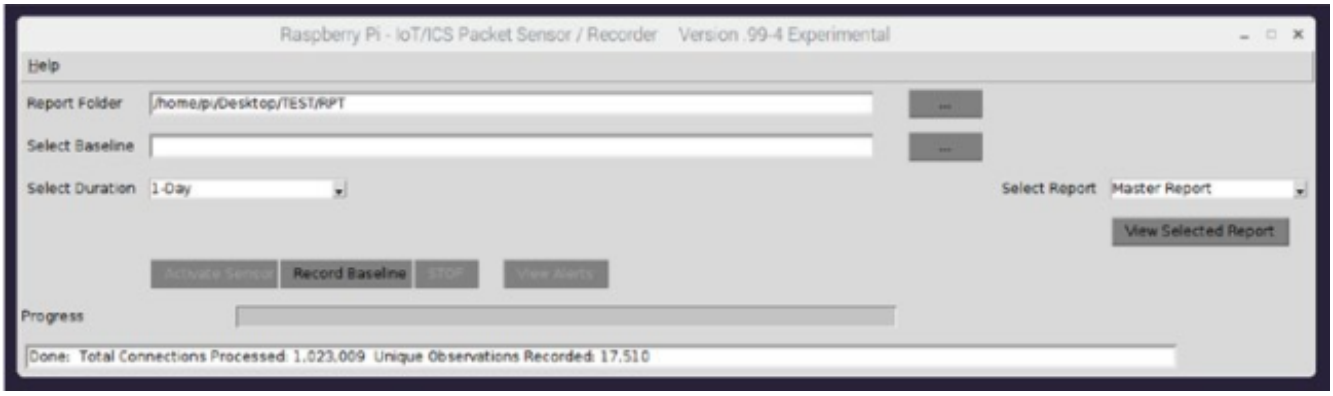

Рис.3.15. Запис базової лінії завершено

Це ключ до методології скорочення даних. Записуються з'єднання з використанням однієї IP-адреси джерела, адреси та порту. Однак замість того, щоб зберігати кожне з'єднання, кількість з'єднань такого типу, що відбуваються, записується для кожного дня тижня та години дня. Ця інформація використовується датчиком для виявлення незвичної поведінки. Це дозволяє також економити ресурси на Рі лише шляхом запису унікальної (аномальної) поведінки.

Є ще кілька важливих результатів операції запису. По -перше, кнопка перегляду звітів тепер активується, коли генеруються звіти за період спостереження. На рис.3.16. зображено вибір наявних звітів.

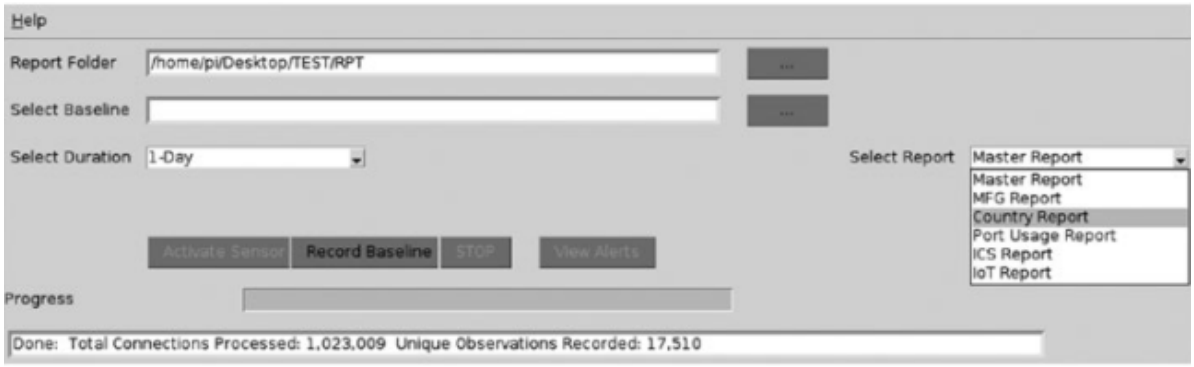

Рис.3.16. Звіт про вибір

Доступні звіти включають наступне:

1. Майстер. Цей звіт містить усі записані спостереження з деталями кожного запису, як показано на рис.3.17.

Alert	Normal		
Src-IP	**Dst-IP**	**Protocol**	**Frame-Type**
192.168.86.36	192.168.86.1	UDP	IPV4
Src-Port	**Src-Port Name**	**Dst-Port**	**Dst-Port name**
60465	Unknown	53	Domain Name Server
Src-MAC	**Src-MFG Name**	**Dst-MAC**	**Dst-MFG Name**
14B31F07219E	US: Dell Inc.	E4F042BF5185	Unknown
Src Country	**Dst Country**		
Internal	**Internal**		
Packet-Size	77		
Morning	0		
Afternoon	2		
Evening	0		
PreDawn	0		
Weekend	0		
Total	2		

Alert	Normal		
Src-IP	**Dst-IP**	**Protocol**	**Frame-Type**
192.168.86.44	239.255.255.250	UDP	IPV4
Src-Port	**Src-Port Name**	**Dst-Port**	**Dst-Port name**
44868	Unknown	1900	UPnP SSDP
Src-MAC	**Src-MFG Name**	**Dst-MAC**	**Dst-MFG Name**
000C8A979D36	US: Bose Corporation	01005E7FFFFA	Unknown
Src Country	**Dst Country**		
Internal	Unknown		
Packet-Size	0		
Morning	1432		
Afternoon	1440		
Evening	1437		
PreDawn	1440		
Weekend	0		
Total	5749		

Рис.3.17. Приклад змісту основного звіту

CN: B-Link Electronic Limited	48022A4A2DAE	48022a4a2dae
CN: B-Link Electronic Limited	48022A4A2DAE	fe80::4a02:2aff:fe4a:2dae
CN: TP-LINK TECHNOLOGIES CO. LTD.	50C7BF4CF802	50c7bf4cf802
CN: Wistron Infocomm (Zhongshan) Corporation	F80F4142D110	192.168.86.38
CN: Wistron Infocomm (Zhongshan) Corporation	F80F4142D110	f80f4142d110
CN: Wistron Infocomm (Zhongshan) Corporation	F80F4142D110	fe80::6d9a:7ea0:5026:d87e
CN: Zhejiang shenghui lighting co. Ltd	B0CE18189451	192.168.86.28
CN: Zhejiang shenghui lighting co. Ltd	B0CE18189451	b0ce18189451
CN: Zhejiang shenghui lighting co. Ltd	B0CE18189451	fe80::b2ce:18ff:fe18:9451
KR: Samsung Electronics Co. Ltd	900628FB11BB	900628fb11bb
KR: Samsung Electronics Co. Ltd	900628FB11BB	fe80::9206:28ff:fefb:11bb
TW: ASUSTek COMPUTER INC.	0015F25CF374	0015f25cf374
TW: ASUSTek COMPUTER INC.	0015F25CF374	192.168.86.47
TW: ASUSTek COMPUTER INC.	0015F25CF374	fe80::8103:ed8e:c718:82ad
US: Amazon Technologies Inc.	40B4CDB12685	0.0.0.0
US: Amazon Technologies Inc.	40B4CDB12685	192.168.86.27
US: Amazon Technologies Inc.	40B4CDB12685	40b4cdb12685
US: Amazon Technologies Inc.	40B4CDB12685	fe80::42b4:cdff:feb1:2685
US: Amazon Technologies Inc.	747548914299	0.0.0.0
US: Amazon Technologies Inc.	747548914299	192.168.86.39
US: Amazon Technologies Inc.	747548914299	747548914299
US: Amazon Technologies Inc.	747548914299	fe80::7675:48ff:fe91:4299

Рис.3.18. Приклад змісту основного звіту з деталями

2. Звіт виробника пристрою. Цей звіт містить спостереження за кожним виробником пристрою разом з відповідними MAC та IP-адресами. Це забезпечує детальне відстеження відомих і, можливо, невідомих пристроїв, розташованих у мережі. Будь-який пристрій, який не спостерігався протягом певного періоду часу або ж новий, відслідковується ретельніше. Звіт на рис.3.19 для скороченого прикладу.

Country	Count
	357228
Canada	29
Czech Republic	1439
Germany	12
Hong Kong	400
Internal	1085886
Ireland	804
Japan	50
Luxembourg	17
Netherlands	832
Singapore	420
Sweden	16
United Kingdom	834
United States	762871

Рис.3.19. Витяг звіту виробника

3. Звіт про країну. Так само, як і звіт виробника, дані впорядковані за країнами спостереження. У звіті наведено кількість підключень до систем у цільовій країні. Знову ж таки, під час фази датчика будь-які з'єднання з країною, які не спостерігалися протягом певного періоду, генерують попередження.

4. Звіт про використання порту. Цей звіт впорядковує дані за спостереженнями за з'єднаннями портів. Звіт містить кожен номер використовуваного порту та відповідне ім’я, а також унікальний вихідний та цільовий IP адреси, тип кадру та відповідний протокол, який був використаний. На рис.3.20. зображено уривок зі звіту про використання порту [28].

Port Usage Observations

Port	PortName	Src IP	Dst IP	Frame	Protocol
53	Domain Name Server	192.168.86.1	192.168.86.36	IPV4	UDP
53	Domain Name Server	192.168.86.1	192.168.86.47	IPV4	UDP
53	Domain Name Server	192.168.86.36	192.168.86.1	IPV4	UDP
53	Domain Name Server	192.168.86.47	192.168.86.1	IPV4	UDP
53	Domain Name Server	8.8.8.8	192.168.86.23	IPV4	UDP
53	Domain Name Server	8.8.8.8	192.168.86.25	IPV4	UDP
68	Bootstrap Protocol Client	192.168.86.36	192.168.86.1	IPV4	UDP
68	Bootstrap Protocol Client	192.168.86.37	192.168.86.1	IPV4	UDP
68	Bootstrap Protocol Client	192.168.86.38	192.168.86.1	IPV4	UDP
68	Bootstrap Protocol Client	192.168.86.47	192.168.86.1	IPV4	UDP

Рис.3.20. Звіт про використання порту

5. Звіти про використання портів ICS та звіти про використання портів IoT. Ці звіти додатково фільтрують використання портів до єдиних портів, які зазвичай використовуються пристроями ICS або IoT. Важливо зауважити, що деякі звіти про порти також можуть використовувати не ICS/IoT.

ICS Report

Possible ICS Observations

Port	Count	Port Name	MAC Address	SRC MFG	DST MFG
443	1	HTTP protocol over TLS/SSL	E4F042BF5185	CN: Wistron Infocomm (Zhongshan) Corporation	Unknown
443	2	HTTP protocol over TLS/SSL	E4F042BF5185	CN: Wistron Infocomm (Zhongshan) Corporation	Unknown
443	8	HTTP protocol over TLS/SSL	E4F042BF5185	CN: Wistron Infocomm (Zhongshan) Corporation	Unknown
443	1	HTTP protocol over TLS/SSL	E4F042BF5185	TW: ASUSTek COMPUTER INC.	Unknown

Рис.3.21. Зразок звіту ICS

Таким чином, звіти називаються «Можливі ICS» та «Можливе використання портів Інтернету речей». Під час роботи датчика будь-які спостереження ICS або IoT, які не існували протягом періоду запису, генерують попередження.

IoT Report

Possible IoT Observations

Observed	Src IP	Dst IP	
84	192.168.86.44	224.0.0.251	
Src Port	Src/Port Name	Dst Port\strong>	Dst/Port Name
5353	Unknown	5353	Unknown
Src MAC	Src MFG	Dst MAC	Dst MFG
000C8A979D36	US Robot Corporation	01005E0000FB	
Observed	Src IP	Dst IP	
38	192.168.86.47	224.0.0.251	
Src Port	Src/Port Name	Dst Port\strong>	Dst/Port Name
5353	Unknown	5353	Unknown
Src MAC	Src MFG	Dst MAC	Dst MFG
0015F25CF374	TW ASUSTek COMPUTER INC	01005E0000FB	
Observed	Src IP	Dst IP	
106	fe80 8100 ndle c718 82ad	ff02 fb	
Src Port	Src/Port Name	Dst Port\strong>	Dst/Port Name
5353	Unknown	5353	Unknown
Src MAC	Src MFG	Dst MAC	Dst MFG
0015F25CF374	TW ASUSTek COMPUTER INC	3333000000FB	

Рис.3.22. Зразок звіту IoT

Тепер, коли є записана базова лінія, можна використовувати цю базову лінію для активації датчика, вибравши конкретну базову лінію, як показано на рис.3.23.

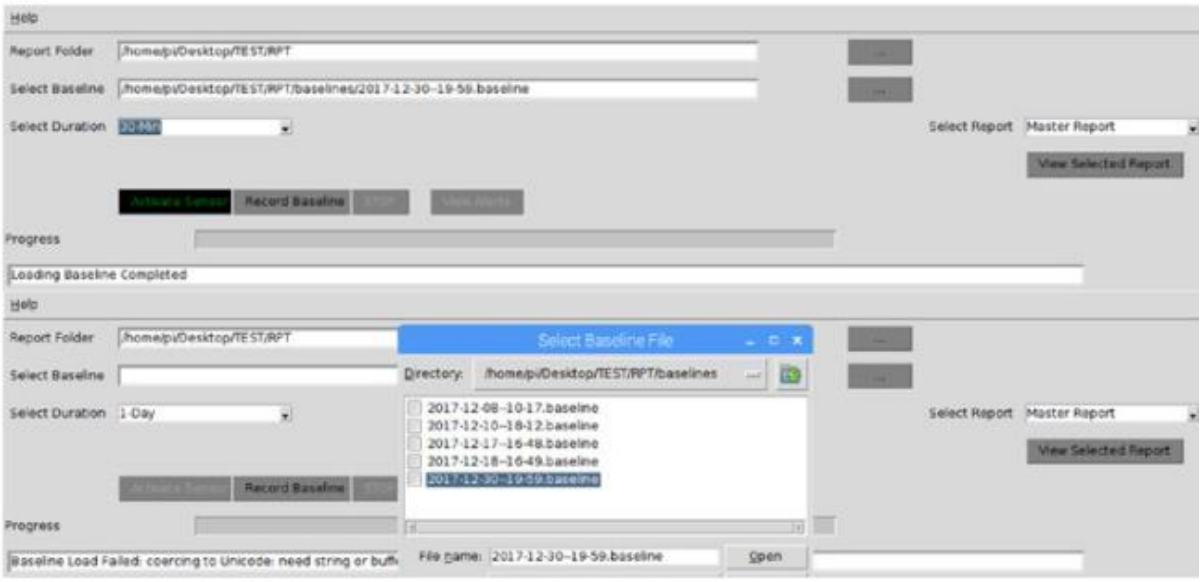

Рис.3.23. Вибір базової лінії

Папка звіту все ще потрібна, а кнопка активації датчика буде недоступною, доки обидва звіту було вибрано папку та базову лінію. Папка звіту є необхідною, оскільки будь-які сповіщення, що генеруються датчиком, будуть зберігатися на один рівень нижче папки звіту у підпапці під назвою ALERTS [29].

Нарешті, потрібно вибрати тривалість роботи датчика та натиснути кнопку активації датчика; потім починається процес моніторингу будь-яких відхилень від записаної базової лінії (рис.3.24).

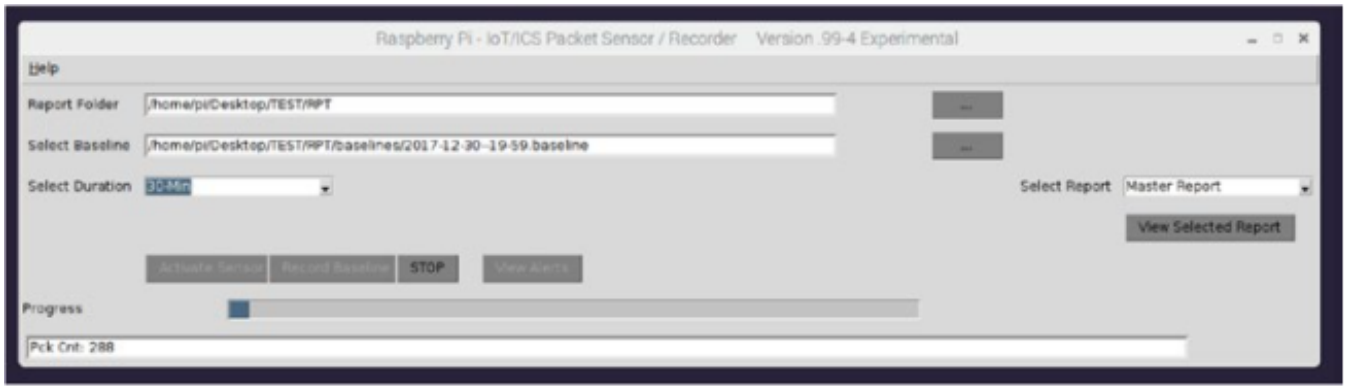

Рис.3.24. Активація датчика

Після завершення роботи датчика можна побачити кількість оброблених пакетів разом із кількістю генерованих оповіщень. Крім того, доступна кнопка перегляду сповіщень, що дозволяє переглядати будь-які сповіщення, створені датчиком. За цей короткий час роботи датчика (30 хвилин) датчик обробив 22 295 з'єднань і виявив 353 аномалії або відхилення від спостережуваної базової лінії (рис.3.25).

Рис.3.25. Завершення роботи датчика

Тепер можна вивчити сформовані попередження, щоб переглянути аномалії, які були виявлені датчиком. Звіт, представлений на рис.3.26 містить скорочений висновок.

Alert	Unusual Packet Time	**Alerts-Observed**	7
Last Observed	Sun Dec 31 11:03:17 2017		
SrcIP	**DstIP**	**Frame-Type**	**Protocol**
209.133.212.170	192.168.86.21	IPV4	TCP
SrcPort	**SrcPortName**	**DstPort**	**DstPortName**
7024	Unknown	39375	Unknown
SrcMAC	**DstMAC**	**PckSize**	
E4F042BF5185	AECA05FA0552	40	
SrcCountry	**DstCountry**	**SrcMFG**	**DstMFG**
United States	Internal	Unknown	Unknown
Alert	Unusual Packet Time	**Alerts-Observed**	8
Last Observed	Sun Dec 31 11:04:16 2017		
SrcIP	**DstIP**	**Frame-Type**	**Protocol**
192.168.86.33	224.0.0.251	IPV4	IGMP
SrcPort	**SrcPortName**	**DstPort**	**DstPortName**
NA	NA	NA	NA
SrcMAC	**DstMAC**	**PckSize**	
B827EBC183AE	01005E0000FB	32	
SrcCountry	**DstCountry**	**SrcMFG**	**DstMFG**
Internal	Unknown	US: Raspberry Pi Foundation	Unknown
Alert	New Observation	**Alerts-Observed**	1
Last Observed	Sun Dec 31 10:50:14 2017		
SrcIP	**DstIP**	**Frame-Type**	**Protocol**
192.168.86.33	50.116.52.97	IPV4	UDP
SrcPort	**SrcPortName**	**DstPort**	**DstPortName**
123	Network Time Protocol	123	Network Time Protocol
SrcMAC	**DstMAC**	**PckSize**	
B827EBC183AE	E4F042BF5185	76	
SrcCountry	**DstCountry**	**SrcMFG**	**DstMFG**
Internal	United States	US: Raspberry Pi Foundation	Unknown

Рис.3.26. Зразок звіту оповіщення

Як можна побачити в уривку, звіт включав незвичайні звіти про час пакетів разом із новим спостереженням. На основі аналізу пакетів, жодне з них не є надто серйозним. Набагато довший запис (тиждень) створив би спостереження, які, ймовірно, містили б обидва ці.

3.5. Оновлення програмного забезпечення Raspberry Pi для захоплення мережевого трафіку

Поряд з Raspberry Pi 4 Model B+, було здійснено кілька важливих оновлень програмного забезпечення датчика. Вони включають вибір NIC та фільтрацію MAC-адрес, як показано на рис.3.27 та позначені A та B відповідно.

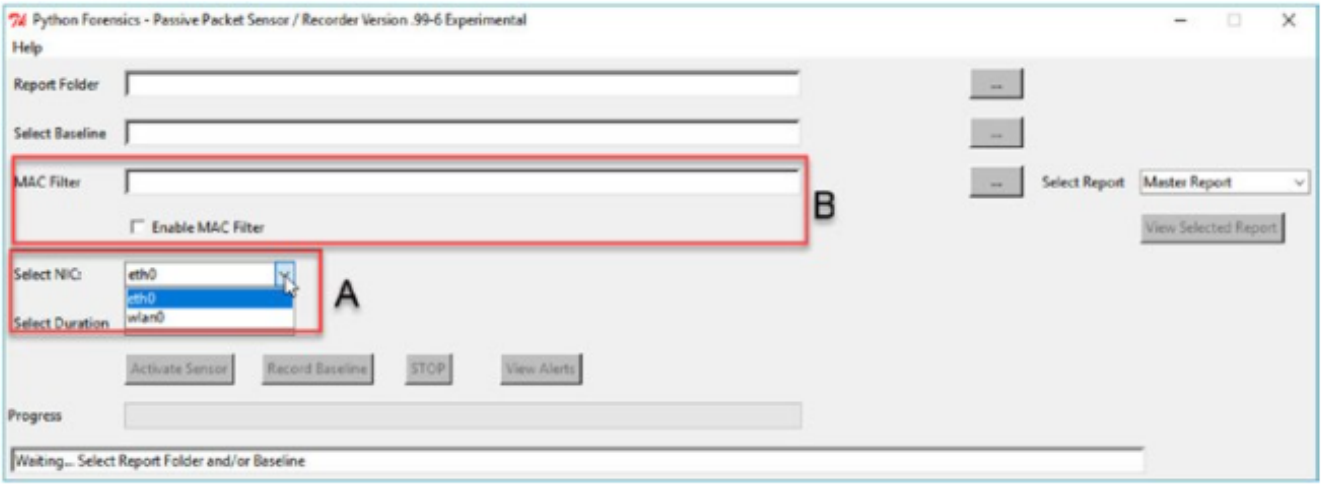

Рис.3.27. Оновлення датчиків: (A) Вибір NIC; (B) Фільтрація MAC-адрес

Визначити доступні інтерфейси на Raspberry Pi досить просто. Каталог/sys/class/net містить імена доступних інтерфейсів. Для реалізації цілей можна надати список можливих інтерфейсів, а найголовніше-вибір безпроводового інтерфейсу на додаток до стандартного порту Ethernet. Програма для моніторингу конкретного пристрою представлено далі:

```
try:
nicList = os.listdir('/sys/class/net')
nicList.sort()
nicTuple = tuple(nicList)
except:
nicTuple= tuple(['eth0'])
self.ethPort['values'] = nicTuple
self.ethPort.current(0)
self.ethPort.grid(row=5, column=1, padx=5, pady=10, sticky='w')
```

Якщо було обрано безпроводову локальну мережу (wlan0), спочатку потрібно підключитися до потрібної безпроводової мережі для моніторингу. На Raspberry Pi можна вибрати, підключитися та увійти до потрібного безпроводового інтерфейсу за допомогою значка у верхньому правому куті (рис.3.28).

Рис.3.28. Вибір безпроводового підключення Raspberry Pi

3.6. Фільтрація MAC-адрес

Друге доповнення - це можливість моніторингу, запису та активації датчика для націлювання на певні MAC-адреси. В межах промислового контролю або розділених середовищ БСМ чи Інтернету речей досить поширений ретельний моніторинг критично важливих датчиків та активів. Цей вибір використовує список MAC-адрес, наданих у вигляді текстового файлу. Рис.3.29 та рис.3.30 демонструють вибір файлу фільтра MAC та прапорець, який дозволяє фільтрацію MAC.

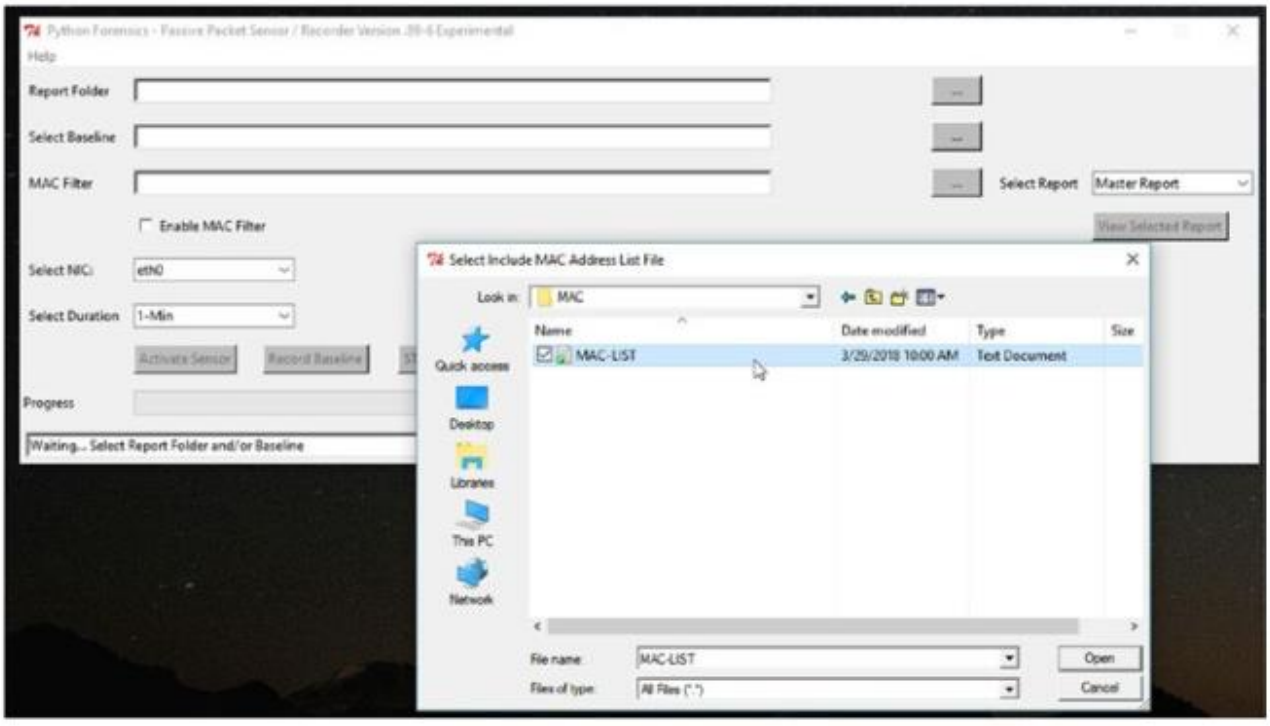

Рис.3.29. Вибір списку фільтрації MAC

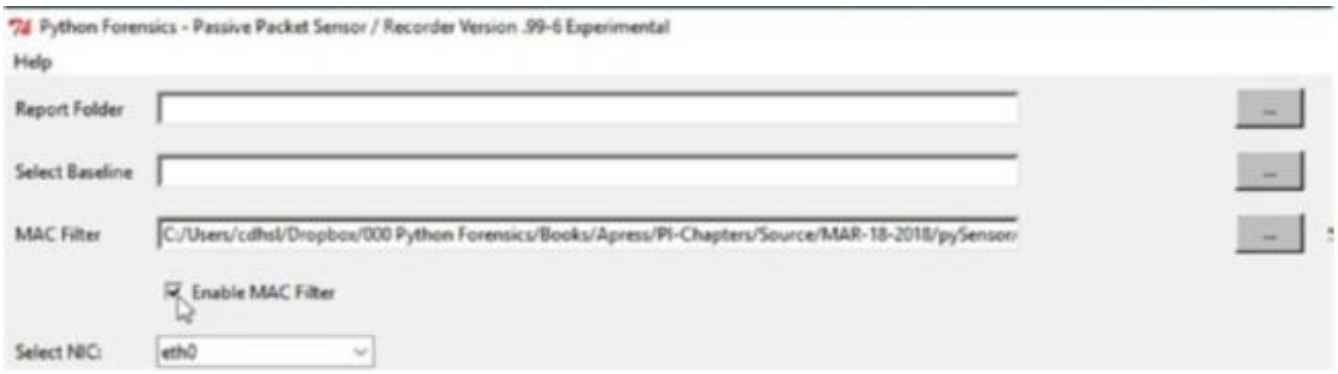

Рис.3.30. Увімкнення фільтра MAC

Текстовий файл MAC-LIST містить простий список MAC-адрес, по одному на рядок, як показано на рис.3.31.

Рис.3.31. Зразок текстового файлу MAC-LIST

IP-адреси для пристроїв динамічно призначаються DHCP, якщо вони не визначені статично. Тому використання MAC-адрес (якими також можна маніпулювати, але для цього потрібні цілеспрямовані дії) забезпечує кращі параметри фільтрації. Під час роботи датчика реєструватимуться лише пакети з MAC-адресами джерела або адреси, зазначеними у списку. Це дозволяє спростити аналіз звітів, таких як використання порту та країна, що дозволяє перевірити вхідний та вихідний трафік із певних пристроїв [30].

Фільтрація MAC-адрес обробляється лише кількома рядками коду. По-перше, створюється список MAC для фільтрації, коли надається файл фільтрації

MAC, і фільтрація MAC включена. Поважаючи вибір фільтрів користувача представлено далі:

```
self.fileSelection = tkFileDialog.askopenfilename
(initialdir = "./",
title = "Select Include MAC Address List File")
self.IncludeFile['text'] = self.fileSelection
if self.fileSelection:
self.macList = []
self.macEnable = True
with open(self.fileSelection) as ips:
for eachLine in ips:
self.macList.append(eachLine.strip())
else:
self.macEnable = False
```

Цей метод забезпечує легку фільтрацію MAC-адрес під час вилучення пакетів.

```
ethernetHeader=packet[0:ETH_LEN]
ethFields =struct.unpack("!6s6sH",ethernetHeader)
# Extract DST MAC, SRC MAC and Frame Type
self.dstMac = hexlify(ethFields[0]).upper()
self.srcMac = hexlify(ethFields[1]).upper()
# Check if MAC Filtering is on
if self.macFilterEnable and self.macFilterSet:
if not (self.dstMac in self.macFilter) and
not (self.srcMac in self.macFilter):
# Filter this packet
return
```

Висновки до третього розділу

Представлено огляд пристрою Raspberry Pi, а також детальний огляд багатьох елементів коду, які підтримують: огляд дизайну, вивчення підходу до графічного інтерфейсу, інтеграція дисплея PaPirus ePaper, детальна інформація про базовий метод запису, деталі методів датчиків, деталі методів формування звіту, метод для зберігання та завантаження отриманих записаних базових ліній

Описано датчик Raspberry Pi, а саме: огляд підключення датчика до активної мережі, запис базової лінії, створення та вивчення звітів, створених під час запису базового сценарію, вибір записаної базової лінії, створеної для використання під час фази датчика, активація датчика на основі конкретної записаної базової лінії, вивчення сповіщень, що генеруються датчиком.

Підкреслено, що Raspberry Pi дає можливість не лише розширити діапазон інтелектуальної складової при використанні сенсорних датчиків, але і додати безпеку, як повноцінну складову безпроводових сенсорних мереж. Зокрема шляхом відстеження будь-якого мережевого трафіку, а також з можливістю націлення на конкретні MAC-адреси.

Визначено, що можливість моніторингу, запису та активації датчика для націлення на певні MAC-адреси є важливою в межах промислового контролю або розділених середовищ БСМ чи Інтернету речей. Адже досить поширений ретельний моніторинг критично важливих датчиків та активів з використанням MAC-адрес забезпечує кращі параметри фільтрації. Під час роботи датчика реєструватимуться лише пакети з MAC-адресами джерела або адреси, зазначеними у списку. Це дозволяє спростити аналіз звітів, таких як використання порту та країни, що дозволяє перевірити вхідний та вихідний трафік із певних пристроїв.Фільтрація MAC-адрес на Raspberry Pi обробляється лише кількома рядками коду і дає можливості уникнути вручань в безпроводову сенсорну мережу з боку зловмисників.

ВИСНОВКИ

В магістерській роботі отримано наступні наукові та науково-практичні результати:

1. Проаналізовано безпроводові сенсорні мережі, як підклас безпроводових мереж, та зазначено, що через обмеженість ресурсів БСМ вони породжують цілий новий набір атак.

2. Представлено узагальнену таксономію атак на БСМ, та приведено аналіз різних видів атак на такі мережі та способи боротьби з ними.

3. Зазначено, що на сьогодні одним із найбільш перспективних напрямків БСМ являється реалізація IoT, а також проведено дослідження вразливості сенсорних датчиків, сенсорів та «розумних» пристроїв, що мають відношення до мережі IoT за допомогою пошукової системи Shodan.

4. Зроблено висновок, що більшість пристроїв БСМ та IoT вразливі та незахищені, адже: або використовують паролі за замовчуванням, або не підключають мережеві фільтри, тому їх легко можна відшукати в глобальному просторі Інтернет.

5. Описано Raspberry Pi, операційну систему Raspbian та мову програмування Python, яка використовується сьогодні в БСМ.

6. Представлено сенсорний скрипт, який буде використовувати попередньо записану базову лінію (створену PacketRecorder) та повідомлятиме про аномалії між базовою лінією та оточуючим середовищем.

7. Представлено огляд пристрою Raspberry Pi, а також детальний огляд багатьох елементів коду, які підтримують: огляд дизайну, вивчення підходу до графічного інтерфейсу, інтеграція дисплея PaPirus ePaper, детальна інформація про базовий метод запису, деталі методів датчиків, деталі методів формування звіту, метод для зберігання та завантаження отриманих записаних базових ліній

8. Описано датчик Raspberry Pi, а саме: огляд підключення датчика до активної мережі, запис базової лінії, створення та вивчення звітів, створених під

час запису базового сценарію, вибір записаної базової лінії, створеної для використання під час фази датчика, активація датчика на основі конкретної записаної базової лінії, вивчення сповіщень, що генеруються датчиком.

9. Підкреслено, що Raspberry Pi дає можливість не лише розширити діапазон інтелектуальної складової при використанні сенсорних датчиків, але і додати безпеку, як повноцінну складову безпроводових сенсорних мереж. Зокрема шляхом відстеження будь-якого мережевого трафіку, а також з можливістю націлення на конкретні MAC-адреси.

10. Визначено, що можливість моніторингу, запису та активації датчика для націлення на певні MAC-адреси є важливою в межах промислового контролю або розділених середовищ БСМ чи Інтернету речей. Адже досить поширений ретельний моніторинг критично важливих датчиків та активів з використанням MAC-адрес забезпечує кращі параметри фільтрації. Під час роботи датчика реєструватимуться лише пакети з MAC-адресами джерела або адреси, зазначеними у списку. Це дозволяє спростити аналіз звітів, таких як використання порту та країни, що дозволяє перевірити вхідний та вихідний трафік із певних пристроїв.Фільтрація MAC-адрес на Raspberry Pi обробляється лише кількома рядками коду і дає можливості уникнути вручань в безпроводову сенсорну мережу з боку зловмисників.