МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Західноукраїнський національний університет
Факультет комп'ютерних інформаційних технологій
Кафедра кібербезпеки

ХОМИЦЬКИЙ Андрій Андрійович

Алгоритми виявлення атак на Інтернет речей з використанням технології приманок/ Attack detection algorithms for the Internet of Things based on honeypot

спеціальність: 125 – Кібербезпека

освітньо-професійна програма –Кібербезпека

Кваліфікаційна робота

Виконав студент групи КБм -21
А.А. Хомицький

Науковий керівник
д.т.н., професор В.В. Яцків

Кваліфікаційну роботу
допущено до захисту:

«____» _____ 2022 р.

Завідувач кафедри
_____ **В.В.Яцків**

ТЕРНОПІЛЬ - 2022

Факультет комп'ютерних інформаційних технологій
Кафедра кібербезпеки
Освітній ступінь «магістр»
спеціальність: 125 - Кібербезпека
освітньо-професійна програма –Кібербезпека

ЗАТВЕРДЖУЮ
Завідувач кафедри

_____ В.В.Яцків
____" _____ 2021 року

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

ХОМИЦЬКИЙ Андрій Андрійович
(прізвище, ім'я, по батькові)

1. Тема кваліфікаційної роботи:
Алгоритми виявлення атак на Інтернет речей з використанням технології приманок/ Attack detection algorithms for the Internet of Things based on honeypot
керівник роботи д.т.н., професор В.В. Яцків

затверджені наказом по університету від 31 грудня 2021 року № 606

2. Строк подання студентом закінченої випускної кваліфікаційної роботи 16 листопада 2022 року.

3. Вихідні дані до кваліфікаційної роботи: завдання на випускну кваліфікаційну роботу студента, наукові статті, технічна література.

4. Основні питання, які потрібно розробити:
– провести аналіз підходів до алгоритми виявлення атак з використанням технології приманки;
– розробити модель алгоритму виявлення атак;
– розробити алгоритм роботи системи виявлення атаки за допоиогою приманок;
– провести аналіз векторів атак на систему інтернет речей.

5. Перелік графічного матеріалу у роботі.
Характеристика комунікаційних протоколів IoT.
Концепція моделювання платформи IoT.
Найпопулярніші User Agent за всіма запитами, взятими з журналів проксі.
Методи запобігання атакам на платформи IoT.

Механізми безпеки комунікаційних протоколів IoT.

6. Консультанти розділів кваліфікаційної роботи

	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання 11 жовтня 2021 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строки виконання етапів кваліфікаційної роботи	Примітка
1	Аналіз предметної області	12.2021 р. – 03.2022 р.	
2	Структура системи виявлення атак на Інтернет речі	03.2022 р. – 05.2022 р.	
3	Аналіз результатів дослідження	05.2022 р. – 11.2022 р.	

Студент _____ Хомицький А.А.
(підпис)

Керівник роботи _____ д.т.н., професор В.В. Яцків
(підпис)

АНОТОЦІЯ

Кваліфікаційна робота на тему “Алгоритми виявлення атак на Інтернет речей з використанням технології приманок” зі спеціальності 125 – кібербезпека написана обсягом 73 сторінок і містить 23 ілюстрацій, 18 таблиць та 28 джерел за переліком посилань.

Мета роботи – підвищення ефективності алгоритмів виявлення атак на Інтернет речей з використанням технології приманок.

Методи дослідження. Для розв’язання поставлених задач у даній кваліфікаційній роботі використано: методи статистики, методи кореляції, методи інтерпретації отриманих результатів.

За результатами роботи, IoTPot продемонстрував здатність виявляти проблеми безпеки на платформах IoT, на основі чого сформульовано методи запобігання атакам на платформи IoT:

1) Шифрування критичного корисного навантаження HTTP REST може бути зашифровано.

2) Замість звичайного тексту, якщо SSDP-пакети notify зашифровані, розташування мосту не буде розкрито.

3) Важливо звернути увагу на безпеку XMPP. Набір даних, згенерований honeypot, можна використовувати для створення сигнатур IDS (наприклад, для двигунів SNORT або Suricata), щоб виявити подібні атаки на платформи Philips Hue та інші загальні REST- сканування пристроїв Інтернету речей.

Ключові слова: ІНТЕРНЕТ РЕЧІ, IoTPot, HONEYPOT, ПРИМАНКА.

ANNOTATION

The master's thesis on the topic "Algorithms for detecting attacks on the Internet of Things using decoy technology" from the specialty 125 - cyber security is written in the volume of 73 pages and contains 23 illustrations, 18 tables and 28 sources according to the list of references.

The purpose of the work is to determine effective algorithms for detecting attacks on the Internet of Things using decoy technology.

Research methods: statistics; correlation, interpretation of the obtained results.

According to the results of the work, IoTPot demonstrated the ability to detect security problems on IoT platforms, based on which methods were formulated to prevent attacks on IoT platforms:

1) Encryption of critical HTTP REST payload can be encrypted. For example, the whitelist of the Philips Hue bridge must be encrypted to prevent it from being hijacked by hackers.

2) Instead of plain text, if SSDP notify packets are encrypted, the bridge location will not be revealed.

3) It is important to pay attention to the security of XMPP. The data set generated by the honeypot can be used to generate IDS signatures (eg for SNORT or Suricata engines) to detect similar attacks on Philips Hue platforms and other general REST scans of IoT devices.

Keywords: INTERNET OF THINGS, IoTPot, HONEYPOT.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

IoT - Інтернет речей

IIoT - промисловий Інтернет речей

ЗВВ - загальні вразливості та викриття

ГМВРД - глобальна мережа великого радіусу дії

ІПП - інтерфейси прикладного програмування

МП - миттєві повідомлення

CVE - Common Vulnerabilities and Exposures

ASN - Autonomous System Number

PHD - Philips Hue devices

HIH - High Interaction Honeypot

LIH - Low Interaction Honeypot

MIH – Medium Interaction Honeypot

УММ - уніфікована мова моделювання

РП - розумні пристрої

ВСТУП

Актуальність теми роботи зумовлена тим, що концепція Інтернету речей (IoT) дуже популярна. Зростає кількість пристроїв IoT з безпрецедентною швидкістю. Такі великі компанії, як Amazon, IBM, Samsung тощо, розробили проекти, пов'язані з IoT, і деякі продукти (платформи, мобільні програми тощо) також були запущені. Навіть Ikea відносно недавно запустила свою платформу розумного освітлення Tradfri в США [1]. Також багато нових компаній зосереджуються на технологіях IoT. Термін «Інтернет речей» є впізнаваним і привертає до себе все більшу увагу. Існує прогноз, що до 2030 року десятки мільярдів одиниць IoT будуть розгорнуті по всьому світу, збираючи велику кількість різноманітних даних [2].

Інтернет речей включає багато областей і багато видів технологій, таких як вбудовані пристрої, комунікаційні технології, сенсорні мережі, Інтернет-протоколи тощо. Були опубліковані опитування, які стосуються різних аспектів технологій IoT.

Система IoT повинна мати можливість обробляти різнорідні об'єкти через Інтернет. Це є критичною вимогою архітектури IoT. Загалом архітектура IoT також може бути багаторівневою, однак визначення та класифікація кожного рівня не є стандартними та різняться в різних джерелах. Дослідження [5] узагальнює усі запропоновані моделі, зробивши висновок, що серед всіх запропонованих моделей базовою моделлю є 3-рівнева архітектура, яка складається з прикладного, мережевого та презентаційного рівнів. Проте також пропонується і використовується 5-рівнева модель. П'ять рівнів: рівень об'єктів; рівень абстракції об'єктів; рівень керування послугами; рівень додатків і бізнес-рівень.

Незалежно від кількості рівнів, архітектури IoT у деяких аспектах схожі на традиційні пристрої, пов'язані з Інтернетом. Це означає, що вони спілкуються один з одним через протоколи на кожному рівні.

Мета і завдання дослідження. Метою роботи є підвищення ефективності алгоритмів виявлення атак на Інтернет речей з використанням технології приманок.

Досягнення визначеної мети передбачає вирішення таких завдань:

– провести аналіз так на Інтернет речі;

– дослідити методи захисту і аналізу атак;

– модель приманок і типи їх робіт;

– використати приманки для збору даних;

– провести аналіз загроз безпеки інтернет речей;

– розробити загальну структуру приманок для захисту Інтернет речей;

Об’єкт дослідження – процес виявлення атак на Інтернет речей;

Предмет дослідження – методи та алгоритми виявлення атак на Інтернет речей з використанням технології приманок.

Методи досліджень. Для розв’язання поставлених задач у даній кваліфікаційній роботі використано: методи статичного збору даних, методи кореляції даних, методи інтерпретація отриманих результатів.

Наукова новизна одержаних результатів. Удосконалено алгоритми виявлення атак на інтернет речі за допомогою методу приманок.

Практичне значення отриманих результатів. Розроблено структуру системи виявлення атак на систему інтернет речей за допомогою приманок.

Публікації та апробація КР.

1. Хомицький А.А. Аналіз існуючих приманок для виявлення атак на інтернет речей. Матеріали наукової конференції «Автоматизація та комп’ютерно-інтегровані технології» (АКІТ – 2022) Тернопіль, 2022. С. 113-114.

2. Хомицький А.А. Аналіз по категорії приманок для виявлення атак на інтернет речей. Матеріали наукової конференції «Кібербезпека та компютерно-інтегровані технології» (КБКІТ-2022) Тернопіль, 2022. С. 51-52

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Аналіз вразливостей Інтернет речей

Пристрої Інтернету речей (IoT) набирають популярності у повсякденному житті, а також у певних сферах, таких як медичні установи. Їх використання може здійснюватися через локальну мережу або Інтернет. Коли IoT-пристрої, такі як SmartTV, консолі, мультимедійні пристрої, холодильники, медичні пристрої тощо, доступні через Інтернет, вони можуть бути більш уразливими, оскільки механізми безпеки IoT-протоколів і програм ще не розроблені як звичайні системи (наприклад, ПК).

Взявши приклад промислового Інтернету речей (IIoT), згідно з [11], опитування брало участь 403 IT-фахівця, які несуть певну відповідальність за цифрову безпеку як значну частину своєї роботи. Було визначено, наскільки їхні організації готові протистояти загрозам, пов’язаним із IIoT, у 2017 році.

Результати опитування такі:

По-перше, коли їх запитали про наступний рік, 96% респондентів відповіли, що очікують збільшення атак на безпеку IIoT.

По-друге, 51% відповіли, що вони не готові до зловмисних атак, які певним чином використовують або зловживають промисловим Інтернетом речей.

Нарешті, 64% учасників уже визнали, що їхнім організаціям необхідно захищатися від таких атак.

Як згадувалося раніше, оскільки речі підключені до Інтернету, традиційні проблеми безпеки (наприклад, вірусні зараження, хакери тощо) також можуть впливати на зв’язок IoT. Крім того, через такі особливості, як неоднорідність, енергозбереження та автоматизація, Інтернет речей стикається з більшими проблемами безпеки. Якщо з пристроями чи програмним забезпеченням виникли проблеми, це може мати серйозні наслідки. Згідно з [12], було знайдено 1,6 мільйона вразливих речей,

підключених до Інтернету, до яких входять без пароля або входять за допомогою облікових даних за замовчуванням.

У 2016 році з'явилися деякі з найбільших розподілених атак типу «відмова в обслуговуванні» (DDos), які, імовірно, досягали приблизно 1 Тбіт/с трафіку та здійснювалися сотнями тисяч скомпрометованих пристроїв IoT. У цьому випадку відомо, що зараження пристроїв IoT відбувається так званим шкідливим програмним забезпеченням Mirai [13], який ставить вразливі пристрої IoT під контроль ботнету та який, як вважають, контролює близько мільйона скомпрометовані пристрої IoT. Подібні інциденти безпеки можуть бути лише початком у контексті IoT.

Наразі немає стандартів, які б узгоджено визначали використання протоколів на платформах IoT. На різних рівнях можна використовувати кілька протоколів, деякі з яких або знаходяться на стадії розробки, або фактично розроблені для інших або більш загальних цілей. Крім того, наявні опитування показують наближення до використання XMPP, MQTT, CoAP, REST API для широкого розгортання платформ IoT.

Дослідники провели ряд опитувань про безпеку IoT з різних точок зору. Автори [14] аналізують існуючі протоколи та механізми захисту комунікацій в IoT, а також відкриті дослідницькі питання. Вони також аналізують, як існуючі підходи забезпечують фундаментальні вимоги безпеки та захищають комунікації в IoT, а також відкриті виклики та стратегії для майбутніх дослідницьких робіт у цій галузі. Аналіз зосереджений на протоколах зв'язку IoT, розділених на чотири основні частини: фізичний (PHY) і рівень керування доступом до середовища (MAC), мережевий рівень, маршрутизація в IoT (RPL) і прикладний рівень. В [15], також розроблено безпеку програми для різних профілів безпеки. Роботи [28; 42] надають огляд проблем безпеки та формулюють кілька порад щодо покращення безпеки.

Підводячи підсумок, результати дослідження безпеки виявили проблеми, які включають:

1) Проблеми безпеки рівня сприйняття:

- ключові проблеми підходу початкового завантаження в IoT (розповсюдження ключів, керування ключами тощо [16; 17];

- криптографія з відкритим ключем проти приватного ключові операції [17].

2) Проблеми безпеки мережевого рівня:

- мережні DoS-атаки на вузли сенсорної мережі [18];

- вузли шлюзу сенсорної мережі можуть вийти з-під контролю [18];

- перешкоди, втручання та обрив сигналу мережі;

- безпека проблем в наскрізній комунікації [19].

3) Проблеми безпеки рівня додатків:

- обмеження безпеки CoAP [14].

Можна виділити шість популярних технологій для безпеки IoT на основі аналізу Форрестера:

1) Безпека мережі.

2) Аутентифікація IoT.

3) Шифрування IoT.

4) IoT PKI.

5) Аналітика безпеки IoT.

6) Безпека IoT API.

Можна відзначити, що більшість опитувань зосереджено на питаннях шифрування та автентифікації на мережевому рівні. Проблема безпеки прикладного рівня також є серйозною, і цьому потрібно приділити особливу увагу. На жаль, хоча існує багато опитувань про протоколи IoT, недостатньо досліджень безпеки протоколів додатків у контексті IoT, що означає прогалину, яка може включати вразливості, які не можна ігнорувати. Наприклад, інструмент атаки XMPPloit [20] показує наявність наявних уразливостей XMPP, якими також можна скористатися в контексті IoT. Таким чином, необхідно досліджувати вразливості та атаки протоколів додатків IoT.

Вразливість може бути не тільки в самих протоколах, а й у їх реалізації. Можуть бути проблеми при реалізації сервера, клієнта. Загальні вразливості та

викриття (ЗВВ), або CVE записує відомі слабкі місця. Зловмисники також можуть використовувати слабкі місця, які не були виправлені, або атакувати реалізації, які не були виправлені вчасно. Тому необхідно дізнатися про вразливі реалізації та відповідні напрямки атак.

Існують різні підходи до проведення аналізу вразливостей в IoT, які описано нижче.

Аналіз протоколу. Деякі дослідники, як [21] вирішив проаналізувати сам протокол. Через неоднорідний характер IoT використання протоколів є складним і різним. Існують проблеми з тестуванням безпеки та аналізом протоколів IoT. Крім того, технологія IoT пов’язана не лише з протоколами, розгортання та інтеграція між протоколами та технологіями також слід брати до уваги під час аналізу вразливостей. Також можна використовувати такі інструменти, як Proverif.

Тестування фаззингу. Надаючи недійсні, неочікувані або випадкові дані як вхідні дані, зловмисник може отримати корисні дані. Фаззинг може перевіряти, чи достатньо безпечний протокол або реалізація.

Honeypot. Honeypot - це механізм безпеки комп’ютера, який може відстежувати дії зловмисників, завдяки чому можна підвищити рівень безпеки та вирішити деякі проблеми безпеки Інтернету речей. В основному його метою є атака у контрольованому середовищі. Впроваджуючи honeypot з протоколами IoT, зловмисники намагатимуться атакувати його, таким чином honeypot записуватиме інформацію зловмисника (наприклад, IP-адресу, інструменти атаки). Така інформація допоможе вивчити проблеми безпеки IoT.

Порівнюючи ці три способи аналізу вразливості, у поточній роботі використовується підхід застосування технології honeypot як методу аналізу та вивчення проблем безпеки зв’язку IoT. Виходячи з поточної ситуації, IoT Honeypot, IoTPot буде розроблено та впроваджено для вивчення проблем безпеки IoT на прикладному рівні. Це буде симуляція платформи IoT замість

єдиних протоколів зв’язку, що дозволяє відповідати поточній ситуації з продуктами IoT і глибше досліджувати безпеку проблеми [22].

1.2 Аналіз протоколів додатків Інтернет речей

Комунікаційні протоколи визначають стандартний спосіб встановлення значущої взаємодії між двома або більше об’єктами, що забезпечує дійсну очікувану поведінку всіх залучених сторін. Далі буде представлено протоколи, які в основному використовуються на прикладному рівні IoT, з подальшим аналізом реалізації застосування XMPP та MQTT.

Оскільки немає чітких стандартів для всіх компонентів Інтернету речей, технології, які зараз використовуються платформами IoT, мають різні характеристики. По суті, можна використовувати будь-яку технологію, яка відповідає вимогам підключення. У нинішній ситуації різні компанії мають власну архітектуру IoT.

Підводячи підсумок, можна сказати, що існує три основних способи налаштувати мережу IoT для зв’язку пристроїв один з одним та з Інтернетом:

1) Пристрої можуть підключатися один до одного за допомогою різноманітних технологій внутрішнього радіо, таких як Wi-Fi, Zigbee або Bluetooth. Для підключення до Інтернету знадобиться концентратор або шлюз. Він збиратиме інформацію з пристроїв, а потім передаватиме кінцевим користувачам через інші протоколи прикладного рівня. Цей шаблон часто використовується в розумних будинках або медичних установах.

2) Пристрої також можуть використовувати стільникові технології, такі як 2G/3G/LTE або 5G. Потім їх можна з'єднати через шлюз. Ці технології підходять для зовнішніх пристроїв, що працюють на великій відстані. Але ці технології споживають більше енергії. Таким чином, для IoT просуваються нові міжміські технології. Глобальна мережа великого радіусу дії (ГМВРД), або - LoRaWAN і вузькосмуговий Інтернет речей (NB-IoT) є двома

домінуючими технологіями, які випускаються різними постачальниками. Цей метод часто використовується у великих сценаріях, таких як розумне місто.

3) Замість розгортання шлюзу пристрої також можна підключити безпосередньо до Інтернету. Наприклад, за допомогою протоколу MQTT датчики можуть публікувати інформацію брокеру MQTT. Потім користувачі можуть отримати інформацію, підключившись до брокера.

Незалежно від типу бездротової технології, яка використовується, дані кінцевих пристроїв можуть бути доступні в Інтернеті, і цього можна досягти двома способами:

1) Надсилання інформації до власної веб-служби або інтерфейсів прикладного програмування (ІПП), або -API, доступних з Інтернету.

2) Використання хмари. API, або хмара будуть базою даних для зберігання та обробки, проміжним вузлом між пристроями та кінцевим користувачем і API, щоб дати кінцевому користувачеві змогу відстежувати та контролювати пристрої віддалено. Наприклад, продукт Philips Hue використовує Zigbee для підключення кінцевих пристроїв (лампа, датчик) і «Bridge», або «Міст» (концентратор). Потім міст підключається до сервера Philips. Кінцевий користувач може контролювати та отримувати дані через сервер за допомогою REST API.

Багато протоколів додатків було визнано придатними для зв’язку IoT. Ось список пов’язаних прикладних протоколів:

MQTT (Message Queue Telemetry Transport). MQTT - це протокол обміну повідомленнями, випущений IBM у 1999 році. Він легкий, відкритий, простий і розроблений таким чином, щоб його було легко реалізувати. Він використовує шаблон публікації/підписки, який працює поверх TCP, тому клієнт не потребує оновлень. Крім того, він має низькі накладні витрати порівняно з іншими протоколами прикладного рівня на основі TCP . Ці функції роблять його легким і відповідають вимогам IoT. Він також забезпечує трирівневу якість обслуговування (QoS).

XMPP (Extensible Messaging and Presence Protocol). XMPP - це протокол зв’язку, який забезпечує базові функції обміну миттєвими повідомленнями (МП), або - IM і статусу присутності. Він був стандартизований IETF і широко використовувався. Останнім часом він привернув більше уваги завдяки своїм функціям, які підходять для IoT. XMPP є розширюваним, оскільки він дозволяє специфікувати протоколи розширення XMPP для розширення функціональності. Було випущено нові засоби для підтримки IoT. Він також використовує шаблон публікації/підписки. Сьогодні багато продуктів підтримують XMPP для налаштування.

AMQP (Advanced Message Queuing Protocol). AMQP – це відкритий стандартний протокол прикладного рівня, який виник у фінансовій галузі. Він використовує комунікаційну модель публікації/підписки та підтримує надійний зв’язок, використовуючи примітивні типи, включаючи доставку принаймні один раз, принаймні один раз і точно один раз. Повідомлялося, що середовище AMQP із 2000 користувачами на п’яти континентах може обробляти 300 мільйонів повідомлень на день [24].

CoAP (Constrained Application Protocol). CoAP - це протокол прикладного рівня, створений робочою групою Constrained RESTful Environment (CoRE) [25; 26]. Він був розроблений для пристроїв з обмеженим доступом. Це протокол запиту/відповіді, який працює через UDP, щоб зберегти легкість і зменшити вимоги до пропускної здатності. Але він підтримує QoS і використовує простий механізм повторної передачі Stop-and-Wait для підтверджених повідомлень [8]. CoAP використовує простий формат для кодування повідомлень [5].

UPnP (Universal Plug and Play). UPnP - це набір мережевих протоколів, які використовуються для виявлення мережевих пристроїв для обміну даними, спілкування та розваг. Це протокол розподіленої відкритої архітектури, заснований на встановлених стандартах, таких як набір протоколів Інтернету (TCP/IP), HTTP, XML і SOAP. Він використовується в Інтернеті речей, оскільки може динамічно забезпечувати надійне з’єднання між пристроями від

різних постачальників. Таким чином, він підходить для домашньої автоматизації.

JMS (Java Message Service). JMS - це інтерфейс прикладного програмування, який використовується для створення асинхронного зв'язку між програмами або розподіленими системами. Він широко використовується і підтримується підприємствами. Більш ніж два клієнти можуть обмінюватися повідомленнями надійно та асинхронно [2].

REST (Representational State Transfer). REST - це архітектурний стиль замість єдиного протоколу, розроблений Роєм Томасом Філдінгом у 2000 році. Він широко використовується в платформах Machine-to-Machine (M2M) та IoT. Він надає орієнтовану на ресурси систему обміну повідомленнями, де ресурси доступні через URI та запит GET, тоді як вхідні дані приймаються через PUT [2, 8].

DDS (Data Distribution Service). DDS - це проміжний протокол від Object Management Group (OMG) [27], що знаходиться між операційною системою та програмами. Він використовує шаблон публікації-підписки для обміну даними, який є масштабованим, з низькою затримкою, надійним, високопродуктивним і сумісним. Його можна використовувати для програм IoT.

Кожен із цих протоколів має унікальні характеристики та може використовуватися в різних сценаріях. Вони також можуть співпрацювати один з одним, будучи реалізованими в різних частинах системи IoT. Порівняння були зроблено в багатьох опитуваннях. Автор також аргументує їх придатність для IoT з точки зору надійності, безпеки та енергоспоживання [5], і дає більш повний короткий опис більшості відповідних протоколів. Це дає змогу дізнатися, як різні протоколи можуть працювати разом без необхідності переглядати RFC.

Можна взяти розумний дім як приклад, щоб розробити різні способи використання різних протоколів: управління розумним освітленням може використовувати XMPP; для електропостачанням - DDS, можна

використовувати для моніторингу двигунів на електростанції; MQTT можна використовувати для перевірки силового кабелю; AMQP може передати всі дані споживання електроенергії електричними пристроями вдома в хмару або домашній шлюз для аналізу.

Протокол XMPP. XMPP - це протокол обміну миттєвими повідомленнями, спочатку розроблений спільнотою Jabber з відкритим кодом. Раніше він називався Jabber і працює через TCP. Він використовує так званий ідентифікатор Jabber (JID), який є унікальним ідентифікатором для кожного пристрою. Цей JID складається з трьох частин: вузла, домену та ресурсу, і його формат схожий на адресу електронної пошти, тобто " node@domain /resource". Вузол і домен розділяються знаком «@» і косою рискою після домену, якщо є ім’я ресурсу.

XMPP використовує модель «клієнт-сервер-сервер-клієнт», як показано на рисунку 1.1. Сервери підключаються один до одного. Клієнти під різними серверами не підключаються безпосередньо. Коли клієнти під різними серверами хочуть підключитися один до одного, усі комунікації здійснюються через проміжний сервер. За замовчуванням клієнт підключається до сервера XMPP за допомогою порту TCP/5222. Сервери XMPP підключаються один до одного через порт TCP/5269.

З’єднання між клієнтом і сервером XMPP спирається на потік XML (eXtensibleMarkup Language) строфи. Комунікація починається початковим тегом XML <stream> і завершується тегом XML </stream>.

Строфа - це структурований елемент, який можна надсилати протягом активності потоку. Немає обмежень на кількість надсилання строф. Строфа XMPP має різні атрибути: п’ять загальних атрибутів: «to», «from», «tipe», «id», «xml:lang» [28]. Він складається з трьох компонентів [5, 28]:

1) «Presence» (присутність).

Строфа присутності використовується для сповіщення та оновлення статусу для сутностей, які на неї підписалися.

2) «Message» (повідомлення).

Строфа повідомлення містить поля теми та основного тексту, які заповнюються назвою та вмістом повідомлення. Це розглядається як механізм проштовхування, коли одна сутність передає інформацію іншим сутностям. Сервер, який отримує повідомлення, потім надсилає його бажаному одержувачу згідно з атрибутом "to" в повідомленні.

3) «Iq» (інформація/запит).

Iq розглядається як механізм «запит-відповідь». Одна сутність може надіслати як "iq" із типом "get" або "set", щоб зробити запит. Одержувач може відповісти " iq result" або " iq error ".

XMPP добре працює в домашній автоматизації. Багато розробників вважають за краще використовувати XMPP як протокол для своїх проектів домашньої автоматизації [29]. Він забезпечує майже всі функції, необхідні системі домашньої автоматизації для обміну повідомленнями, наприклад однорангове з'єднання, маршрутизація до користувача, автентифікація користувача, публікація, підписка тощо. Існуючі продукти, такі як Philips Hue, також можна ідеально поєднувати з XMPP таким чином, щоб він був сумісний з іншими продуктами, що полегшують системи домашньої автоматизації.

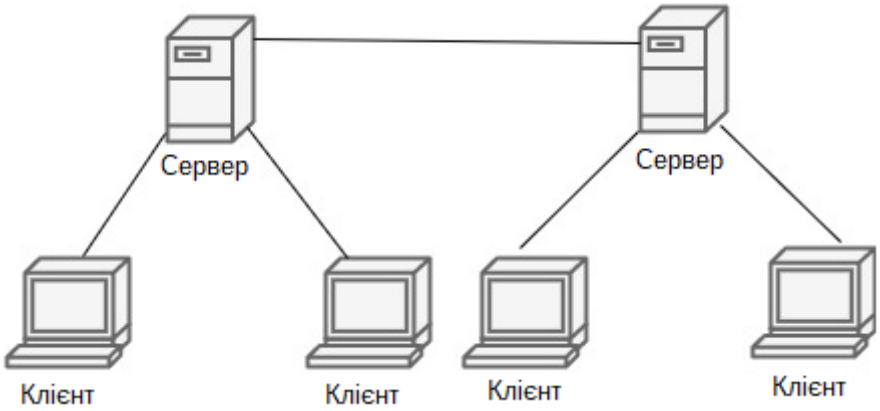

Рисунок 1.1 – Комунікація XMPP

Протокол MQTT. MQTT - це протокол передачі повідомлень для публікації/підписки, який є відкритим і простим у реалізації. Він вважається придатним для M2M та IoT.

Елементи зв’язку MQTT можна розділити на клієнта та брокера, який є сервером. Клієнтів можна розділити на два типи відповідно до ролей у комунікації: видавець, який надсилає певні повідомлення з певною темою, і підписник, який отримує певні повідомлення з певною темою. Один клієнт може бути як видавцем, так і передплатником у різних темах. Брокер – це той, хто посередині, отримує кожне повідомлення та підключається до клієнтів. Його завдання - фільтрувати повідомлення та розподіляти їх відповідно до теми. На рис. 1.2 показано приклад того, як брокер MQTT і клієнти працюють разом для обміну повідомленнями. Таким чином, відправнику та одержувачу не потрібно бути «онлайн» одночасно. MQTT також підтримує три рівні QoS: «at most once», «at lease once» і «exactly once».

Протокол REST. REST (від англ. Representational State Transfer - «передача репрезентативного стану») широко використовується в платформах Machine-to-Machine (M2M) та IoT для передачі повідомлень і команд керування між інтерфейсом користувача та системою пристроїв. Таким чином, дані в системі пристрою можуть бути доступні іншим Інтернет-додаткам.

Багато продуктів IoT використовують REST API для передачі інформації між пристроями та користувачами. До таких відносяться:

1) Philips Hue використовує REST API, щоб користувачі надсилали команди на міст Hue. Потім міст керує освітленням відповідно до команд або надсилає інформацію назад.

2) Xively також підтримує HTTP для публікації повідомлень.

3) Комутатор Wemo від Belkin підтримує REST API, щоб користувачі могли надсилати запит комутатору через URL-адресу.

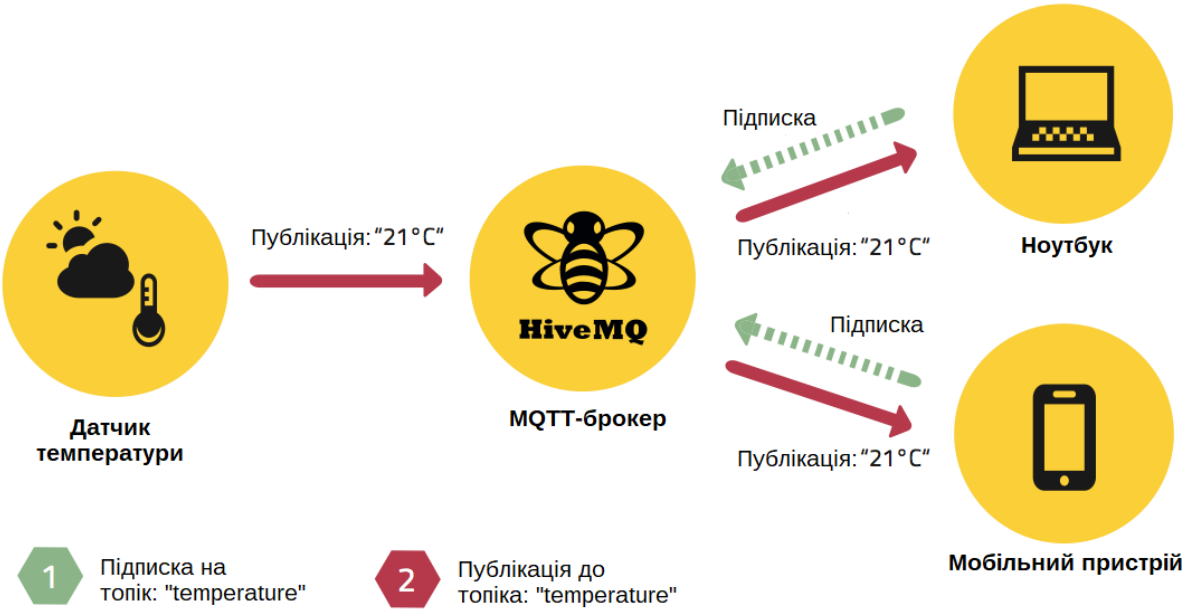

Рисунок 1.2 - Комунікація MQTT

1.3 Безпека протоколів додатків Інтернет речей

Зараз пристрої та платформи IoT набувають широкого поширення. Люди хотіли б купувати інтелектуальні пристрої від різних компаній і будувати індивідуальну систему розумного будинку. Ця персональна система розумного дому зазвичай містить панель керування пристроєм, деякі розумні пристрої та пристрої з доступом до Інтернету (хаб, міст тощо). На такій платформі IoT користувачі можуть інтегрувати разом пристрої різних компаній.

Одна з поширених структур платформи Інтернету речей показана на рис. 1.3. Він включає API для роботи та зв'язку з пристроями, протоколи миттєвого зв'язку для побудови зв'язку між користувачами та API, а клієнти (користувачі) можуть отримувати доступ як через API, так і через протоколи миттєвого зв'язку. Серед багатьох способів створення налаштованої платформи IoT протокол XMPP є одним із найвідоміших протоколів зв'язку, які використовуються. Є багато прикладів такої архітектури, як-от Wemo,

Philips Hue, IoTfy тощо. Прикладом цього фреймворку є система Philips Hue, поєднана користувачами з XMPP.

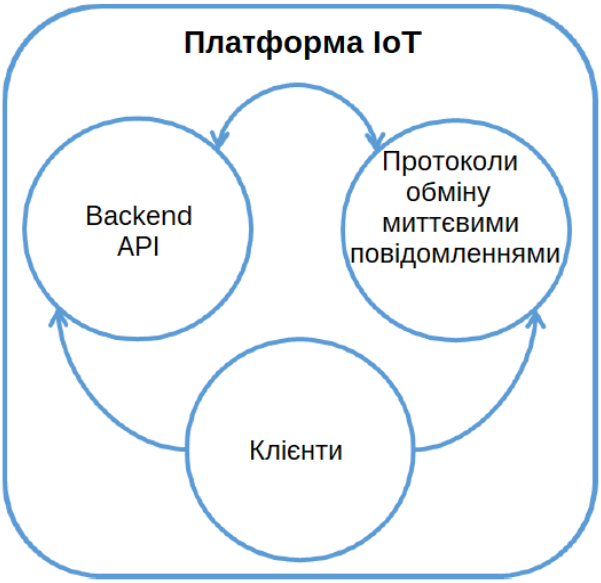

Рисунок 1.3 - Поточна ситуація платформи IoT

Безпека IoT привертає все більше уваги з боку дослідників, а також галузей промисловості, проте безпека IoT ще не є зрілою. Можна розглядати проблеми безпеки з трьох аспектів: недоліки самих протоколів, проблеми під час впровадження протоколу, уразливості під час інтеграції з IoT.

При реалізації протоколів з платформами IoT використовується механізм безпеки самих протоколів. Переглянувши згадані раніше протоколи зв’язку, таблиця 1.2 узагальнює механізми безпеки, які застосовуються в кожному протоколі зв’язку.

Актуальні проблеми безпеки комунікацій IoT. Доцільним було б розглянути аспекти вразливості самих протоколів зв'язку. Наприклад, інструмент атаки XMPPloit [20] показує наявність наявних вразливостей XMPP.

Таблиця 1.2 - Механізми безпеки комунікаційних протоколів IoT

Протокол	Характеристика
MQTT	Проста автентифікація імені користувача/паролю, TLS/SSL для шифрування даних
XMPP	Автентифікація SASL, TLS/SSL для шифрування даних
AMQP	Автентифікація SASL, TLS/SSL для шифрування даних
CoAP	DTLS/IPSEC
JMS	Спеціально для постачальника, але зазвичай на основі TLS/SSL. Зазвичай використовується з JAAS API
SOAP	Адресація до WS-Security

Нижче наведено список потенційних проблем безпеки цих протоколів. Проблеми безпечного завантаження, брандмауера та безпечного оновлення та виправлення для всіх протоколів. Автор перелічує проблеми безпеки, з якими може зіткнутися IoT, процес безпечного завантаження, брандмауер і безпечне оновлення та виправлення можуть мати деякі проблеми. У роботі [16] наводиться приклад: на вбудований пристрій має бути дозволено завантажувати лише авторизоване програмне забезпечення. Тому необхідні такі відносні заходи, як цифровий підпис або ключі для шифрування. Зважаючи на велике розгортання та обмежені ресурси, це буде проблемою для безпеки.

Проблеми використання DTLS для CoAP. У [8] вказується, що DTLS не розроблено для CoAP, тому можуть виникнути проблеми. Наприклад, DTLS не підтримує групову передачу. [14] перелічено обмеження застосування DTLS у CoAP та деякі пропозиції щодо вирішення цих проблем. Але ці пропозиції можуть не підходити для CoAP і спричинити проблеми з безпекою. Потрібні подальші дослідження.

Проблеми режиму NoSec CoAP. У [14] представлено режими безпеки, з яких перший - «NoSec Mode». У цьому режимі повідомлення CoAP передається без реалізованого механізму безпеки.

Проблеми шифрування та аутентифікації MQTT. Автори описують новий протокол під назвою Secure MQTT [21], який використовує шифрування на основі «lightweight attribute based encryption». Відмінності між sMQTT і MQTT можуть бути слабкою стороною MQTT.

Проблеми автентифікації SASL. Можуть виникнути проблеми з автентифікацією SASL, оскільки багато існуючих механізмів SASL не забезпечують належного захисту. Для захисту від атак можуть знадобитися додаткові послуги захисту. І XMPP використовує SASL для автентифікації. Тому будь-яка слабкість SASL буде слабкістю XMPP.

Відома вразливість UPnP. Вразливість UPnP не нова тема, про це вже було багато дискусій як в Інтернеті і в академічній області . Згадується, що вразливі місця безпеки в UPnP продовжують з’являтися і продовжують наражати на небезпеку зламу мільйони домашніх мережевих пристроїв. Автор також пояснює, як використовувати вразливості UPnP для здійснення атак.

Під час впровадження протоколу (наприклад, створення сервера та його встановлення) можуть виникнути проблеми безпеки, пов’язані з розробкою (наприклад, помилки, слабкі сторони, відсутність перевірок тощо), які створюють уразливості у всій реалізації. Згодом ці вразливості можуть бути відображені в базі даних Common Vulnerabilities and Exposures (CVE). CVE містить список відомих уразливостей для програмних продуктів, включаючи операційні системи, бібліотеки, фреймворки тощо, включаючи реалізації з відкритим і закритим кодом. Існують різні бази даних, які мають загальний список ідентифікаторів CVE; наприклад, деталі CVE [1] і бази даних NVD NIST. Ці бази даних описують усю доступну інформацію про вразливість, таку як постачальник, продукт, час, уразливі версії, тип уразливості, опис тощо. Таким чином, шляхом пошуку CVE протоколів, пов’язаних з Інтернетом речей, можна ідентифікувати колишні вектори експлуатації для програм або

платформ IoT, які використовують ці протоколи. Розробники повинні оновити та знайти рішення, щоб запобігти використанню будь-яких відомих уразливостей.

Деякі приклади пов’язаних CVE:

1) CVE-2014-2746: відома як «xmpp bomb», де зловмисник може надіслати дійсний стиснутий елемент XML із великою кількістю пробілів і спричинити відмову в обслуговуванні (DoS).

2) CVE-2016-9877: автентифікація з’єднання MQTT за допомогою пари ім’я користувача/пароля завершується успішно, якщо вказано наявне ім’я користувача, але пароль пропущено в запиті на з’єднання.

3) CVE-2010-0305: деякі версії ejabberd (сервер XMPP) дозволяють віддаленим зловмисникам викликати відмову в обслуговуванні за допомогою великої кількості повідомлень c2s (відомих як client2server), які викликають перевантаження черги.

Вразливі місця під час інтеграції з IoT. Платформа IoT об’єднує різні технології та протоколи IoT для спільної роботи. Обговорювалося, що IoT все ще є незрілою технологією як платформа, яка схильна до численних атак на безпеку. Ці проблеми безпеки виникають під час інтеграції різних протоколів додатків IoT, і це може бути лише початком у контексті IoT. Таким чином, можна застосовувати різні підходи для збору, аналізу та, зрештою, ідентифікації будь-яких варіантів щодо загроз платформ IoT.

Для досягнення мети необхідно вирішити наступні задачі:

1) Провести дослідження поширених протоколів, пов’язаних з Інтернетом речей, таких як MQTT, XMPP, XDS, AMQP, UPnP, щоб отримати та зрозуміти їх функціональні особливості. Така робота має забезпечити основу для зосередження уваги на конкретних протоколах і потенційних підтемах дослідження.

2) Визначити потенційні недоліки вищезгаданих протоколів; виконати огляд літератури, щоб сформувати розуміння відомих проблем безпеки.

3) Проаналізувати роботу технології Honeypot, її призначення, особливості архітектури та потенційні переваги для впровадження протоколу IoT. Існуючі технології honeypot як для загальної платформи, так і для IoT потрібно переглянути, щоб визначити потенційні шляхи вдосконалення та відповідні можливості для нових реалізацій honeypot IoT.

4) Після визначення теоретичної основи потрібно опрацювати реалізацію приманки IoT.

5) Розробити механізм аналізу даних для збору та обробки журналів, створених honeypot (тобто мережевий трафік, журнали активності).

2 СТРУКТУРА СИСТЕМИ ВИЯВЛЕННЯ АТАК НА ІНТЕРНЕТ РЕЧЕЙ

2.1 Технологія Honeypot

Технологія honeypot - це механізм, який фіксує дії зловмисника, імітуючи реальну систему, але розміщену в захищеному середовищі. Зловмисник досягає honeypot, коли він розглядається як справжня система або пристрій. Оскільки honeypot реалізовано захищеним і контрольованим способом, діяльність зловмисника не завдасть шкоди системі, а інформація зловмисника записується. Основною концепцією технології honeypot є емуляція або розгортання протоколів зв'язку, що працюють через служби (програмне забезпечення), які діють як системи-приманки або пастки для зловмисників, автоматизованого шкідливого програмного забезпечення або будь-якого джерела зловмисної чи аномальної діяльності.

Переваги технології Honeypot. Ефективність підходів, які будуть реалізовані, залежить від контексту, типу мережі, обмежень щодо того, як пристрої IoT розгортаються в мережі. Усі три згадані альтернативи можуть надати корисні дані для аналізу, однак поточне дослідження спрямоване на вивчення використання honeypots через три основні причини:

1) Honeypot є перевіреним способом збору даних про:

- шаблони атак (статистика трафіку, тенденції напрямків атак);
- зразки шкідливих програм;
- потенційні недоліки реалізації протоколу;
- обсяг даних не обов'язково є індикатором збору корисних наборів даних.

2) Honeypot використовують переваги емуляції для вирішення:

- заходи безпеки та більш контрольоване/обмежене середовище;
- відсутність фактичних пристроїв і, отже, гнучкості для охоплення більшої кількості контекстів.

3) IoT все ще перебуває на ранніх стадіях розробки, тому Honeypot може допомогти зрозуміти їх розвиток з точки зору безпеки на ранніх стадіях, щоб:

- висновки можуть дати інформацію для розвитку механізмів виявлення (наприклад, підписи IDS, ACL);

- підтримка генерації зразків посилань (тобто трафіку/моделей), які можна використовувати для подальших приманок або будь-якого механізму виявлення чи протидії.

Підхід до технології honeypot і зв'язок із платформами IoT

Вивчаючи вразливості, згадані раніше, приманка може використовувати інформацію та симулювати вразливу версію реалізації. Таким чином це може залучити зловмисників до виконання атак. І таким чином фіксувати дії та інформацію зловмисника.

Використовуючи honeypot, можна емулювати не лише пристрої IoT, але й цілу платформу IoT, яка забезпечує контроль доступу та системи керування для пристроїв IoT. Емуляція може виконуватися на кількох рівнях і в кількох сферах, щоб зловмисник міг отримати ширший рівень взаємодії, а отже, аналізувати більше інформації.

Емуляція пристрою може бути досягнута шляхом розгортання як фактичних, так і емульованих пристроїв, що надають підроблені дані та функції (наприклад, датчики руху, температури, «розумне освітлення» тощо).

Емуляція платформи може бути досягнута шляхом розгортання як фактичних, так і емульованих серверів (наприклад, XMPP, MQTT, серверні модулі REST api, які обслуговують керування IoT).

Рівень взаємодії можна визначити як діапазон можливостей, які honeypot надає зловмисникові. Загалом існує три типи приманки за рівнем взаємодії.

Приманки високої взаємодії (HIH). HIH (High Interaction Honeypots) - це в основному реальні системи, які використовують стандартні реалізації протоколів. Її головна особливість полягає в тому, що, будучи реальною системою, можна здійснювати повну взаємодію зі зловмисниками. Однак існують проблеми безпеки, які необхідно вирішити, оскільки зловмисні дії

можуть реалізуватись. Таким чином, HIH передбачає розгортання систем моніторингу та мережевого контролю, які забезпечують безпеку середовища під час дій зловмисників.

Платформа IoT може бути реалізована шляхом розгортання приманок XMPP/MQTT/REST HIH, будучи реальними системами. Тоді зловмисники можуть взаємодіяти з початкової точки. Подальшу емуляцію (тобто взаємодію з пристроями) можна реалізувати через LIH.

Приманка низької взаємодії (LIH). Приманки з низьким рівнем взаємодії (Low Interaction Honeypot) виявляють зловмисників за допомогою програмної емуляції характеристик конкретної операційної системи, програми, мережевих служб або протоколів поверх операційної системи хоста. Перевага цього підходу полягає в тому, щоб отримати кращий контроль над діяльністю зловмисника та зменшити ризики для безпеки, оскільки зловмисник обмежений емульованими функціями, які за реальних умов експлуатації атаки не працюватимуть, але вони будуть записані. З іншого боку, недоліком цього підходу є той факт, що honeypot з низьким рівнем взаємодії емулює служби або кроки в рамках протоколу, але він може не емулювати повний дизайн або функції таких програм або протоколів, що також може обмежити отримання даних і взаємодію зі зловмисником. Прикладами такого типу honeypot є Dionaea, Honeyd, NetBait, Кіппо тощо.

Емуляцію пристрою IoT можна виконати через LIH. Крім того, можна використовувати взаємодію як з реальними, так і з іншими емульованими XMPP і існуючими службами MQTT.

Приманки середньої взаємодії (MIH). MIH (MediumInteraction Honeypot) - це приманки, які пропонують зловмисникам більше можливостей для взаємодії, ніж приманки з низьким рівнем взаємодії, але меншу функціональність, ніж рішення з високою взаємодією. Вони можуть очікувати певної активності та розроблені, щоб давати певні відповіді, на відміну від того, що дасть приманка з низьким рівнем взаємодії. MIH поєднує в собі

функції як LIH, так і HIH, однак їх розробка та розгортання можуть бути більш складними.

Прототип запропонованого IoT honeypot фактично включає MIH. Однак не передбачається включати повну функцію дизайну для роботи honeypot, оскільки ранніх етапів і незалежної емуляції модуля (пристроїв ig) може бути достатньо для збору даних для подальшого аналізу.

Таблиця 2.1 – Порівняння HIH, LIH та MIH

	Переваги	Недоліки
HIH	-забезпечує більш привабливе середовище для взаємодії; -більш глибока взаємодія зі зловмисником; - збір більшої кількості даних.	-висока трудомісткість реалізації і підтримки; - вищий ризик бути зламаним і контрольованим зловмисником.
LIH	- низька трудомісткість реалізації; - менший ризик бути зламаним і контрольованим зловмисником; - висока масштабованість; -не вимагає складного обчислювального механізму.	- обмеження даних, які він може отримати; - відсутність глибшої взаємодії з атакуючим.
MIH	-забезпечує баланс вартості та отриманої цінності даних; - вища функціональність, ніж LIH.	- вищі витрати, ніж LIH.

Поточна ситуація використання honeypot для IoT.

Існує лише кілька відомих реалізацій honeypots, орієнтованих на Інтернет речей:

1) Telnet IoT honeypot - це приманка, яка реалізує сервер Telnet на Python для виявлення зловмисного програмного забезпечення IoT.

2) HoneyThing - це приманка, розроблена для протоколу TR-069 (CPE WAN Management Protocol), призначеного для Інтернету TR-069 Things.

3) IoTPOT - нова приманка для емуляції служб Telnet різних пристроїв IoT для глибокого аналізу поточних атак. IoTPOT складається з інтерфейсного відповідача з низьким рівнем взаємодії, який співпрацює з серверними віртуальними середовищами з високим рівнем взаємодії під назвою IoTBOX.

IoTBOX працює з різними віртуальними середовищами, які зазвичай використовуються вбудованими системами для різних архітектур:

- Dionaea - фреймворк honeypot, який, серед іншого, реалізує модуль MQTT;

- ZigBee Honeypot - приманка, яка імітує шлюз ZigBee;

- Багатоцільовий IoT honeypot- приманка IoT, яка зосереджена на Telnet, SSH, HTTP та CWMP.

2.2 Топологія мережі

Необхідно реалізувати перевірку концепції IoTPot, щоб побачити поведінку та виявити дії зловмисників. Так, потрібно описати підтвердження концепції (PoC), а також один із варіантів використання (платформа Philips Hue) таким чином, щоб можна було продемонструвати, як IoTPot можна реалізувати, відправляти та отримувати корисні дані.

У поточній роботі реалізуються миттєвого зв’язку. REST можна використовувати для створення RESTful api. Його легко реалізувати та він може надавати функції, які повинна мати серверна частина платформи IoT. Багато літератури вказують на XMPP як на IoT-протокол для спілкування в реальному часі. Таким чином, XMPP і REST api вибрано для використання в IoTPot. Подібно до опису загальної платформи IoT, використання XMPP і REST для спільної роботи може бути достатнім для розгортання платформи IoT, подібної до існуючих реалізацій. Таким чином, оскільки чим більш

відкритим є honeypot, тим більше шансів для зловмисників дістатися до приманки, тому послуги будуть реалізовані таким чином, щоб вони мали якомога більшу видимість в Інтернеті. На рисунку 2.1 показано топологію приманки, що досліджується у поточній роботі.

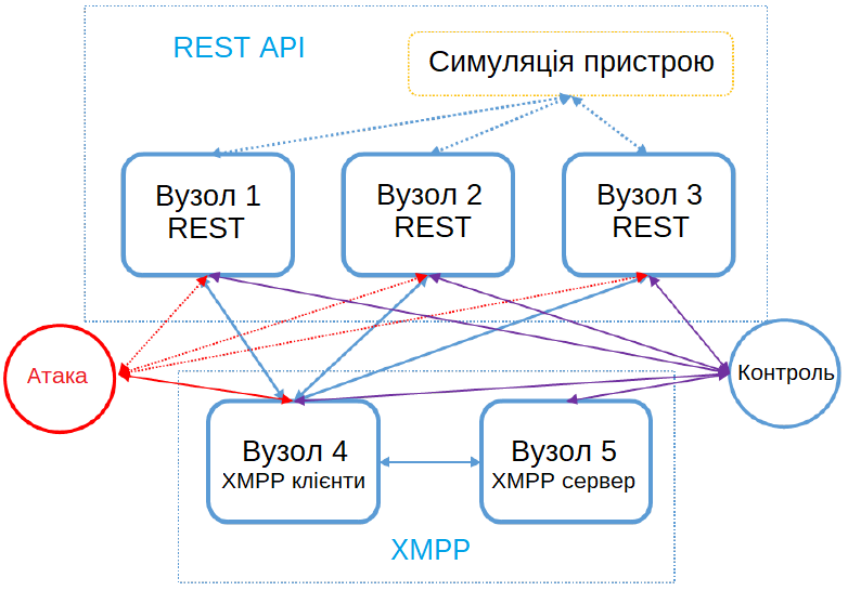

Рисунок 2.1 – Фізична топологія реалізації IoTPot

Реалізація IoTPot включає три основні компоненти:

- вузли XMPP, де два вузли використовуватимуться як частина XMPP, включаючи один вузол, який запускає клієнтські служби XMPP, і один вузол, який запускає служби сервера XMPP;

- вузли REST, де присутні три вузли, які мають публічні IP-адреси, будуть використані для реалізації частини REST honeypot;

- контролер.

У цій реалізації також буде реалізовано контролер за дизайном IoTPot. Функціями контролера є спостереження за журналами, контроль, резервне копіювання даних, оновлення коду тощо. Контролер має різні доступи зловмисника через ssh до різних портів.

Кожен вузол розгорнуто на різних пристроях, включаючи Raspberry Pi (RPi) і віртуальні сервери, що працюють у хмарі (наприклад, Amazon EC2 і Keyweb). Деякі вузли запускають екземпляр серверної частини REST api, тоді

як інші запускають служби XMPP (сервер або клієнт). Служби працюють у блоці, встановленому на RPis, оскільки блок можна легко перемістити на інший вузол. Блоки надають (у певній мірі) додатковий рівень безпеки, оскільки існує ізоляція мережі та процесів від інших служб. У контексті Honeypot із низьким рівнем взаємодії цей рівень безпеки може зменшити ймовірність того, що в результаті фактичного зламу зловмисник досягне фактичної системи, що стоїть за емуляцією служби IoTPot. Крім того, на вузлах 1 і 2 проксі-сервери працюють як резервні у випадку, якщо api тимчасово не працюватиме. Проксі також може допомогти замаскувати серверну частину від веб-сцени. Частина REST і частина XMPP можуть спілкуватися одна з одною, дотримуючись певних правил, які є такими ж, як і для реальної системи.

У таблиці 2.2 показано IP-адреси та служби на кожному вузлі. Серед цих вузлів вузол 1 і 5 фактично працюють на одному RPi, але в різних блоках, до яких можна дістатися через різні порти. Таким чином, ці дві служби є повністю незалежними та не мають взаємного впливу через спільний Rpi. Тому вони названі двома номерами відповідно до різних служб, щоб зробити топологію більш зрозумілою.

На рисунку 2.2, ліва частина (червоного кольору) показує потенційного нападника. Верхня частина (жовтим кольором) показує пристрій, який знаходиться за api. Червона лінія показує потенційний зв’язок, до якого може дістатися зловмисник. Через певний порт зловмисник міг отримати доступ до honeypot і взаємодіяти з приманкою. Номер порту, до якого зловмисник може отримати доступ і взаємодіяти, показано в таблиці 2.2. Згідно з характеристиками XMPP, загальнодоступна IP-адреса не потрібна, щоб клієнтська служба XMPP була доступною, оскільки вона підключається до сервера XMPP, який має публічну IP-адресу. Сервер XMPP правильно оброблятиме трафік. Однак, припускається, що метою зловмисника є пристрій. Таким чином, зловмисник може досягти пристрою двома шляхами (рис. 2.2).

Таблиця 2.2 – Список IP-адрес вузлів

Вузол	Сервіс	IP-адреса	Порт
1	REST	94.210.46.X	TCP/80
2	REST	83.84.11.X	TCP/80
3	REST	84.19.177.X, 84.19.176.X	TCP/80
4	REST	Внутрішній сервер	-
5	Сервер XMPP	94.210.46.X	TCP/5222 і TCP/5269

Специфікація випадку використання Philips Hue. Щоб підтвердити ефективність дизайну IoTPot для отримання фактичних даних, пов'язаних із платформами IoT, для емуляції обрано варіант використання існуючої комерційної реалізації. Так, вибір сценарію використання базується на тому факті, що це має бути існуюча модель, у якій враховані всі компоненти загальної платформи IoT (тобто пристрої, серверна частина, протокол зв'язку). Таким чином, розглядається Philips Hue, оскільки це набір розумних домашніх пристроїв, які є одним із найпопулярніших інтелектуальних бездротових ламп, які використовують REST і легко доступні на ринку. У поточній роботі реалізація IoTPot базується на прикладі використання PHD, симулюючи платформи, які інтегрують XMPP з PHD.

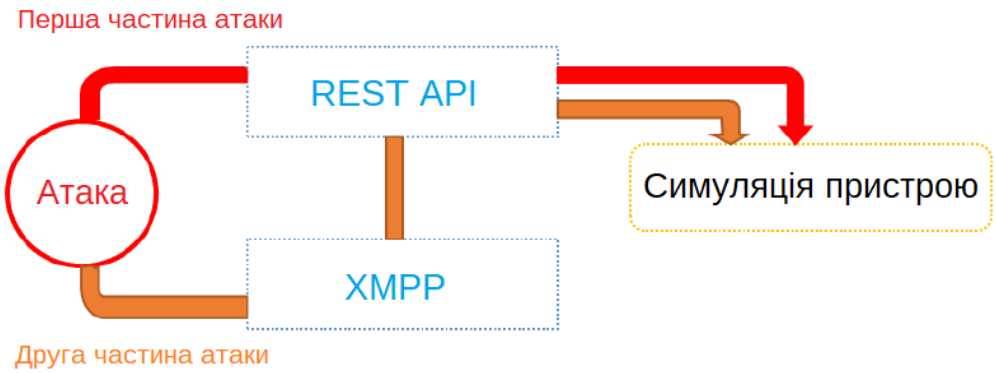

Рисунок 2.2 – Шляхи атаки

Характеристика платформи Philips Hue IoT. Продукт PHD складається з бездротових світлодіодних ламп і бездротового мосту. Лампами можна керувати за допомогою додатків для iOS та Android або через веб-сайт Hue. Багато людей встановили його вдома, грали з додатком на смартфоні та складали власний план освітлення. Деякі розробники знаходять способи інтегрувати Philips Hue із протоколом XMPP. Є стороння реалізація (вихідний код) і технічні посібники як працює XMPP PHD. Така платформа IoT впроваджується все більшою кількістю людей, а проблеми безпеки рідко розглядаються. Проблеми виникають або через протоколи зв'язку, або під час інтеграції пристроїв і протоколів. Таким чином, IoTPot симулює платформу IoT, яка включає пристрої PHD і панель керування користувача (частина XMPP).

На рисунку 2.4 показана топологія платформи інтеграції PHD і XMPP, де задіяні 4 основні частини:

1) Сервер XMPP, який призначений для обміну даними XMPP. Він передає всі повідомлення між клієнтами XMPP. Цей сервер XMPP може бути публічним сервером, створеним кимось. Або користувач також можете створити власний сервер XMPP і налаштувати його для відповідного обміну даними.

2) Пристрої PHD, до яких входять міст PHD і лампи (або інші датчики, наприклад датчики руху). Міст PHD Bridge є центром розумних ламп і датчиків. Лампи та датчики виходять в Інтернет не напряму, а через міст. Міст PHD зв'язується з лампами та датчиками за допомогою ZigBee, а з сервером PHD або користувачами за допомогою REST api. Він містить всю інформацію про підключені пристрої.

3) ПК-інтегратор PHD і XMPP. Автор представив спосіб інтеграції XMPP із пристроями PHD шляхом запуску наданого коду. За допомогою цього інтегратора кожен розумний пристрій може мати JID. Користувач може надіслати повідомлення до JID, щоб керувати світлом, наприклад увімкнути чи вимкнути, налаштувати яскравість тощо. Цей інтегратор є сценарієм, який

використовує бібліотеку sleekxmpp і PHD. Користувач також може змінити код відповідно до власних вимог. По суті, він отримує повідомлення xmpp, потім перетворює його на запит REST і надсилає запит на міст PHD. Код можна запустити на пристрої, який знаходиться в одній мережі з мостом.

4) Клієнт XMPP. Тут користувач може подати заявку на отримання облікового запису XMPP на будь-якому доступному сервері XMPP. А потім спілкуватися з пристроями PHD через XMPP.

2.3 Моделювання сценарію роботи системи

Однією з особливостей honeypot є «приманка», що означає, що honeypot повинен переконати зловмисника повірити, що це система, яку варто атакувати. Таким чином, чим більше приманка буде схожа на реальний сценарій, тим більша ймовірність того, що приманка виявлятиме та реєструватиме дії зловмисників. Honeypot імітує сценарій, який є звичайним у реальному житті, це робить атаку більш переконливою та привабливою для зловмисника. У цьому підрозділі буде описано сценарій, який honeypot збирається симулювати та представити зловмисникам.

Змодельований сценарій: система Philips Hue, яка має дві розумні лампи, підключені до одного мосту PHD, розміщення десь у будівлі. А арі мосту PHD має загальнодоступну IP-адресу, доступ до якої можна отримати в Інтернеті. Користувач розробив клієнт XMPP, який може спілкуватися з сервером. JID зареєстровано на сервері XMPP для керування та моніторингу однієї розумної лампи. Один JID прив’язує одну розумну лампу, якою користувач міг би керувати. Її представляє JID, через зв’язок XMPP з JID.

Насправді існує багато JID, запущених для збільшення ймовірності атаки. На рисунку 2.3 показано зв’язок між JID, які прослуховують клієнтську приманку XMPP, і розумну лампу, доступну в приманці REST. Суцільна лінія означає, що JID пов’язаний із розумною лампою.

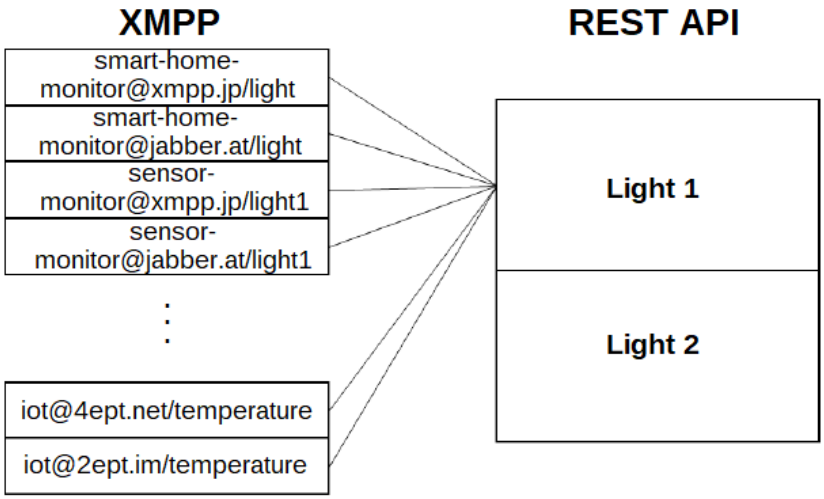

Рисунок 2.3 - Взаємозв'язок облікового запису та пристрою між XMPP і REST honeypot

Реалізація REST. IoTPot містить api RESTful, який імітує поведінку api мосту PHD. Django використовується для створення api. Рис. 2.10 - це блок-схема приманки REST у реальній реалізації. Він відповідає дизайну Backend API, описаному раніше. Він розроблений для моделювання поведінки мосту PHD з деякими функціями, щоб зробити взаємодію зловмисника легшою та глибшою. З блок-схеми можна побачити, що honeypot REST в основному складається з двох частин: API та менеджера. Ця структура заснована на дизайні Django, де менеджер призначений для налаштувань і керування api, а api призначений для реалізації відповідних функцій.

Під елементом api в основному приймають 6 компонентів для honeypot REST:

1) urls: конфігурація шаблонів URL honeypot. Тут відображаються шаблони URL-адрес у функції Python.

2) views: це код Python для отримання запитів і повернення відповідей.

3) Data Resource: тут включено три файли Json шаблонів відповіді. Шаблон і конфігурація відповідають формату структури даних ресурсу мосту PHD із вмістом, який виглядає як справжній пристрій. Додатковою функцією

реалізації honeypot, якої немає у справжньому мосту PHD, є tempfile. Вона розрахована на те, щоб зловмисник міг дізнатися більше інформації про пристрій. Таким чином, необхідно перейти на подальшу взаємодію, і приманка зможе ловити більше дій і даних.

4) GetTemplate: це обробник для обробки відповіді на запити з Data Resource.

5) Service: це клас, який включає інструменти, які будуть викликані для деяких конкретних функцій, що безпосередньо не пов'язані з honeypot, наприклад, генерування випадкового рядка, розбір заголовка HTTP тощо.

6) middlewarelogrequest: Django використовує механізм проміжного програмного забезпечення для служб «plugin». Тут створюється налаштоване проміжне програмне забезпечення, яке розміщується під api. Це служба для системи журналу цієї реалізації honeypot і експорту її у файл json. Він аналізує всі вхідні запити та корисну інформацію журналу у форматі Json, який є гнучким форматом даних, і його легко аналізувати. Це дуже важливий компонент у такій реалізації, тому що належне реєстрування інформації має важливе значення для аналізу даних приманки. Це означає, що в журналі не повинно бути жодної корисної інформації, а формат даних має бути достатньо зрозумілим для легкого аналізу.

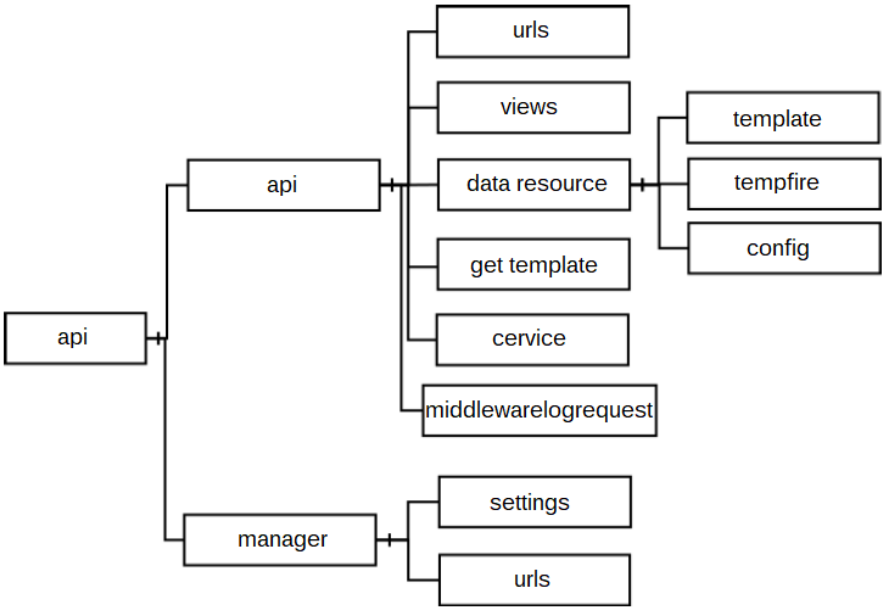

Рисунок 2.4 – Блок-схема реалізації Honeypot REST

Визначення URL-адреси та керування ресурсами даних у реалізації IoTPot дотримуються основного правила api мосту PHD. Але дії не зовсім збігаються з реальними. Honeypot api моделює деякі дії мосту PHD, але не повністю.

Моделювання Honeypot api включає:

1) Реєстрація для нового імені користувача.

Нове ім'я користувача, яке буде включено в білий список, можна зареєструвати, надіславши запит POST на міст після натискання кнопки мосту (через 30 секунд). Він поверне повідомлення про успішне виконання, яке включає 40-значний рядок, який є новим іменем користувача в разі успіху. Інакше він надішле повідомлення про помилку. Honeypot імітує таку поведінку, але приймає всі запити та повертає повідомлення про успішне виконання з 40-значним рядком, згенерованим випадковим чином. Це зроблено для того, щоб зловмисник повірив, що він успішно отримав дійсне ім’я користувача, щоб мати стимул діяти далі. Таким чином, зловмисник матиме більше дій, а не просто намагатиметься зареєструвати нове ім’я користувача.

2) Отримання всієї інформації про відповідний міст.

Як згадувалося раніше, дані справжнього мосту PHD мають багатошарову структуру та формат Json. Повні дані Json можна отримати за допомогою HTTP-запиту GET /api/<username>, де <username> має бути дійсним.

3) Отримання інформації про підрівень.

Це може бути - "ligts"/"chonfig"/"scenes"/"whitelist"/ і т.д. Будь-яку інформацію на будь-яких рівнях можна отримати за допомогою HTTP-запиту GET із дійсною URL-адресою. Наприклад, щоб отримати «whitelist» мосту, перевіривши URL-адресу «http://<bridge IP address>/api/<username>/config/whitelist».

4) Керування освітленням.

Значення керованих параметрів, згаданих раніше, можна контролювати за допомогою HTTP-запиту в цьому api honeypot. Коли зловмисник надсилає дійсний запит HTTP PUT, api відповість «успішно» повідомленням у тому самому форматі, що й справжній міст PHD.

5) Зміна параметру за допомогою HTTP PUT.

Це схоже на «Керування світлом». На справжньому мосту PHD деякі параметри (наприклад, ім'я елемента в конфігурації) можна змінити за допомогою запитів PUT. Honeypot імітує таку поведінку, що надсилає повідомлення про успішне виконання, коли отримує такий запит.

Реалізація XMPP

Реалізація частини XMPP honeypot відповідає рішенню, показаному на рис. 2.2 і робочий процес, який показано в UML (рис. 2.3, 2.4). На рис. 2.5 показано компоненти, які були реалізовані для підтвердження досліджуваної концепції. Виділена частина (білим) це ті, які реалізовані. Так, є можливість підключатися до мосту PHD як незалежний honeypot і підключитися до api honeypot. Таке рішення включає XMM, XCM, XHM, XLM, XHC.

XCM - це модуль з'єднання для XMPP Honeypot. Включені функції підключення до сервера XMPP. Сценарій IoT_PhilipsHueapi _meng_api.py викликається XMM у випадку використання Philips Hue. Він повідомляє правому серверу XMPP, що один JID (як один із вхідних параметрів) знаходиться в мережі. І один JID пов'язаний з одним пристроєм на мосту PHD. Цей сценарій визначає команди, які можна надсилати через чат XMPP або через XEP0323 і XEP0325. Він також перетворює ці команди на дійсні запити до мосту PHD. У цій реалізації міст PHD є частиною REST honeypot, яка є api, створеним Django. Отже, XCM тут є з'єднувачем для сервера XMPP і api. Він створений на основі sleekxmpp і PHD-lib шляхом імпорту бібліотек і перезапису деяких функцій. У цій реалізації, клієнтська приманка XMPP підключається до REST api приманки. І всі JID зареєстровані для «light 1» мосту PHD у приманці REST.

Частина з XLM призначена для обробки вхідних запитів і поведінки системи в журналах. У реальній реалізації XLM розміщується всередині XCM. Він визначає новий реєстратор, який повинен зареєструвати деякі повідомлення у форматі Json та експортувати у файл. Спосіб реєстрації дуже важливий для XMPP Honeypot. Він має бути повним і зрозумілим для подальшого аналізу даних.

У XHC розміщено файл конфігурації. Файл конфігурації Honeypot має формат YAML. YAML розглядається як одна з основних мов розмітки для конфігурації Python, і має 6 основних компонентів.

Так, «names», «domains», «resources» використовуються для JID. Оскільки сценарій у XCM може зареєструвати лише один JID на сервері для одного пристрою, і вони тут стосуються імені вузла, домену та імені ресурсу в JID відповідно. Комбінація трьох елементів буде JID. У файлі конфігурації під кожним елементом буде розміщено список імен вузлів, доменів і ресурсів. Потім, коли файл конфігурації викликається XMM, відбувається ітерація комбінацій цих трьох елементів. Умова полягає в тому, щоб комбінація, яка створює JID, була застосована на сервері XMPP.

У цій реалізації застосовано кілька JID, які мають однакове ім’я вузла на різних серверах XMPP (що робить домен іншим). І JID, які використовуються в цій приманці, названо таким чином, що, очевидно, це для пристроїв IoT. Наприклад, імена вузлів: «smart-home-monitor», «sensormonitor», «iot», «home» тощо. Таким чином, це може бути більш релевантним для зловмисника (див. рис. 2.11).

Іншою функцією згідно зі згаданим раніше дизайном клієнтської приманки XMPP є один JID (названий менеджером JID) для контролю та моніторингу інших JID. Це реалізується розробкою інтуїтивно зрозумілого відстежування стану усіх JID.

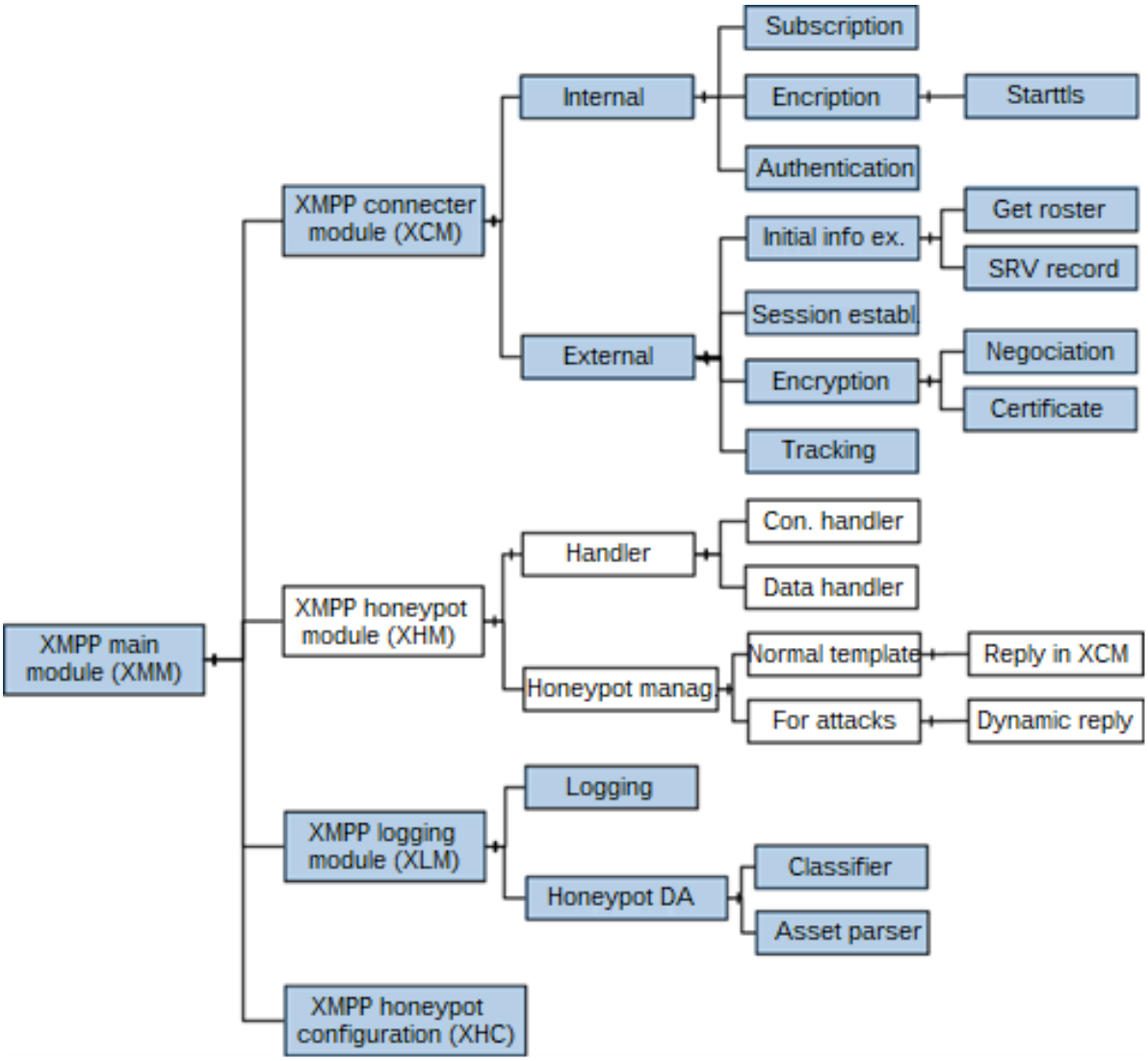

Рисунок 2.5 - Модулі клієнта XMPP honeypot

Коли JID перебуває в режимі онлайн, він надсилає повідомлення XMPP диспетчеру JID із повідомленням «XXX(JID) у мережі». Якщо розробник увійшов до облікового запису менеджера JID, інтуїтивно зрозуміло, який обліковий запис успішно прослуховується.

2.4 Система журналу honeypot

Система журналу honeypot є важливою, оскільки аналіз даних в основному базується на журналах, створених системою журналу. Кожна

частина (REST і XMPP) honeypot має власну систему журналу. Тим часом системи журналів також мають деякі спільні риси для кореляції, які можуть сприяти аналізу даних.

Система журналу REST. Система журналу REST класифікує журнали на два класи: звичайний журнал консолі Django та вхідні запити і необхідні відповіді. Різні типи журналів будуть експортовані в різні файли.

Система журналу має власний спосіб іменування файлів журналу. Щоб зробити це зрозумілим і уникнути перезапису через повторний запуск honeypot, мітка часу використовується в назві файлів журналу. Таким чином, ім’я файлу журналу ніколи не буде однаковим під час кожного запуску honeypot. Крім того, щоб розрізняти різні класи журналів, імена файлів журналів починаються з різних імен. Звичайний журнал консолі Django запускається за допомогою sys і має назву «sys + TIMESTAMP.log». Журнал запитів запускається з json і має назву "json+TIMESTAMP.log". Журнали консолі Django ведуться за допомогою налаштувань Django. Тут реєструються всі системні дії Django.

Система журналу запитів реєструє всі запити та відповідні відповіді незалежно від того, дійсні чи недійсні вони. Це робиться шляхом створення проміжного програмного забезпечення Django для обробки вхідних запитів. Після аналізу заголовка та іншої корисної інформації генерується повідомлення журналу у форматі Json і зберігається в окремому файлі. Кожен рядок цього файлу є одним записом Json для одного запиту. На рисунку 2.6 показано приклад з журналу, який зареєстрував приманку REST. Це один запис у файлі журналу, який аналізується онлайн-аналізатором JSON Editor Online . Ключі цього запису Json:

- time: поточний час. Часовий пояс може знадобитися встановити вручну.

- body: тіло запиту. Коли запит це PUT або POST, тіло зазвичай не є порожнім. Зловмисники можуть надсилати певні дивні команди.

- header: заголовок HTTP-запиту. Він містить багато корисної інформації, такої як агент користувача, IP-адреса, ім’я хоста тощо.

- url: URL-адреса запиту.

- entry_id: Лічильник записів. Це унікальне значення в одному файлі журналу, і його можна використовувати для надання ідентифікатора для кожного запису журналу Json.

- type: тип HTTP-запиту, наприклад «PUT», «POST», «GET» тощо. Це важливе значення, яке можна проаналізувати пізніше.

- remote_ip: віддалений ip зловмисника.

- reply_content: відповідь honeypot на запит буде зареєстровано тут. Відповідно до реалізації REST, відповідь завжди надається у форматі Json шляхом імітації справжнього мосту PHD.

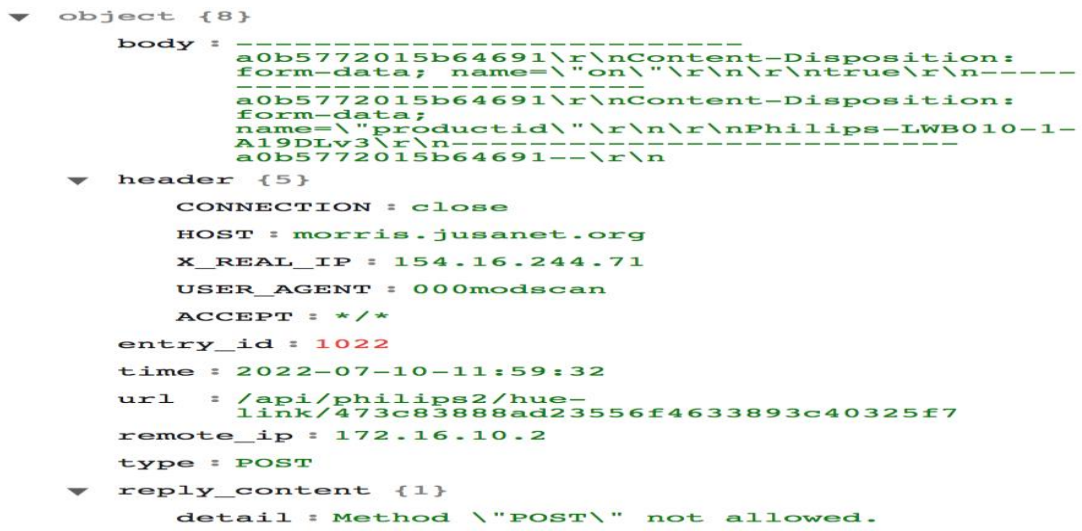

Рисунок 2.6 - Приклад одного запису файлу журналу honeypot REST у форматі Json, проаналізованого Json Online

Система журналу XMPP. Система журналу XMPP у honeypot також має два класи журналів. Оскільки приманка XMPP базується на sleekxmpp і PHD-lib, один клас журналу - це журнали, які показують статус і інформацію про цей процес. Він називається системним журналом XMPP. Система журналу

зберігає один файл журналу для кожного JID з іменем JID, який є унікальним і не буде перезаписаний.

Каталог із назвою «syslog + TIMESTAMP.log» у каталозі шляху до файлу, указаному у файлі конфігурації, створюватиметься щоразу на початку запуску цього honeypot. Файли системного журналу зберігаються в цьому каталозі.

Інший вид журналу схожий на журнал запитів REST honeypot. Це у форматі Json і збережено у файлі, кожен рядок якого є записом Json. Він називається журналом трафіку XMPP. Оскільки він реєструє весь трафік, включаючи вхідні та вихідні повідомлення через чат XMPP або XML-потік до сервера та взаємодію з api (HTTP-запит і відповідь). На рисунку 2.7 показано приклад одного запису Json із журналу трафіку XMPP.

Запис Json містить:

- shared_id: використовується для взамодії з REST щодо подальшого аналізу даних.

- jid: це вказує, з якого облікового запису JID створено журнал, і оскільки весь трафік реєструється в одному файлі, незалежно від JID, цей елемент використовується для розрізнення;

- unexpected: генерується в результаті першого аналізу XMPP-клієнта honeypot, і значення «True» означає, що цей трафік потенційно є атакою зловмисника, яка потребує подальшого аналізу;

- content: це вміст трафіку, і якщо трафік є повідомленням XMPP, це буде вміст повідомлення, а якщо трафік є відповіддю HTTP - це дані відповіді з api;

- time: це час створення журналу;

- message: «xmpp» або «api», що вказує, звідки або куди здійснюється цей трафік;

- type: або «receive», або «send», вказується, чи походить трафік від інших об’єктів, чи надіслано клієнтською приманкою XMPP;

- mode: це тип HTTP-запиту, наприклад «PUT», «POST», «GET», та існує лише тоді, коли це запит HTTP, надісланий клієнтом XMPP honeypot.

- address: це URL-адреса HTTP-запиту, та існує лише тоді, коли це запит HTTP, надісланий клієнтом XMPP honeypot.

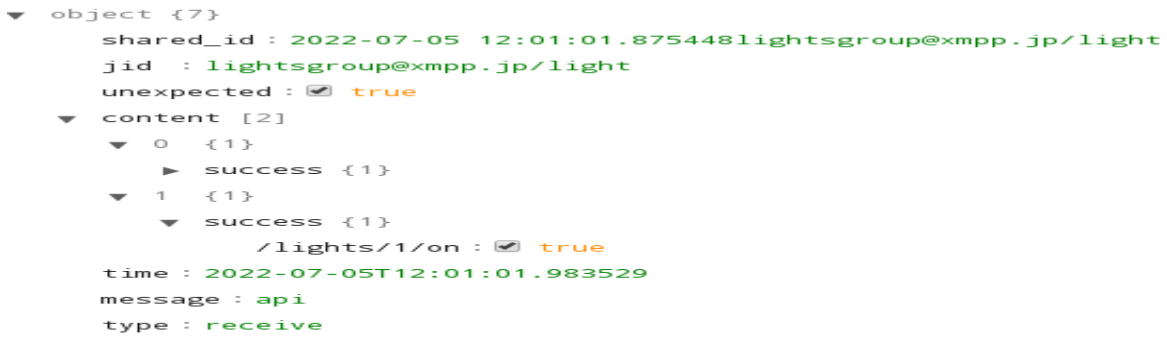

Рисунок 2.7 - Приклад одного запису файлу журналу трафіку XMPP у форматі Json, аналізованого Json Online

Метод "shared_id"

Метод використовується для взаємодії REST і XMPP honeypot у системі журналу. Як згадувалося раніше, JID клієнтської приманки XMPP пов’язано з розумною лампою приманки REST. Отже, система журналу REST і XMPP не повинна бути абсолютно незалежною. Метод «shared_id» запропоновано та застосовано тут для зв’язування записів журналу між журналом Json REST і журналом Json XMPP.

Під час отримання повідомлення або трафіку від частини XMPP система журналу збирається згенерувати «shared_id», який базується на пов’язаному JID і поточній позначці часу. Формат "shared_id": TIMESTAMP+JID (див. рис. 2.7. Потім цей згенерований «shared_id» записується в заголовок HTTP-запиту до api і передається в приманку REST.

3 АНАЛІЗ РЕЗУЛЬТАТІВ ДОСЛІДЖЕННЯ

3.1 Методика аналізу даних

Аналіз даних в основному базується на журналах кожного вузла. Протягом часу дії приманки знімаються журнали з кожного вузла. Однак, спостерігаючи за журналами на вузлі XMPP, не було знайдено відповідних даних. Таким чином, під час аналізу даних журнали REST із трьох вузлів арі 1, 2, 3 братимуться до уваги як основне джерело. Ці дані мають два види: журнал проксі та журнал арі. Проксі-сервер встановлено на вузлах 1 і 2. Таким чином, журнал проксі-сервера не може відображати поведінку вузла 3. Хоча всі три вузли мають журнал арі. Однак журнал арі не охоплює всю інформацію проксі-сервера, оскільки він може обробляти лише запит HTTP. Журнал проксі працює весь час та використовується для спільної роботи (взаємодоповнення) і обміну повідомленнями. Методика аналізу даних показана на рисунку 3.1.

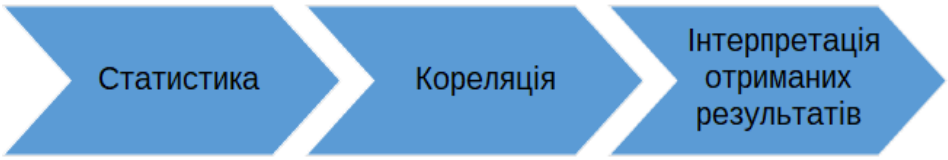

Рисунок 3.1 – Методика аналізу даних

Аналіз даних складається з трьох етапів:

1) Статистика.

Таблиці та зображення статистики створюються з необроблених даних (журналів), щоб мати початкове розуміння того, що сталося в приманці. Вона передбачає класифікацію, підрахунок і просту інтерпретацію даних. Статистика є набором даних для етапу кореляції.

2) Кореляція.

Кореляції між статистичними даними, представленими на першому етапі, виконується для пошуку додаткової інформації про зловмисників і атаки. Замість того, щоб аналізувати кожну частину статистики окремо,

кореляція полягає у пошуку взаємозв’язку серед усієї наданої інформації.

3) Інтерпретація отриманих результатів.

Після проведення кореляції розуміння даних приманки має стати більш чітким. Таким чином надається можливість зробити підсумок і має бути сформульований ряд висновків. Атаки можна проаналізувати на основі статистики та кореляції.

3.2. Результати обробки даних

Цільові запити. Переглянувши всі запити, їх можна розділити на три види: цільові, невизначені, нецільові. Цільовий запит означає, що цей запит на щось конкретне, а не на загальне сканування. Цільові запити можуть базуватися на певних знаннях про цю реалізацію. Невизначені запити означають, що незрозуміло, цільові вони чи ні, потрібен додатковий аналіз. З журналу проксі загальна кількість запитів становить 61 637. З журналу вузла 3 загальна кількість запитів становить 52 104. Таким чином, загальна кількість усіх запитів у honeypot становить 113 741. У таблиці 3.1 показано кількість цільових, невизначених і нецільових запитів і відповідне вираження у відсотоках. У стовпці «Цільовий?» «Так»означає, що це цільовий запит. А варіант «Ні» означає, що запит є звичайним скануванням.

Таблиця 3.1 - Розподіл запитів, класифікованих за цільовими чи ні

Цільовий	Підрахунок	Відсотоків
так	47,297	41,5%
-	10 444	9,2%
ні	56 000	49,2%
Всього	113,741	

Інша половина - це звичайне сканування до служб. Цільові запити більш цікаві, тому що це означає, що зловмисник може шукати певний розумний пристрій або деякі відомі вразливості.

Запити HTTP REST, які починаються з «/api»

Оскільки більш цікаві дані - це в основному HTTP запити REST, тоді аналіз запитів REST HTTP дає більше інформації про дії зловмисника. Запити, які починаються з «/api», швидше за все, є цілеспрямованою атакою, ніж скануванням. Оскільки всі дійсні URL-адреси, визначені приманкою api, починаються з «/api». І це ґрунтується на відомостях мосту PHD.

Таблиця 3.2 - Розподіл запитів, класифікованих за цільовим напрямком

«/api»?	Підрахунок	Відсотоків
Так	48705	42,9%
Ні	64760	57,1%
Всього	113465	

TOP tables. Тут підраховується та класифікується статистика запитів, status code (для HTTP-запитів), user agent, referrer, IP-адреса джерела та тип запиту.

У таблиці 3.3 показано 10 найпопулярніших запитів, беручи до уваги всі журнали. Деякі записи зі схожими шаблонами об’єднуються, щоб записати загальну кількість. «X» у колонці «Запит» може бути будь-яким символом.

Таблиця 3.3 - Топ-10 запитів із журналів усіх вузлів

Підрахунок	Запит
31568	«POST /api/ HTTP/1.1»
5175	«GET /sfi9876.XXXXXXX HTTP/1.1»
3014	/http:/84.19.176.29:80/PMA201X/
2984	«POST /api/list/ HTTP/1.1»
1659	«POST / HTTP/1.1»

Продовження таблиці 3.3

891	«GET / HTTP/1.1»
622	«GET http://testp3.pospr.waw.pl/testproxy.php HTTP/1.1»
520	«GET /api.XXXX/ HTTP/1.1»
342	«GET / HTTP/1.0»
280	/http:/84.19.176.29:80/mysql/admin/

Складено список URL-адрес, які починаються з «/api». Таблиця 3.4 показує 10 найпопулярніших URL-адрес, які починаються з «/api», що, цілком ймовірно, є цільовими запитами. У цій таблиці знак «-» у стовпці «Примітка» означає, що нема впевненості, що кожен запит є цільовим. Але, звичайно, це дуже можливо, оскільки вони починаються з «/api». «Цільовий» у стовбці «Примітка» означає, що це точно цілеспрямована атака. У цій таблиці можна побачити, що включено ключове слово «hue», «philips». Дуже важливими ключовими словами є «belkin» і «wemo», оскільки Wemo від Belkin є ще одним популярним брендом пристроїв для розумного дому. Ці URL-адреси показують, що ціллю зловмисників є розумні пристрої.

Таблиця 3.4 - 10 найпопулярніших URL-адрес HTTP, які починаються з «/api», взяті з журналів проксі

Підрахунок	URL	Примітка
31607	/api/	-
2984	/api/list/	-
25	/api/config	цільовий
7	/api/philips1/hue/8a92a9360fac1de8ada3903dce0795ed	цільовий
6	/api/belkin/wemo/af3bdcebd800931357951db376e0dad7	цільовий
5	/api/belkin/wemo/af3bdcebd800931357951db376e0dad7/	цільовий

Вихідні IP-адреси. IP-адреса є дуже важливим параметром для аналізу. Сама IP-адреса вже може сприяти безпеці IoT. Крім того, багато інформації можна отримати з IP-адреси, яку використовує зловмисник. Таблиця 3.5 містить список 20 найпопулярніших IP-адрес, які використовуються та реєструються проксі-сервером, включаючи всі служби.

Виявлено, що мережа TOR (платформа для анонімного спілкування [11]) використовується деякими зловмисниками, щоб приховати справжню IP-адресу джерела. IP-адреси в таблиці 3.5 перевіряються, на те, чи є вони в списку TOR. Стовпець «TOR?» показує, чи IP-адреса використовує технологію TOR. «Так» і «Ні» означають, що TOR використовується чи ні.

Таблиця 3.5 - Топ-20 вихідних IP-адрес з усіх запитів, взятих із журналів проксі

Підрахунок	IP-адреса	TOR
4982	193.70.95.180	Так
3810	93.115.95.207	Так
1669	104.223.123.98	Так
1191	163.172.101.137	Так
1060	192.42.116.16	Так
889	89.234.157.254	Так
861	79.137.79.167	Так
834	91.219.237.244	Так
736	79.172.193.32	Так
717	204.85.191.30	Так
694	94.242.246.24	Так
671	94.242.246.23	Так
669	91.223.82.156	Так
663	5.254.79.66	Так

Продовження таблиці 3.5

652	78.109.23.1	Так
622	91.196.50.33	Так
610	176.126.252.11	Так
572	216.218.222.13	Так
565	163.172.212.115	Так
564	185.170.42.4	Так

Також цікаво дізнатися, які IP-адреси найчастіше використовуються у відповідних HTTP-запитах (див. табл. 3.6).

Таблиця 3.6 - Топ-10 IP-адрес, які використовуються в запиті HTTP

Підрахунок	IP-адреса
4962	193.70.95.180
3803	93.115.95.207
1668	104.223.123.98
1191	163.172.101.137
1060	192.42.116.16
889	89.234.157.254
861	79.137.79.167
834	91.219.237.244
736	79.172.193.32
717	204.85.191.30

Зі списку IP-адрес можна отримати більше інформації. У таблиці 3.7 показано список країн, з яких походять IP-адреси. З нього видно, що сюди входять: США, Китай, Франція та Нідерланди.

Таблиця 3.7 - Топ країн, у яких розташовані IP-адреси

Підрахунок	Країна
143	HAC
67	CN
40	RF
36	NL

Статистика HTTP. Оскільки більшість запитів є HTTP-запитами. Аналіз HTTP-запитів є важливим і корисним. Важливими параметрами є: тип HTTP, status code, user agent та referrer. У таблиці 3.8 наведено кількість запитів HTTP, підрахованих за типом запиту.

Таблиця 3.8 - Кількість HTTP-запитів, взята з журналів проксі

Підрахунок	Тип HTTP
50776	HEAD
38749	POST
23708	GET
307	PUT
9	CONNECTED
4	PROPFIND
4	OPTIONS

Status Code HTTP - це код, який представляє інший статус цього запиту відповідно до api-конфігурації. Кожен код має власне визначення. У таблиці 3.9 перелічуються коди стану HTTP вхідних запитів за рангом.

User Agent є дуже важливою інформацією для аналізу даних honeypot api. Це поле запиту HTTP, яке містить інформацію про тип програми, операційну систему, постачальника програмного забезпечення або версію програмного забезпечення користувача програмного забезпечення, що надсилає запит. Таким чином, з цього поля можна отримати більше інформації

про запитувача для аналізу. Його можна розглядати як відносно унікальний ідентифікатор, який представляє шаблон у цих журналах і під час цього процесу аналізу.

Таблиця 3.9 - Кількість Status Code HTTP відповідей, взятих із журналів проксі

Підрахунок	Status Code	Примітка
34702	200	Запит виконано.
5960	499	Потрібен маркер. Це не офіційні коди. Це означає, що маркер повторно вимагався, але не був наданий.
65666	404	Потрібний ресурс (URL) не знайдено.
2477	502	Невірний шлюз. Проксі отримав недійсний запит.
2419	405	Недозволений метод. Це означає, що URL-адреса дійсна, але має неправильний тип.
2224	301	Запит має бути перенаправлено на новий URL.
320	400	Сервер не може або не обробить запит через очевидну помилку клієнта (наприклад, занадто великий розмір, недійсне фреймування повідомлення запиту або оманлива маршрутизація запиту).
85	304	Не змінено. Це перенаправлення з проксі-сервера, яке буде виконувати клієнт (api).
41	504	Час очікування шлюзу. Довірена особа не отримала своєчасної відповіді від сервера.
4	302	Знайдено. Для перенаправлення URL.
1	500	Внутрішня помилка сервера.

У таблиці 3.10 перераховано 20 користувачів для всіх вхідних запитів. У колонці «Примітка» виконується швидкий пошук, щоб швидше оцінити інформацію.

Referrer - це поле в HTTP-запиті. Ввоно визначає адресу веб-сторінки, яка посилається на запитаний ресурс. Це означає, що зловмисник може отримати доступ до ресурсу через посилання на веб-сторінці. Він містить корисну інформацію, яка може сприяти аналізу даних приманки.

Таблиця 3.10 - Запити User Agent, відповіднодо журналів проксі

Підрахунок	User Agent	Примітка
48460	Mozilla/5.0 Jorgee	Шкідлива програма, яка сканує вразливості в Інтернеті
31567	«shooter»	Інформація не знайдена
9229	«Mozilla/5.0 SF/2.10b»	Типовий агент користувача для версії Skipfish 2.10b [18]
2984	«botlight»	Інформація не знайдена
2378	«000modscan»	modscan: мережевий сканер Scada Modbus
1867	«httpget»	Інформація не знайдена
831	«Mozilla/5.0 (Windows NT 10.0; WOW64; Trident/ 7.0; rv:11.0) like Gecko»	З веб-браузера IE11 в Windows 10 OS
622	Mozilla/5.0 (Windows NT 5.1; rv:32.0) Gecko/20100101 Firefox/31.0»	З веб-браузера Firefox на Windows XP OS
607	«0000modscan»	modscan: мережевий сканер Scada Modbus
313	«Mozilla/5.0 (Windows NT 6.1; WOW64; Trident/ 7.0; rv:11.0) like Gecko»	Веб-браузер IE11 на Windows 7 OS

Продовження таблиці 3.10

275	«Mozilla/5.0 (Macintosh; Intel Mac OS X10.11;rv:47.0) Gecko/20100101 Firefox/47.0»	Веб-браузер Firefox на Mac OS X 10.11
248	ioscan	Інформація не знайдена
236	«Mozilla/5.0 (X11; U; Linux i686; en-US;rv:1.9.0.2) Gecko/2008092809 Gen Firefox/ 3.0.2»	
186	«Mozilla/5.0 (Windows NT 6.3; WOW64; Trident/7.0; rv:11.0) like Gecko»	
158	«Mozilla/5.0 (Windows NT 6.3; WOW64; Trident/7.0; Touch; rv:11.0) like Gecko»	
154	«Mozilla/5.00 (Nikto/2.1.5) (Evasions:None)(Test:map_codes)»	Nikto: інструмент веб-сканера
148	«Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko)Chrome/59.0.3071.115 Safari/537.36»	
110	«Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/ 537.36 (KHTML, like Gecko) Chrome/58.0.3029.110 Safari/537.36»	

3.3 Аналіз запитів URL-адрес і кореляції з іншими статистичними даними

Є ряд позицій, які можна отримати зі списку URL-адрес. Аналізуючи таблицю 3.4, можна помітити, що цільові запити, швидше за все, відбуваються в URL-адресі, яка починається з «/api». І є у цій таблиці URL-адреси, які містять такі ключові слова, як «Philips», «hue». Перевіривши деталі журналів, виявлено, що існує велика кількість URL-адрес, які мають дуже схожий префікс, який містить цікаве ключове слово. Хоча лише остання частина (яка є випадковим рядком фіксованої довжини) відрізняється. Таким чином, виконується подальший статистичний аналіз і результат відображається в таблиці 3.11. У цій таблиці стовпець «URL» - це не конкретна URL-адреса, а шаблон. Він підраховує кількість URL-адрес, префікс яких однаковий.

Таблиця 3.11 - URL-адреси починаються з «/api» у цільовому шаблоні, взятому з журналів api.

Підрахунок	URL
1258	/api/philips/hue
522	/api/hue
396	/api/philips2/hue-link
382	/api/belkin/wemo
306	/api/philips1/hue
4	/api/index
4	/api/resource

Можна побачити, що найпопулярніші URL-адреси в цій таблиці, очевидно, націлені на деякі ресурси - пристрої Philips Hue або Belkin Wemo. А

рядок після префікса завжди є 32-значним, який, імовірно, є шифруванням MD5. Це принаймні доводить, що зловмисники шукають розумні пристрої.

Кореляція User Agent та Status Code. У таблиці 3.12 показано список User Agent деяких цікавих URL-адрес. Він не повний, але дає можливість помітити деякі закономірності:

1) Більшість запитів «/api/» надходять від «shooter». Перевіривши кількість запитів від «shooter» (див. табл. 3.10), помітно, що всі запити «shooter» є «/api/». І майже всі успішні (200), це є результатом налаштувань honeypot.

2) Усі запити «/sfi9876.XXXXXXX» надходять від User Agent «Mozilla/5.0 SF/2.10b». Це Skipfish. З цього запиту можна зробити висновок, що зловмисник використовує Skipfish як інструмент для сканування. Більшість запитів - це 404, що означає, що під час загального сканування непросто знайти правильну URL-адресу.

3) Усі запити «/api/list/» надходять від «botlight». Перевіривши кількість запитів від «shooter» (див. табл. 3.10), видно, що всі запити «botlight» є «/api/list». Ця URL-адреса недійсна в api, тому всі коди стану - 404.

4) Є розумне припущення, що «000modscan», і навіть «curl/7.52.1» походять від однієї особи або відповідають одному шаблону, і вони надсилають подібні запити до приманки.

Подивившись на всі відповідні коди стану, видно, що більшість із них - 502, 200 і 405. Значення 502 говорить про те, що api у цей час може не працювати. А 405 означає, що URL-адреса дійсна, лише метод заборонений. Зрозуміло, що це не загальне сканування. Таким чином, він міг потрапити на дійсну URL-адресу, але метод неправильний. Також можна побачити багато значень - 200, що означає успішний запит. Кількість 200 у цих запитах набагато більша, ніж 200 у загальному скануванні. З цього видно, що цільове сканування має вищу ймовірність охоплення ресурсу.

Таблиця 3.12 - Деякі URL-адреси та їх User Agent та Status Code

Підрахунок	URL	User Agent	Status Code
31607	/api/	«shooter» : 31567 «Mozilla/5.0» : 35 wget/1.16» : 2	200:31604 502: 3
5175	/sfi9873.XXXXXXXX	«Mozilla/5.0 SF/2.10b»: 5175	404: 5384 200:2
2984	/api/list/	«botlight»: 2984	404: 2984
1258	/api/philips/hue/{32_chars}	«000modscan»: 1158 «0000modscan»: 100	502: 941 405: 216 200:100 400: 1
499	/api/hue/{0-750}	«null»:251 «ioscan»:248	
440	/api/phi/light/{32_chars}	000modscan»: 304 «0000modscan»: 100 «httpget»: 36	
396	/api/philips2/hue-link/{32_chars}	«000modscan»: 234 «0000modscan»: 100 «curl/7.52.1»: 81	405:296 200: 100 502:18
382	/api/belkin/wemo/{32_chars}	«000modscan»: 200 «0000modscan»: 106	301: 386 200: 5 400: 1
306	/api/philips1/hue/{32_chars}	«000modscan»: 200 «0000modscan»: 106	405: 200 200: 106
300	/api/tplink/light/{32_chars}	«000modscan»: 200 «0000modscan»: 100	301: 300

IP-адреси. Серед усіх вхідних запитів у трьох вузлах залучено близько 1383 різних IP-адрес. IP-адреса є досить важливим параметром для аналізу, оскільки він може давати інформацію про місцезнаходження, ASN тощо. Під час аналізу IP-адрес, якщо виявлено, що багато IP-адрес використовують мережу TOR, важливо знати, використовується технологія TOR чи ні. Якщо TOR не використовується, то IP-адресу можна розглядати як дійсну IP-адресу ресурсу. Серед IP, наведених у табл. 3.5, більшість IP використовують технологію TOR. А 91.196.50.33 і 196.54.55.13 немає в списку TOR, що означає, що вони можуть відображати дійсне джерело зловмисних дій.

Тип запиту. З таблиці 3.8 можна побачити, що POST і GET є дуже поширеними типами, які використовуватимуть зловмисники. За допомогою POST можна додавати нові елементи. І GET може отримати дані з api. Це означає, що зловмисники зацікавлені в зміні, додаванні або отриманні значень цільового api.

Кореляція з кодом статусу. Важливо проаналізувати код статусу кожного типу HTTP-запиту. У таблиці 3.13 показано код статусу кожного типу HTTP-запитів. Звідси можна побачити статус кожного виду запитів. Більшість запитів HEAD мають 404, що означає, що URL-адреса недійсна. Однак більшість запитів POST успішно виконуються зі значенням - 200.

Таблиця 3.13 - Підрахунок кожного типу запитів HTTP, взятих із журналів проксі

Підрахунок	Тип HTTP	Status Code
50776	HEAD	404:50753 200: 18 302: 4 400: 1

Продовження таблиці 3.13

38749	POST	200:31572 502: 956 404:2992 405: 2401 301: 782 400: 5
23708	GET	404:10526 499: 5960 200: 2785 502: 1512 301: 1434 304: 49 400: 55 504: 41 405: 5 500: 1
307	PUT	200: 278 404: 29
4	PROPFIND	404: 4
4	OPTIONS	404: 4

User Agent. User Agent (програмний агент, який діє від імені користувача) певним чином можна розглядати як унікальний ідентифікатор для даних. Оскільки кожен агент користувача може працювати по-різному. Але запити, які використовують ті самі агенти користувача, швидше за все, матимуть однакову продуктивність. Співвіднесення агента користувача з іншими значеннями може бути корисним для розуміння атак.

Перевіряючи таблицю 3.10, можна побачити, що багато запитів надходять від агентів користувачів, які виглядають як веб-браузер. Це означає, що ці запити можуть надсилатись з веб-браузера вручну зловмисником. Уточнивши частоту та підрахунок, можна припустити, що запити можуть надходити від деяких плагінів веб-переглядача, або зловмисник використовує цей спосіб, щоб приховати справжнього агента користувача.

Кореляція з типом запиту та кодом статусу. У таблиці 3.14 показано співвідношення між агентом користувача, типом запиту та кількістю IP. Кожен рядок відповідає одному агенту користувача з відповідною кількістю типів та IP-адрес, які використовує цей агент користувача. URL-адреса або шаблон URL-адреси агента користувача також досить інформативні.

Аналізуючи таблиці 3.12 та 3.14, а також перевіряючи деталі (URL-адреси) частини журналу, можна сформувати підсумок на основі кожного агента користувача:

1) Mozilla/5.0 Jorgee.

«Jorgee» - це зловмисне програмне забезпечення, яке намагається використати недоліки веб-адміністратора sql. Загалом він з’являється 48 460 разів із залученими 360 IP-адресами. Кожна IP-адреса надсилає лише від 200 до 400 запитів. Усі типи запитів є HEAD. URL-адреси містять ключові слова «db», «admin», «pma», «php», «sql», «web», «database» і «my». URL-адреса є перестановкою та комбінацією цих слів. Цей агент користувача з’являється лише на вузлі 3.

2) Shooter.

Інформації про shooter в інтернеті досить обмежена. Це означає, що цей агент користувача створюється вручну. Він згенерував 31567 запитів на honeypot із задіяними 92 IP-адресами. Технологія TOR використовується для приховування справжньої IP-адреси джерела. Усі URL-адреси запитів мають вигляд «/api/» з методом POST і певним основним вмістом. На рис. 3.2 показано зразок записів журналу, які містять всю відповідну інформацію цього запиту. З цього зображення можна побачити, що вміст є досить цікавим.

Основний вміст має добре організований формат, який відповідає формату структури даних Philips Hue. Повторне відтворення цього запиту виконується на реальних пристроях (Philips Hue White). Відповідь від справжнього мосту PHD - це повідомлення про помилку, яке повідомляє, що параметр недоступний.


{"body": "{\"groups\":{\"2\":{\"state\":{\"all_on\":\"true\"},\"action\":{
lse\"}},\"1\":{\"state\":{\"all_on\":\"true\"},\"action\":{\"on\":\"false\
hts\":{\"1\":{\"state\":{\"bri\":\"false\",\"on\":\"true\",\"reachable\":\
\"bri\":\"true\",\"on\":\"true\",\"reachable\":\"true\"}}},\"sensors\":{\"
lse\"}}}}", "header": {"CONNECTION": "close", "HOST": "morris.jusanet.org"
161", "USER_AGENT": "shooter", "ACCEPT": "*/*"}, "entry_id": 34604, "time"
l": "/api/", "remote_ip": "172.16.10.2", "type": "POST", "reply_content":
e0s5cJTufws9IaF3PTggBwQsvb3WR685EHMqcwP"}}]}

Рисунок 3.2 - Один із записів журналу, User Agent якого є shooter

3) Mozilla/5.0 SF/2.10b.

Це типовий агент користувача для Skipfish [18]. Задіяно 7 IP. Всього 9229 запитів. Усі запити: 20115 GET, 47 PUT і 47 FOO.

Skipfish - це інструмент сканування безпеки веб-додатків. Він був розроблений для перевірки безпеки, але ним також можуть скористатися зловмисники.

4) Botlight.

Від цього агента користувача надійшло 2984 запити. Усі запити є запитами POST з однаковою URL-адресою «/api/list/». Задіяно 21 IP-адресу, а технологія TOR використовується для приховування справжньої IP-адреси джерела. На рисунку 3.3 показано один із записів журналу, агент користувача якого «botlight». Цікаво, що вміст тіла запитів слідує за multipart/form-data, які використовуються для завантаження файлів. Але також є багато % і 0, що виглядає як розмитість.

Рисунок 3.3 - Один із записів журналу, User Agent якого є botlight

5) 000modscan.

000mods може мати лише запит POST із задіяними 12 IP-адресами. URL-адреса відповідає шаблону, показаному в таблиці 5.12: /api/philips/hue/{32_chars}, /api/philips2/hue-link/{32_chars}, /api/belkin/wemo/{32_chars} і /api/philips1/hue/{32_chars}. Ці запити POST мають досить цікавий вміст, який схожий на «botlight». Один зразок запиту показаний на рис. 3.4. На цьому зображенні видно, що цей вміст тіла також відповідає multipart/form-data. Крім того, у вмісті частина слів пов’язана з пристроєм Philips Hue.

Рисунок 3.4 - Один із записів журналу, User Agent якого 000modscan

6) httpget.

Усі запити, агентом користувача яких є httpget, є HTTP GET із залученими 7 IP-адресами. URL-адреси мають три шаблони: /api/phi/light/{32_chars}/tokens, /api/{32_chars}/tokens і /api/{32_chars}.

Шаблон {32_chars} означає випадковий рядок довжиною 32, який містить лише цифри та малі літери. Оскільки URL-адреси за шаблоном /api/32_chars/tokens, можна припустити, що всі запити з «httpget» надходять з одного джерела. Це можна розглядати як цільову атаку.

7) «Mozilla/5.0(WindowsNT5.1;rv:32.0)Gecko/20100101 Firefox/31.0».

Всі запити є HTTP GET із задіяними 14 IP-адресами. Тоді як більшість запитів надходить з IP 91.196.50.33. URL-адреса більшості запитів -

/http:/testp3.pospr.waw.pl/testproxy.php. Це URL-адреса, яка шукає відкриті проксі-сервери.

8) 0000modscan.

Усі запити, агентом користувача яких є «0000modscan», є GET-запитами з 12 IP-адресами. URL-адреса запитів: /api/tplink/light/{32_chars}, /api/philips/hue/{32_chars}, /api/phi/light/{32_chars}, /api/philips2/hue-link/{32_chars}, /api/belkin/wemo/{32_chars},/api/philips1/hue/{32_chars}. Це цільове сканування HTTP GET, націлене на розумні світлові пристрої.

9) «Mozilla/5.0 (Macintosh; Intel Mac OS X10.11; rv:47.0) Gecko/20100101 Firefox/47.0».

Запити лише з двох IP: 183.129.160.229 (193 рази) і 60.191.38.77 (82 рази). Усі вони GET з URL. Цей агент користувача означає, що він походить від операційної системи Mac із веб-браузером Firefox. Перевіряючи час запитів, він, швидше за все, буде надісланий вручну. Таким чином, можна припустити, що це хтось, хто знайшов проксі та вручну запитав ресурс даних.

10) ioscan.

Усі запити, агентом користувача яких є «ioscan», є HTTP GET із задіяними 2 IP-адресами. Усі URL-адреси відповідають шаблону /api/hue/{0-216}, де {0-216} - це число в діапазоні від 0 до 216. Це можна розглядати як цільове сканування, оскільки «hue» є ключовим словом для honeypot.

11) Python-urllib/2.7.

Python-urllib/2.7 - це бібліотека Python для отримання даних у всесвітній мережі. Усі запити, агентом користувача яких є «Python-urllib/2.7», є HTTP GET із залученими 3 IP-адресами. Кількість запитів - 94, і всі вони надійшли з одного IP (185.77.172.42). З цього можна побачити, що повинні бути запущені сценарії, а не вручну від одного зловмисника, який намагається просканувати honeypot. Перевіривши URL-адресу цих запитів, можна виявити, що URL-адреса досить випадкова, але загалом шукає вразливості веб-адміністратора sql.

Таблиця 3.14 - Найпопулярніші User Agent за всіма запитами, взятими з журналів проксі

Підрахунок	User Agent	Тип запиту	IP-адреса
48460	Mozilla/5.0 Jorgee	HEAD: 48460	360
31567	«shooter»	POST: 31567	92
9229	«Mozilla/5.0 SF/2.10b»	GET: 9171 PUT: 29 FOO: 29	7
2984	«botlight»	POST: 2984	20
2378	«000modscan»	POST: 2378	12
1867	«httpget»	GET: 1867	7
622	«Mozilla/5.0 (Windows NT 5.1; rv:32.0) Gecko/20100101 Firefox/31.0»	GET: 622	14
607	«0000modscan»	GET: 607	12
275	«Mozilla/5.0 (Macintosh; Intel Mac OS X	GET: 275	2
248	ioscan	GET: 248	2
96	Python-urllib/2.7	GET: 96	3

Корисний вміст. Під час аналізу корисного вмісту, можуть бути знайдені цікаві дані, наприклад, випадкові рядки для фаззингу, корисний вміст для певної мети, пов’язаної з конкретним пристроєм тощо). На рисунку 3.5 показано зразок записів журналу цих запитів.

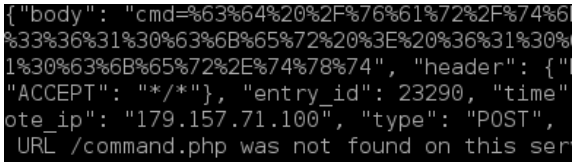

Рисунок 3.5 - Один із записів журналу який містить основний вміст «testPost=true»

У запитах, тіло яких «test-Post=true», URL-адреса /http:/check.proxyradar.com/azenv.php, а referer - https://proxyradar.com/. Це інструмент для пошуку відкритих проксі.

Можна проаналізувати закодований запит (див. рис. 3.6). Його вміст, згідно з , це пошук вразливих маршрутизаторів в Інтернеті.

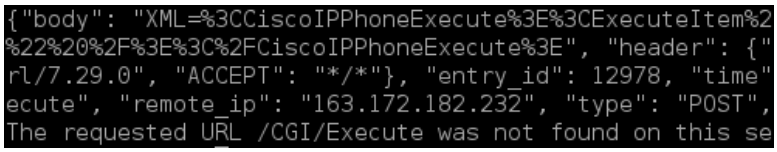

Рисунок 3.6 - Один із записів журналу, який містить основний вміст «cmd=%63%64...»

На рисунку 3.7 показано закодований запит, основний вміст якого схожий на зловмисне програмне забезпечення, яке намагається отримати доступ до phpmyadmin/scripts/setup.php .

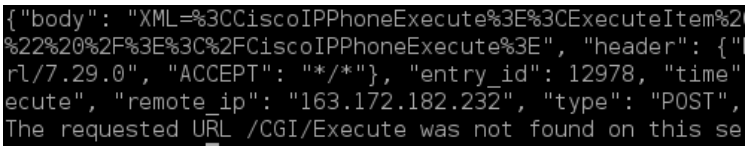

Рисунок 3.7 - Один із записів журналу, який містить «XML=%3CCiscoIPPhoneExecute...»

3.4 Аналіз результатів обробки даних

Після збору інформації з журналів можна зробити висновки, щоб узагальнити інформацію, наведену в останній частині роботи. Атаки можуть бути перераховані відповідно до кореляцій. Однак не можна гарантувати повний список атак, знайдених за допомогою цієї приманки.

Цілеспрямована атака, яка намагається взяти під контроль

Знайдено одну цілеспрямовану атаку. Атакою є запит HTTP POST із певним тілом (body) HTTP. Body HTTP - це дані у форматі json, які відповідають формату справжньої відповіді мосту Philips Hue. Один із прикладів показано на рис. 3.2. В інструкції Philips Hue api сказано, що для керування пристроєм Hue на міст Philips Hue слід надіслати POST-запит із тілом json і дійсною URL-адресою. Можна побачити, що ця атака симулює запит на зміну значення мосту Philips Hue. Це цілеспрямована атака, і аказує на те що зловмисник уже отримав інформацію про цю реалізацію (знаючи, що йдеться про Philips Hue) і метод керування Philips Hue.

Атака у форматі multipart/form-data. Це атака, яка виконується через HTTP POST із певним основним вмістом. Як показано на рисунках 3.3 і 3.4, основний вміст HTTP POST відповідає дуже подібним шаблонам. Таким шаблоном є: form-data; name=\»productid\»\r\n\r\n{random_payload}\r\n-{16_chars}–\r\n», де 16_chars} - випадковий рядок із 16, що містить лише малі літери та цифри; {random_payload} - це щось випадкове, де «botlight» заповнюється «%0000», а «000modscan» не заповнюється нічим.

Інший масовий агент користувача також надає POTS-запити, які відповідають цьому шаблону, {random_payload} який заповнюється надзвичайно довго повторюваним (9944 рази в одному корисному вмісті) «%A/telnet» або «%A/xmpp» або «% A/upnp». Це очевидно є розмиттям. URL-адреса цієї атаки також відповідає шаблону, який містить ключове слово (philips, hue, wemo, belkin, device, token тощо) і випадковий рядок довжиною 32.

Підсумком цієї атаки може бути те, що це цілеспрямована атака, яка шукає розумні пристрої, служби xmpp, upnp або telnet.

Атака за допомогою URL

Цей вид атаки здійснюється через HTTP GET з URL-адресами, які відповідають певному шаблону:

- /api/philips/hue/{32_chars};
- /api/phi/light/{32_chars};

- /api/philips1/hue/{32_chars};

- /api/philips2/hue-link/{32_chars};

- /api/belkin/wemo/{32_chars};

- /api/tplink/light/{32_chars} 7./api/hue/{0-750};

- /api/phi/light/{32_chars}/tokens;

- /api/{32_chars}/tokens;

- /api/{32_chars}/

Так, {32_chars} означає рядок довжиною 32, який містить малі літери та цифри у випадковому порядку. І {0-750} означає число в діапазоні від 0 до 750. Усі ці запити є цільовими атаками сканування.

Загальні засоби та бібліотеки сканування. За даними honeypot можна знайти ряд інструментів сканування та бібліотек. Інструменти або бібліотеки сканування тут означають те, що вони не орієнтовані на розумні пристрої:

- skipfish;

- Nikto;

- Jorgee: за поведінкою запиту, цю атаку можна вважати Jorgee, яка є веб-сканером;

- masscan: це також можна знайти у списку агентів користувачів, де адреса вихідного коду також міститься в заголовку;

- Бібліотека Python: urllib;

- /http:/testp3.pospr.waw.pl/testproxy.php: може бути показана IP-адреса проксі у мережі користувача;

- Proxyradar: на https://proxyradar.com/ користувач можете знайти відкриті проксі.

ВИСНОВКИ

В кваліфікаційній роботі розв’язано актуальну задачу підвищення ефективності алгоритмів виявлення атак на Інтернет речей з використанням технології приманок. При цьому отримано наступні результати.

У першому розділі проаналізовано підходи до захисту Інтернет речей.

У другому розділі проаналізовано методи виявлення атак на інтернет речі і їх аналіз.

У третьому розділі проаналізовано вихідні данні по приманкам у мережі інтернет речей. Атаки можуть бути перераховані відповідно до кореляцій. Однак не можна гарантувати повний список атак, знайдених за допомогою цієї приманки.

На основі проведених досліджень сформульовано методи запобігання атакам на платформи IoT:

1. Шифрування критичного корисного навантаження HTTP REST може бути зашифровано. Слід покращити метод автентифікації. Наприклад, білий список мосту Philips Hue має бути зашифрований, щоб уникнути викрадення зловмисниками.

2. Більш того, замість звичайного тексту, якщо SSDP-пакети notify зашифровані, розташування мосту не буде розкрито.

3. Хоча комунікаційна мережа XMPP платформи IoT працює більш структуровано, і дії зловмисників не виявляються в IoTPot, все одно важливо звернути увагу на безпеку XMPP. Необхідний захист облікової інформації. Крім того, набір даних, згенерований honeypot, можна використовувати для створення сигнатур IDS, щоб виявити подібні атаки на платформи та інші загальні REST- сканування пристроїв Інтернету речей.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Matthew Garrett. A quick look at the ikea trådfri lighting platform. http://mjg59.dreamwidth.org/47803.html.

2. Jorge Granjal, Edmundo Monteiro, and Jorge Sá Silva. Security for the internet of things: a survey of existing protocols and open research issues. IEEECommunications Surveys&Tutorials, 17(3):1294–1312, 2015.

3. Kevin Ashton. That ‘internet of things’ thing. RFiD Journal, 22(7):97–114, 2009.

4. ITU Strategy and Policy Unit (SPU). The internet of things-executive summary. 2005. doi:http://www.itu.int/osg/spu/publications/internetofthings/InternetofThings_summary.pdf.

5. Ala Al-Fuqaha,Mohsen Guizani,MehdiMohammadi,Mohammed Aledhari, andMoussa Ayyash. Internet of things: A survey on enabling technologies, protocols, and applications. IEEE Communications Surveys & Tutorials, 17(4):2347–2376, 2015.

6. Zhengguo Sheng, Shusen Yang, Yifan Yu, Athanasios Vasilakos, JulieMccann, and Kin Leung. A survey on the ietf protocol suite for the internet of things: Standards, challenges, and opportunities. IEEEWireless Communications, 20(6):91–98, 2013.

7. Luigi Atzori, Antonio Iera, and GiacomoMorabito. The internet of things: A survey. Computer networks, 54(15):2787–2805, 2010.

8. Vasileios Karagiannis, Periklis Chatzimisios, Francisco Vazquez-Gallego, and Jesus Alonso-Zarate. A survey on application layer protocols for the internet of things. Transaction on IoT and Cloud Computing, 3(1):11–17, 2015. URL http://icas-pub.org/ojs/index.php/ticc/article/view/47.

9. HIVEMQ. Mqtt 101 – how to get started with the lightweight iot protocol. http://www.hivemq.com/blog/how-to-get-started-with-mqtt.

10. XMPP-IOT. Python read and write. http://www.xmpp-iot.org/tutorials/python-tutorial/.

11. RAY LAPENA. More than 90with iiot in 2017. https://www.tripwire.com/state-of-security/featured/90-pros-expect-attacks-risk-vulnerability-iiot-2017/.

12. Linda Markowsky and George Markowsky. Scanning for vulnerable devices in the internet of things. In Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS), 2015 IEEE 8th International Conference on, volume 1, pages 463–467. IEEE, 2015.

13. LILY HAY NEWMAN. The web-shaking mirai botnet is splintering—but also evolving. https://www.wired.com/2016/11/web-shaking-mirai-botnet-splintering-also-evolving/.

14. Silva Granjal, Monteiro. Security for the internet of things: a survey of existing protocols and open research issues. IEEE Communications Surveys & Tutorials, 17:1294 – 1312, 2015. doi: 10.1109/COMST.2015.2388550.

15. Oscar Garcia-Morchon, Sandeep Kumar, Rene Struik, Sye Keoh, and Rene Hummen. Security considerations in the ip-based internet of things. draft-garcia-core-security-04, 2013.

16. Kim Thuat Nguyen, Maryline Laurent, and Nouha Oualha. Survey on secure communication protocols for the internet of things. Ad Hoc Networks, 32:17–31, 2015. doi: http://dx.doi.org/10.1016/j.adhoc.2015.01.006.

17. Yong Wang, Garhan Attebury, and Byrav Ramamurthy. A survey of security issues in wireless sensor networks. IEEE Communications Surveys & Tutorials, 2006.

18. Gang Gan, Zeyong Lu, and Jun Jiang. Internet of things security analysis. In Internet Technology and Applications (iTAP), 2011 International Conference on, pages 1–4. IEEE, 2011.

19. Martina Brachmann, O Garcia-Mochon, Sye-Loong Keoh, and Sandeep S Kumar. Security considerations around end-to-end security in the ip-based internet

of things. InWorkshop on Smart Object Security, in conjunction with IETF83, Paris, France,March 23, 2012, 2012.

20. Darknet. Xmpploit – a tool to attack xmpp connections. http://www.darknet.org.uk/2012/08/xmpploit-a-tool-to-attack-xmpp-connections/.

21. Meena Singh, MA Rajan, VL Shivraj, and P Balamuralidhar. Secure mqtt for internet of things (iot). In Communication Systems and Network Technologies (CSNT), 2015 Fifth International Conference on, pages 746–751. IEEE, 2015.

22. Gadde, Sai & Ganta, Rama & Gupta, ASALG & K, Raghava & Rao, KRR. (2018). Securing Internet of Things(IoT) Using HoneyPots. International Journal of Engineering & Technology. 7. 820. 10.14419/ijet.v7i2.7.11075.

23. Dinesh Thangavel, Xiaoping Ma, Alvin Valera, Hwee-Xian Tan, and Colin Keng-Yan Tan. Performance evaluation of mqtt and coap via a common middleware. In Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP), 2014 IEEE Ninth International Conference on, pages 1–6. IEEE, 2014.

24. Joel L Fernandes, Ivo C Lopes, Joel JPC Rodrigues, and Sana Ullah. Performance evaluation of restful web services and amqp protocol. In Ubiquitous and Future Networks (ICUFN), 2013 Fifth International Conference on, pages 810–815. IEEE, 2013.

25. Zach Shelby, Klaus Hartke, and Carsten Bormann. The constrained application protocol (coap). 2014.

26. Carsten Bormann, Angelo P Castellani, and Zach Shelby. Coap: An application protocol for billions of tiny internet nodes. IEEE Internet Computing, 16(2):62–67, 2012.

27. Хомицький А.А. Аналіз існуючих приманок для виявлення атак на інтернет речей. Матеріали наукової конференції «Автоматизація та комп’ютерно-інтегровані технології» (АКІТ – 2022) Тернопіль, 2022. С. 113-114.

28. Хомицький А.А. Аналіз по категорії приманок для виявлення атак на інтернет речей. Матеріали наукової конференції «Кібербезпека та компютерно-інтегровані технології» (КБКІТ-2022) Тернопіль, 2022. С. 51-52.

ДОДАТОК А

Копії публікацій