Хмельницький національний університет
Факультет інформаційних технологій
Кафедра комп'ютерної інженерії та інформаційних систем

КВАЛІФІКАЦІЙНА РОБОТА

бакалавр
Освітній рівень

Програмний засіб моніторингу мережевого трафіку пристроїв Інтернету речей
Назва теми

КвРКІ. 200227.02.03 ПЗ
Шифр

Галузь знань *12 «Інформаційні технології»*
Шифр, назва

Спеціальність *123 «Комп'ютерна інженерія»*
Шифр, назва

Освітня програма *«Комп'ютерна інженерія та програмування»*
Назва

Виконав: студент IV курсу, група КІ2-20-2 _____ В. А. Басистий
Підпис, Ініціали, прізвище

Керівник _____ М. О. Слободян
Підпис, дата, Ініціали, прізвище

Нормоконтролер _____ С. М. Лисенко
Підпис, дата, Ініціали, прізвище

До захисту допускаю:
Зав. кафедри комп'ютерної інженерії та інформаційних систем _____ Т.О. Говорущенко
Підпис, Ініціали, прізвище

«24» червня 2024 р.

Хмельницький 2024

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет Інформаційних технологій

Кафедра Комп'ютерної інженерії та інформаційних систем

Освітній рівень бакалавр

Галузь знань 12 Інформаційні технології

Спеціальність 123 Комп'ютерна інженерія

Освітня програма «Комп'ютерна інженерія та програмування»

ЗАТВЕРДЖУЮ

Зав. кафедри Т.О.Говорущенко

" 10 " 01 2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ БАКАЛАВРА

Басистому Віталію Анатолійовичу

Прізвище, ім'я, по батькові студента

1. Тема проєкту (роботи) Програмний засіб моніторингу мережевого трафіку пристроїв Інтернету речей

Керівник проєкту (роботи) Слободян М.О.

Прізвище, ім'я, по батькові, науковий ступінь, вчене звання

Затверджена наказом ректора університету від 15.02.2024 р. № 8

2. Строк подання студентом проекту (роботи) на кафедру 01.06.2024 р.

3. Вихідні дані до проекту (роботи) Завдання на кваліфікаційну роботу

4. Зміст пояснювальної записки (перелік питань, які потрібно розробити)

Дослідження предметної області та постановка задачі кваліфікаційної роботи

Проєктування програмного засобу моніторингу мережевого трафіку пристроїв Інтернету речей.

Реалізація та тестування програмного засобу моніторингу мережевого трафіку пристроїв Інтернету речей

5. Перелік графічного матеріалу (із зазначенням обов'язкових креслень)

Схема мережі IoT

Модель варіантів використання програмного засобу

Схема бази даних програмного засобу

6. Консультанти розділів дипломного проекту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Нормоконтроль	Лисенко С.М., професор кафедри КІІС		
Антиплагіат	Нічепорук А. О., доцент кафедри КІІС		

7. Дата видачі завдання «_10_»_____01_____2024 р.

КАЛЕНДАРНИЙ ПЛАН

№з/п	Назва етапів (розділів) дипломного проекту (роботи)	Термін виконання етапів проекту (роботи)	Примітка
1	Вибір напряму дослідження та узгодження тематики кваліфікаційної роботи з керівником	10.01.2024	виконано
2	Ознайомлення з предметною областю; формулювання мети та задач дослідження; визначення об'єкта та предмета дослідження	01.02.2024	виконано
3	Робота над розділом 1 – дослідження предметної області та постановка задачі кваліфікаційної роботи	01.03.2024	виконано
4	Робота над розділом 2 – проєктування програмного засобу моніторингу мережевого трафіку пристроїв Інтернету речей	01.04.2024	виконано
5	Робота над розділом 3 – реалізація та тестування програмного засобу моніторингу мережевого трафіку пристроїв Інтернету речей	29.04.2024	виконано
6	Оформлення пояснювальної записки згідно вимог	25.05.2024	виконано
7	Попередній захист ВКР	26.05.2024	виконано
8	Захист ВКР на засіданні ЕК	24.06.2024	

Студент _____ В. А. Басистий
Підпис Ініціали, прізвище

Керівник роботи _____ М. О. Слободян
Підпис Ініціали, прізвище

№ рядка	Формат	Позначення	Найменування	Кіл. листів	№ екз.	Примітки
			Текстові документи			
1	А4	КвРКІ. 200227.02.03 ПЗ	Пояснювальна записка	56		
			Графічні матеріали			
2	А4	КвРКІ. 200227.02.03 Е8	Модель варіантів			
			використання	1		Копія
						А2
3	А4	КвРКІ. 200227.02.03 Е8	Схема мережі	1		Копія
						А2
4	А4	КвРКІ. 200227.02.03 Е8	Схема бази даних			Копія
						А2

КвРКІ. 200227.02.03 ВР

Зм	Арк	№ докум	Підпис	Дата
Розробив		Басистий В.А		
Перевір.		Слободян М.О		
Н. контр.		Лисенко С.М		
Затв.		Говорущенко І.Ю		24.06

Програмний засіб моніторингу мережевого трафіку пристроїв Інтернету речей

Відомість роботи

Літера	Аркуш	Аркушів
У	1	1

ХНУ, КІ2-20-2

АНОТАЦІЯ

Тема кваліфікаційної роботи: «Програмний засіб моніторингу мережевого трафіку пристроїв Інтернету речей».

Автор роботи: Басистий Віталій Анатолійович.

Керівник роботи: Слободян Максим Олегович.

Пояснювальна записка: 56 с., 22 рис., 6 табл., 5 дод., 47 джерел.

Графічна частина: 3 плакати.

Ключові слова: мережевий трафік, програмний засіб, Інтернет речей, база даних, моніторинг.

Кваліфікаційна робота присвячена вирішенню технологічної задачі щодо підвищення ефективності адміністрування систем на основі Інтернету речей шляхом автоматизації моніторингу мережевого трафіку.

Метою роботи є розробка програмного засобу моніторингу мережевого трафіку, який підвищує ефективність роботи системного адміністратора щодо підтримки застосувань Інтернету речей.

В ході виконання роботи було проведено аналіз вимог до програмного засобу моніторингу мережевого трафіку пристроїв Інтернету речей; дано загальну характеристику та розробити структурно схеми кінцевого пристрою IoT на базі розумного сенсору; розроблено схему мережі IoT, де відображено підключення IoT-пристроїв до локальної мережі та до мережі Інтернет; розроблено модель даних та схему бази даних для моніторингу трафіку; розроблено програмний засіб для отримання параметрів трафіку Інтернет-пристроїв з подальшим записом їх в базу даних.

Підпис студента

19.06.2024

Дата

ЗМІСТ

ВСТУП

Інтернет речей всебічно проникає в усі сфери діяльності людей, цифровізуючи та автоматизуючи роботу повсякденних пристроїв та вимірювальних приладів, а також засобів контролю для зручності користувачів. Будучи варіацією комп'ютерної мережі, де в якості кінцевих пристроїв виступають малопотужні обчислювальні модулі під керуванням спеціалізованих операційних систем та програмного забезпечення, такі системи та мережі потребують належного адміністрування. Так, зокрема моніторинг і аналіз мережевого трафіку є важливими компонентами процесу адміністрування та підтримки системи з метою забезпечення безпеки, продуктивності та надійності. Такі інструментальні засоби дозволяють системному адміністратору виявляти і усувати різного роду проблеми, які можуть виникати на різних рівнях функціонування системи, зокрема – на мережевому рівні та рівні додатків.

Актуальність кваліфікаційної роботи полягає у вирішення технологічної задачі щодо підвищення ефективності адміністрування систем на основі Інтернету речей шляхом автоматизації моніторингу мережевого трафіку.

Метою роботи є розробка програмного засобу моніторингу мережевого трафіку, який підвищує ефективність роботи системного адміністратора щодо підтримки застосувань Інтернету речей. Для досягнення поставленої мети необхідно вирішити такі задачі:

1. Провести аналіз вимог до програмного засобу моніторингу мережевого трафіку пристроїв Інтернету речей.

2. Дата загальну характеристику та розробити структурно схеми кінцевого пристрою IoT на базі розумного сенсору.

3. Розробити схему мережі IoT, де відобразити підключення IoT-пристроїв до локальної мережі та до мережі Інтернет.

4. Розробити модель даних для моніторингу трафіку.

5. Виконати програмну реалізацію компонентів програмного засобу моніторингу трафіку для Інтернету речей.

6. Виконати розгортання та тестування програмного засобу.

1 ДОСЛІДЖЕННЯ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ КВАЛІФІКАЦІЙНОЇ РОБОТИ

1.1 Аналіз предметної та обґрунтування актуальності теми роботи

В застосуваннях Інтернету речей (Internet of Things, IoT) моніторинг і аналіз мережевого трафіку є важливими компонентами процесу адміністрування та підтримки системи з метою забезпечення безпеки, продуктивності та надійності. Такі інструментальні засоби дозволяють системному адміністратору IoT-системи виявляти і усувати різного роду проблеми, які можуть виникати на різних рівнях функціонування системи, зокрема – на мережевому рівні та рівні додатків.

В результаті аналізу літературних джерел згідно предметної області [1-4] були виділені такі ключові аспекти моніторингу мережевого трафіку IoT:

- моніторинг на рівні мережі: використовується для слідкування за загальною активності мережі IoT (обсяг трафіку, типи протоколів, IP-адреси пристроїв);
- моніторинг на рівні пристроїв: використовується, для відстеження трафіку, який направляє кожний IoT-пристрій з метою детектування аномальної поведінки, діагностування проблем з пристроями, тощо;
- моніторинг на рівні додатків: стеження за мережевою активністю прикладних додатків користувача, а також системних утиліт, що генерують трафік, з метою визначення порушень в роботі IoT пристроїв викликаних відмовами програмного забезпечення (ПЗ);
- моніторинг на рівні безпеки: комплекс заходів протидії кіберзагрозам, в тому числі відстеження підозрілої активність, несанкціонованого вторгнення в мережу і виявлення шкідливе програмне забезпечення (ШПЗ);

- моніторинг на рівні користувача: відстеження взаємодії користувача з IoT-пристроями та додатками, аналіз забезпечення безпеки та конфіденційності даних, що використовуються користувачем;

- моніторинг на рівні мережевих послуг: оцінка якості і доступності мережевих послуг IoT, наприклад, аналіз часу відгуку, швидкості передачі даних і рівень доступу користувача.

Зважаючи на вищеописану класифікацію, актуальним узагальнюючим критерієм для програмного засобу моніторингу трафіку IoT є функції контролю мережевої активності кінцевих IoT-пристроїв, а саме кількість переданих та прийнятих кілобайт даних та швидкість передачі із деталізацією за пристроями, мережевими інтерфейсами та процесами.

Серед причин аномальної активності мережевих пристроїв IoT є як внутрішні відмови апаратних та програмних компонентів, так і зовнішній фактори, зокрема кібератаки та загрози втручання в систему з метою порушення її нормальної роботи. Виділено такі типи кіберзагроз:

- атака ботнетів [5];

- атака на відмову в обслуговуванні (distributed denial-of-service attack, DDoS); [6]

- спам та фішинг через кінцеві пристрої IoT; [7]

- атака через прошивку(оновлення вбудованого ПЗ); [8]

- атака на інфраструктуру IoT і внутрішні системи; [9]

- витік, маніпулювання і пошкодження даних; [10]

- атаки у вигляді «людина посередині». [11]

З метою систематизації та структурування найбільш типових загроз IoT, був проведений аналіз проблеми, сформований перелік загроз і основні жертви цих загроз.

Атака ботнетів полягає у тому, що зловмисники заражають велику кількість Іото-пристроїв засобами ШПЗ з метою ініціації розподіленої мережі заражених пристроїв – ботнету. Після накопичення певної кількості таких пристроїв,

зловмисники можуть централізовано здійснювати DDoS атаки, викрадення даних, навантаження мережі, відволіканням уваги, а також інші злочинні дії. Об'єктами атаки ботнетів зазвичай є особисті дані людей, банківська інформація та інша конфіденційна інформація, що уражена до такого типу атак.

На рисунку 1.1 схематично показано принцип атаки хакера засобами ботнету і різні види загроз, які можуть бути реалізовані, як окремо, так і одночасно на об'єкт атаки.

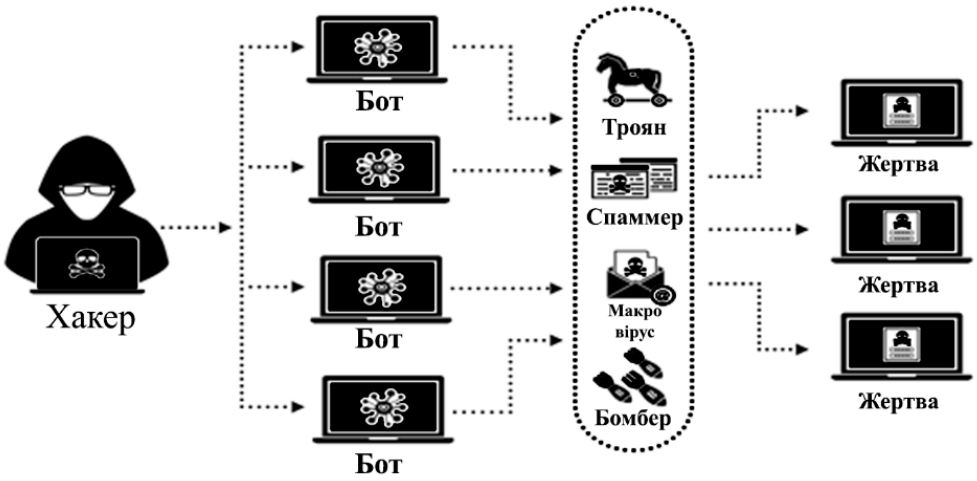


Рисунок 1.1 – Схема атаки ботнету з різними видами загроз

До найвідоміших атак ботнетів можна віднести Conficker [12], який активно поширювався в 2008-2009 роках. Цей ботнет використовував різні методи зараження, включаючи вразливості в операційних системах (ОС) Windows з метою розсилки спаму, крадіжки паролів і запуску DDoS-атак.

Іншим прикладом відомого ботнету є Mirai [13], що відомий своєю роллю в DDoS-атаках, включаючи атаку на хостинг-постачальника Dyn у 2016 році, яка призвела до відключення популярних веб-сайтів, таких як Twitter, Netflix і Reddit. Mirai використовував незахищені Інтернет пристрої для створення ботнету.

Суть DDoS атаки на пристрій, систему або мережу, полягає у перешкоджанні доступу для користувача або повної відмови IoT пристроїв та інфраструктури. Такий тим атак часто блокує можливість доступу з метою

вимагання оплати за поновлення доступу. Основними об’єктами DDoS атак, є користувачі ОС, веб-сайти, онлайн-сервіси, промислові та транспортні мережі та системи.

На рисунку 1.2, показана схема атаки через клієнти системи доменних імен (Domain Name System, DNS). Хакер керує через бот-машину DNS-клієнтами і надає можливість блокувати або зупиняти роботу цілі.

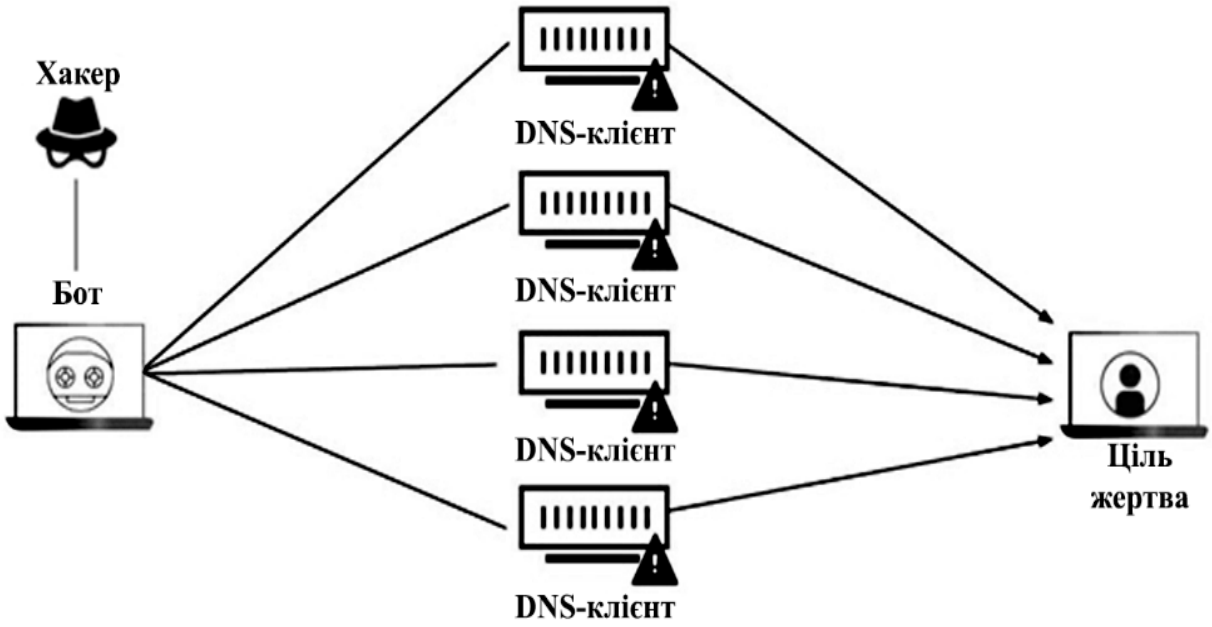


Рисунок 1.2 – Приклад DDoS атака через DNS

Відомі такі приклади найбільш руйнівних DDos атак:

1. Атака на Github, яка була здійснена 28 лютого 2018 року. Платформа була атакована масовими запитами, через що адміністраторам довелось запросити підтримку в Akami Prolexic – сервісу, який знешкоджує шкідливі запити і вирішує подібні проблеми [14].

2. Кібератака на Естонію в 2007 році, яка вивела з ладу сайти парламенту Естонії, міністерств і засоби масової інформації. На момент 2007 року, більше 90% банківських операцій були пошкоджені, не оброблені або мали інші проблеми [15].

Спам і фішинг через IoT пристрої часто комбінується з атакою ботнетів маючи на меті масово атакувати всіх користувачі мережі. Зловмисники використовують текстові повідомлення, електрону пошту та інші канали ураження з метою доставка користувачеві небезпечних файлів. Атаки спаму і фішингу зазвичай направлені на недосвідчених користувачів мережі задля крадіжки особистих даних, номерів банківських карток, паролів тощо. Спам-повідомлення і спам-боти, які надсилають користувачам недостовірні повідомленнями із сумнівним контентом з метою шахрайства.

На рисунку 1.3 показано приклад спам-бота, який постійно надсилає повідомлення користувачам системи Телеграм і може займатись фішингом, щоб отримати дані жертви.

Прикладом відомої фішинг атаки є атака на Microsoft Office 365 в 2021 році. На електрону пошту надходив лист про прохання надати дані, для авторизації. Коли користувач надавав інформацію і заходив на фейк-сайт, шахраї отримували облікові записи користувачів. [16]

Вразливість кінцевих IoT-пристроїв до атака через прошивку може виникати якщо користувач довгий час не оновлював компоненти ПЗ захисту, що може стати причиною зламу таких систем кіберзлочинцями. Цей тип атаки відбувається ззовні і зсередини, щоб було легше отримати доступ до жертви. Пристрій жертви відкривається для зловмисників через офіційне програмне забезпечення, що ускладнює захист через той факт, що загроза була встановлена ще на початку введення пристрою в експлуатацію.

Основними об’єктами таких атак, є користувачі, які не оновлювали ПЗ протягом довгого періоду часу, телефонні оператори, інтернет провайдери і особисті речі користувача, з яких можна отримати інформацію. Наприклад, телефон, смарт-годинник, комп’ютер, тощо.

На рисунку 1.4 показано атаку на ядро і бази даних (БД) компаній, мобільний зв’язок, персональні комп’ютери (ПК) користувачів та мережі.

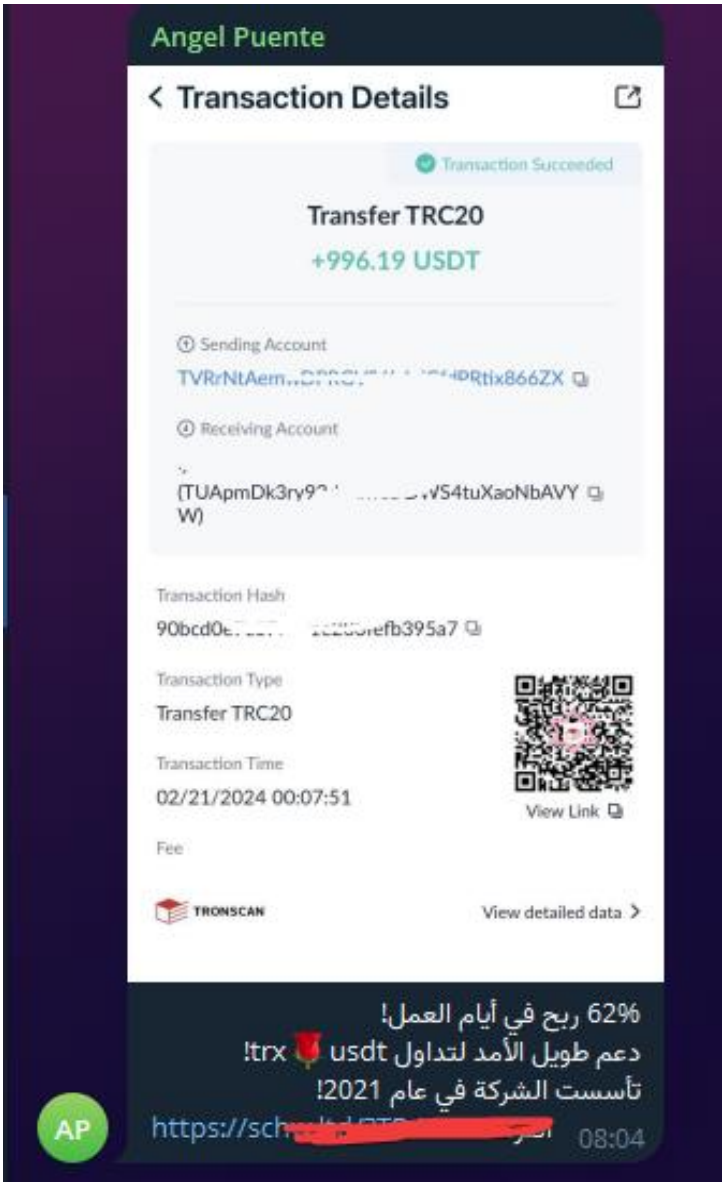


Рисунок 1.3 – Спам-бот в системі Телеграмі

Відомою атакою даного типу є атака на постачальника програмного забезпечення SolarWinds [17]. В грудні 2020 року, хакери отримали доступ до прошивки Orion і до систем багатьох організацій, включаючи Уряд США, FireEye, міністерства торгівлі, міністерства фінансів, міністерства національної безпеки та інші компанії, які користувались цим програмним забезпеченням.

Атаки на інфраструктуру IoT і внутрішні системи направлені на сервери, хмарні платформи, мережеві пристрої в результаті обходу механізмів захисту. В багатьох випадках ціллю зловмисників є ядро системи, БД або інші мережеві пристрої, які мають доступ до критичних вузлів системи. Жертвами таких атак

стають державні системи, медичні системи, системи керування, сервери де розміщені веб-сайти тощо.

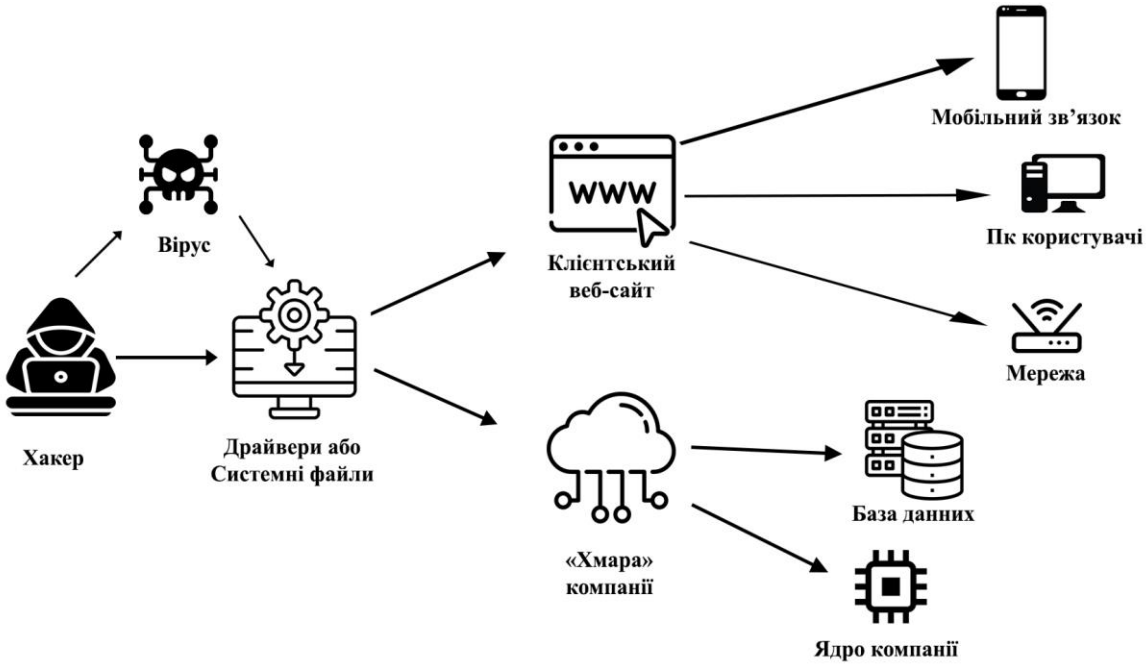


Рисунок 1.4 – Атака через прошивку з основними цілями

Витік, маніпулювання і пошкодження даних виникають у випадках, коли зловмисники отримують доступ до критично важливих даних без контролю доступу та обмеженням щодо копіювання та редагування. В більшості випадках, інформація переносяться на свої хмарні середовища (витік інформації). Найбільш небезпечним варіантом є пошкодження файлів без можливості відновлення. Цілями таких атак стають телефоні оператори, сайти з великою кількістю користувачів, а також сервіси, які надають послуги віртуальних приватних мереж (Virtual Private Network, VPN).

1.2 Огляд та загальна характеристика Інтернету речей

В попередньому підрозділі роботи були окреслені актуальні задачі моніторингу мережевого трафіку для Інтернету речей акцентуючи прямий зв'язок цієї задачі із питаннями захисту інформації та кібербезпеки; також були наведені найбільш типові приклади кібератак, які можуть бути реалізовані щодо

IoT-інфраструктури. Разом із тим, питання захисту інформації та кібербезпеки виходять за рамки даної роботи, в той час коли головний фокус зосереджено на загальній структурі IoT-мереж, кінцевих пристроїв та засобів моніторингу мережевого трафіку.

Згідно концепції IoT, кінцевому користувачу системи надаються послуги з доставки даних, які були отримані сенсорами, за допомогою проводових та безпроводових засобів зв'язку та мережі Інтернет. Прикладами таких вбудовані систем можуть бути засоби побутової техніка, мобільне обладнання, натільні комп'ютери тощо. Джерелом даних може бути інтелектуальний сенсор, який розміщений в зовнішньому середовищі та передає даних до сервера збору інформації проводовими або безпроводовими каналами зв'язку. [18].

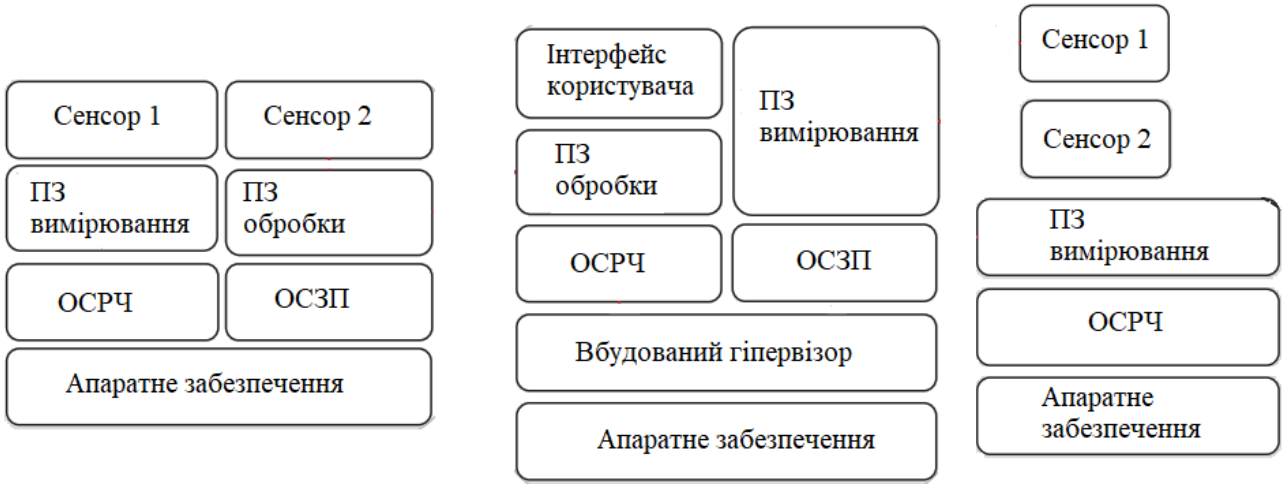
Виділимо такі групи Інтернет-пристроїв згідно їхніх функціональних особливостей:

- шлюзи;
- користувацькі пристрої;
- інтелектуальні сенсор.

Узагальнена структурна схема вищевказаних пристроїв показана на рисунку 1.5.

Шлюз виконує функції керування іншими пристроями, а також акумуляцію та обробку даних смарт-сенсорів. Цей пристрій реалізує методи віртуалізації для підтримки різних пристроїв і додатків Інтернету речей. Ключовою є вимога щодо універсальності та підтримки пристроїв, які можуть керуватись різними операційними системами (ОС), в тому числі операційними системами загального призначення (ОСЗП) та операційними системами реального часу (ОСРЧ).

Кожна ОС взаємодіє з кінцевим користувацьким пристроєм і сенсором через додатки вимірювання фізичних величин та обробки даних, що відіграє важливу роль у наданні послуг мережею Інтернету речей [19].



а) б) в)

Рисунок 1.5 – Структура пристроїв IoT: пристрій користувача (а), шлюз (б) та розумний сенсор (в)

Пристрій користувача має доступ до шлюзу або хмарного сервера та отримує оброблену інформацію, яка потім подається користувачеві в структурованій формі згідно вимог. Користувацький пристрій оснащений власними сенсорами, які дозволяють йому збирати локальні дані середовища. Такий пристрій характеризується вищою обчислювальною потужністю, ніж смарт-сенсори. Прикладами таких пристроїв можуть бути мобільні телефони, телевізори, холодильники та інші подібні міні-пристрої, які є в безпосередньому доступі до користувачів і, таким чином, забезпечують негайне обслуговування і відіграють ключову роль для виконання покладених на них функцій.

Розумний сенсор характеризується вищою за звичайні сенсори вартістю, що є причиною наявності в його складі обчислювальних блоків для попередньої обробки виміряних фізичних величин, тому він є необхідним компонентом сервісу IoT щодо збору інформації та передача даних. В той же час обчислювальна потужність смарт-сенсору нижча, ніж у деякого мобільного пристрою [20], через це вони не підходять для одночасного запуску багатьох додатків. До головних задач розумного сенсору належить збір інформації щодо даних вимірювання, що виконуються самостійно, або у складі сенсорної мережі.

Якість надання послуг такою системою залежить від цілісності та повноти даних, тому ці система мають бути захищеними від витоків інформації та відмов в обслуговуванні. Низька обчислювальна потужність IoT-пристроїв унеможливлює використання програмних методів захисту, які базуються на шифруванні потоку даних, тому актуальним для таких пристроїв є апаратні рішення на основі апаратних засобів, наприклад програмованих логічних інтегральних схем (ПЛІС) [21].

Загальна архітектура середовища IoT середовища показана на рисунку 1.6.

Нижче наведені причини можливих обмежень для IoT які вносять окремі вимоги до програмного засобу моніторингу трафіку:

- низька обчислювальна потужність та обсяг пам’яті;
- потреба захисту технології віртуалізації;
- необхідність підтримки різних ОС.

З обмеженнями щодо продуктивність процесора, обсягу пам’яті та енергоспоживання, існуючі технології кодування та рішення для захисту не можуть бути використані в пристроях IoT. Сенсори є досить легкими пристроями, тому низьке енергоспоживання є ключовим фактом. Для безпечного керування сенсорами в існуючому хмарному середовищі, технології віртуалізації використовуються для захисту додатків або пристроїв, а також для захисту самої системи управління. Однак, гіпервізор, який можна відповідним чином завантажений на IoT-пристрій, вже займає певний обсяг пам’яті та процесорного часу, а отже, не всі функції безпеки є доступними [22]. Таким чином, виникає необхідність у більш ефективних гіпервізорах для ефективного розгортання засобів моніторингу мережі.

Пристрої IoT складаються з малопотужного процесора, який працює під управлінням певні ОС на основі ядра відповідно до покладених задач. Надійність критично важливої інформації може бути підвищити за рахунок захисту

пристрою, а також надійної та безпечної роботи ОС, які працюють на різних пристроях [23].

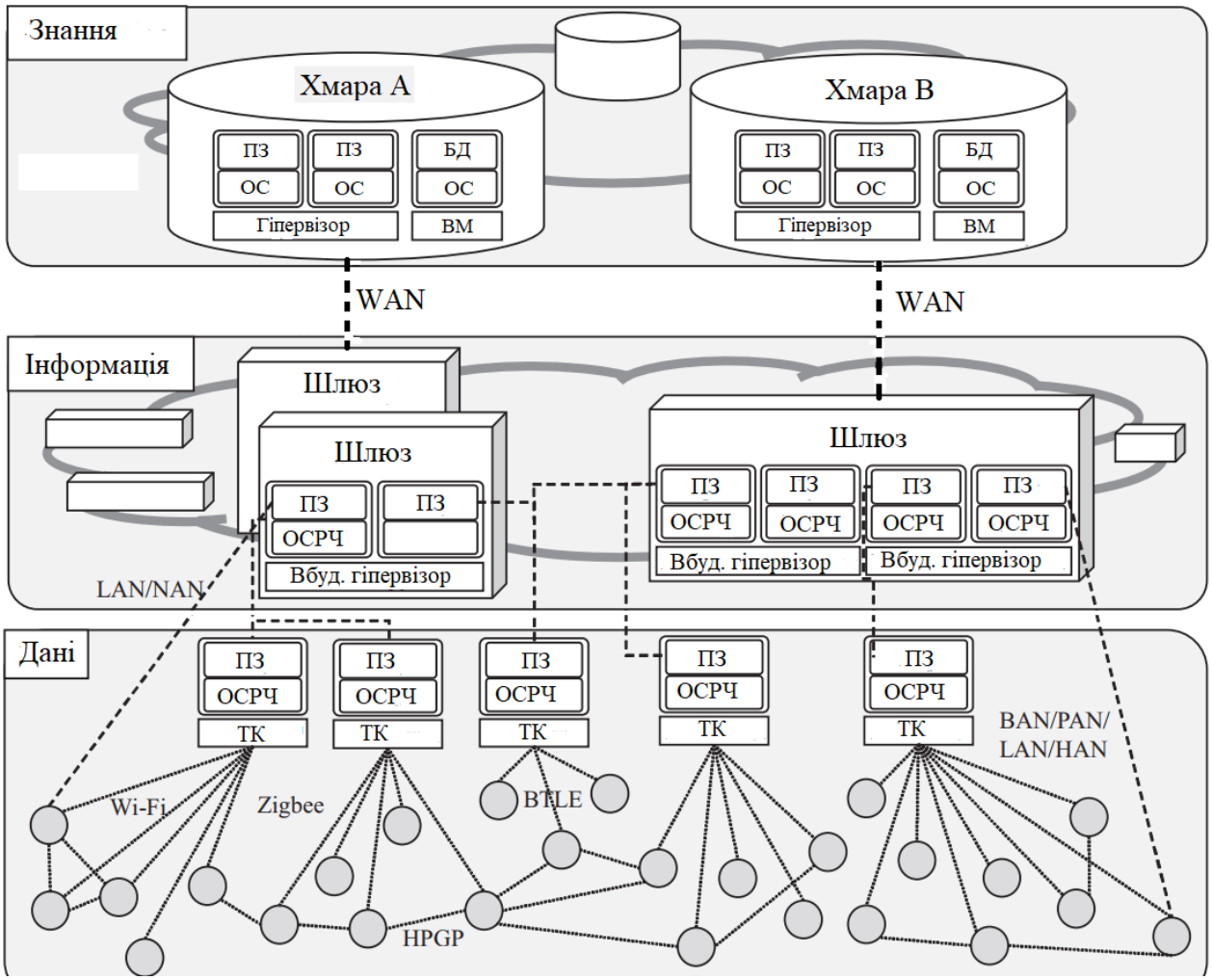


Рисунок 1.6 – Загальна архітектура середовища IoT

1.3 Огляд інструментів моніторингу трафіку

Інструменти моніторингу трафіку – це програмні засоби або сервіси, які дозволяють відстежувати, аналізувати, контролювати і взаємодіяти з трафіком мережі або Інтернету. Вони можуть надавати різну інформацію про мережу, таку як: швидкість передачі, джерела передачі, типи підключень, можливі втрати пакетів даних.

Найвідоміші інструменти моніторингу трафіку:

- Wireshark[24].

- Nmap[25].

- Metasploit[26].

Wireshark – це безкоштовний і відкритий інструмент, аналізатор протоколів мережевого трафіку. Використовується мережевими адміністраторами, фахівцями кібербезпек, хакерами, а також викладачами і студентами. На рисунку 1.7 показано вікно програми Wireshark.

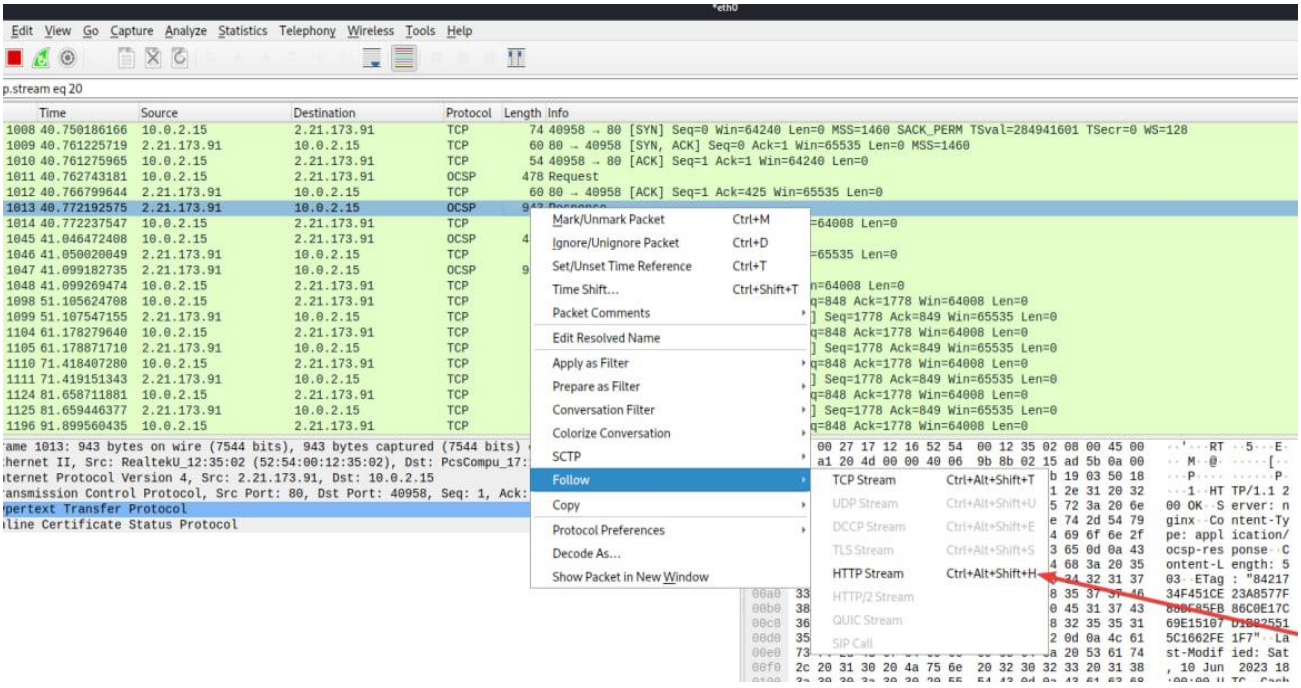


Рисунок 1.7 – Моніторинг трафіку за допомогою Wireshark і фільтрація за протоколом HTTP

До основних можливостей Wireshark відносяться:

- можливість захоплювати пакети даних, що передаються по комп'ютерній мережі. Ці пакети даних можна потім аналізувати, щоб дізнатися більше про мережевий трафік;

- функції аналізу пакетів даних з метою визначення визначити їхнього вміст. Це може бути корисно для налагодження проблем мережі або для вивчення роботи мережевих протоколів;

- фільтрація пакетів даних: з метою відбору потрібних пакетів. Це може бути корисно для звуження великої кількості даних, які захоплює Wireshark;

- функції декодування пакетів даних для представлення вмісту в зрозумілому форматі;

- генерування статистики про мережевий трафік. Ця статистика може бути корисною для налагодження проблем мережі або для отримання уявлення про використання мережі.

- підтримка протоколів широкого спектру мережевих протоколів та стандартів для аналізу трафіку різних типів мереж.

- можливість розширення за допомогою плагінів, які можуть додавати нові функції до Wireshark або підтримувати нові мережеві протоколи.

Утиліта NMAP – це програмний засіб прослуховування і аудиту безпеки комп'ютерних мереж. Основні можливості програми:

- сканування портів TCP/UDP з метою виявлення запущених процесів. Після сканування виводиться інформацію про тип, версію і активність процесу;

- сканування IP-адрес в діапазонах для визначити активних хостів; визначення типу ОС хоста;

- використання різних способів сканування і фільтрування даних.

На рисунку 1.8 показано приклад роботи програми NMAP щодо перевірки сайту та віртуальної машини хоста.

Програмний засіб Metasploit використовується для тестування на проникнення і розробку експлойтів (програма або послідовність команд). Основні можливості Metasploit:

- підтримка великої кількості експлойтів для різних операційних систем, програмного забезпечення, тощо;

- сканування мережі та системи на наявність вразливостей, використовуючи різні методи, такі як Nmap, Nessus[27] та OpenVAS[28];

- наявність вбудованого фреймворку для розробки власних експлойтів.

- автоматизації завдань тестування на проникнення, таких як сканування, експлуатація та пост-експлуатація.

- можливість створення тунелів Secured Shell (SSH) та VPN, які можна використовувати для обходу брандмауерів та мережевих обмежень.

- доступність для різних платформ, включаючи Windows, Linux, macOS та Kali Linux[29].

На рисунку 1.9 показано, як створюється вірус-прослуховувач. На рисунку 1.10 показано кінцевий результат, коли вірус підключився до віртуальної машини з IP адресою: 127.0.0.1:50242.

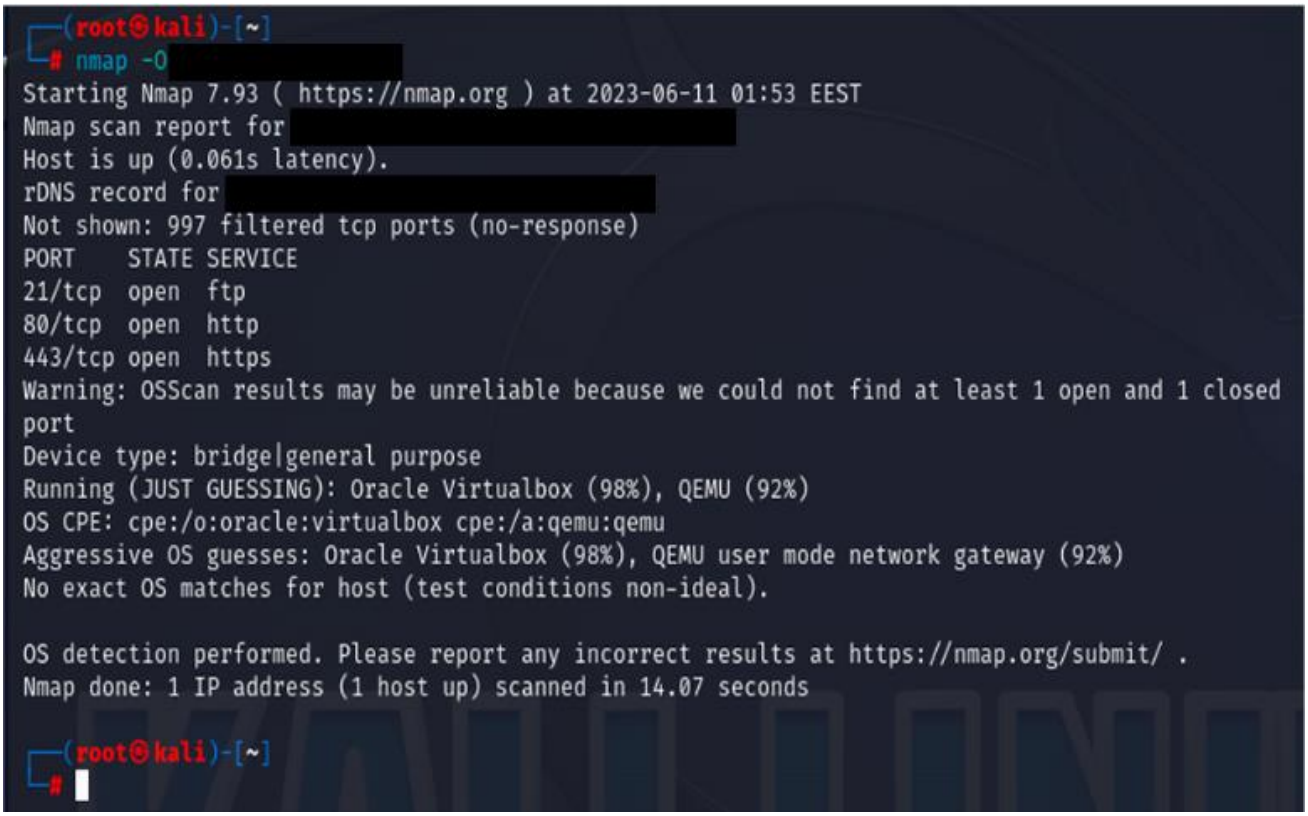


Рисунок 1.8 – Результат сканування NMAP

В таблиці 1.1 наводяться основні переваги та недоліки існуючих програмних засобів моніторингу мережевого трафіку для інтернету речей.

Рисунок 1.9 – Створення віруса-прослуховувача

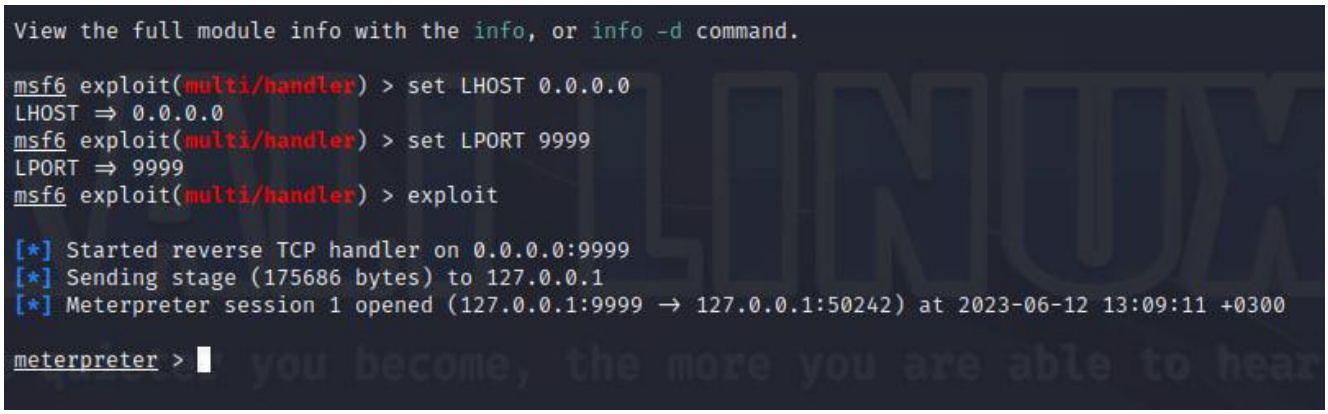


Рисунок 1.10 – Підключення до віртуальної машини

1.4 Постановка задач кваліфікаційної роботи

В результаті аналізу предметної області, існуючих програмних засобів та їхніх недоліків та переваг були сформульовані такі задачі до вирішення:

1. Провести аналіз вимог до програмного засобу моніторингу мережевого трафіку пристроїв Інтернету речей.

2. Дата загальну характеристику та розробити структурно схеми кінцевого пристрою IoT на базі розумного сенсору.

3. Розробити схему мережі IoT, де відобразити підключення IoT-пристроїв до локальної мережі та до мережі Інтернет.

4. Розробити модель даних для моніторингу трафіку.

5. Виконати програмну реалізацію компонентів програмного засобу моніторингу трафіку для Інтернету речей.

6. Виконати розгортання та тестування програмного засобу.

Таблиця 1.1 – Переваги і недоліки існуючих програмних засобів моніторингу мережевого трафіку

Назва	Опис	Переваги	Недоліки
1	*2*	*3*	*4*
Wireshark	Інструмент для аналізу мережевого трафіку, який дозволяє перехоплювати, відстежувати і аналізувати пакети даних, що проходять через комп'ютерну мережу.	Можливість аналізу всіх шарів мережевого стеку. Широкий спектр підтримувальних протоколів.	Складність використання для новачків. Потреба значних обчислювальних ресурсів і обсягу пам’яті. Необхідність прав адміністратора (Windows)
Wifite [31]	Інструмент для тестування на проникнення бездротових мереж Wi-Fi.	Автоматизація процесу: Wifite надає зручний інтерфейс для автоматизації сканування мереж.	Низька ефективність проти захищених wi-fi мереж, типу WPA2. Обмежена підтримка адаптерів.
NMAP	Відкрите програмне забезпечення для сканування мережі, призначене для виявлення комп'ютерів та визначення активних хостів на мережі.	Мультиплатформенність: NMAP підтримується на всіх основних ОС Гнучкість налаштування.	Можливість спричинення перевантаження системи. Є ризики втрати ефективності або повної втрати результатів сканування.

Продовження таблиці 1.1– Переваги і недоліки існуючих програмних засобів моніторингу мережевого трафіку

John the Ripper [30]	Програма для відновлення паролів, яка використовує методи атаки, такі як brute-force (груба сила), словникова атака та комбіновані методи для розшифрування.	Гнучкість та розширюваність: John дозволяє налаштовувати різні типи атак, комбіновані методи та параметри підбору.	Потреба у великих потужностях: Перебір паролів, потребує або потужної відео карти або процесора. Обмежена ефективність для специфічних паролів.
Wifite [31]	Інструмент для тестування на проникнення бездротових мереж Wi-Fi, який дозволяє автоматизувати процеси сканування, атак і взлому захищених мереж Wi-Fi.	Автоматизація процесу: Wifite надає зручний інтерфейс для автоматизації сканування мереж, вибору атак і зламу мереж Wi-Fi.	Низька ефективність проти захищених wi-fi мереж, типу WPA2, з використанням довгих паролів Обмежена підтримка адаптерів.
Kismet [32]	Інструмент для моніторингу та виявлення бездротових мереж Wi-Fi, який дозволяє аналізувати мережевий трафік, виявляти пристрої, розпізнавати мережеві атаки і здійснювати геолокацію пристроїв.	Програма надає розширені можливості аналізу мережевого трафіку, включаючи виявлення пристроїв, аналіз протоколів, розпізнавання мережевих атак.	Високі вимоги до ресурсів Складність налаштування для початківців Можливі проблеми з сумісністю адаптерів і не точна геолокація

Продовження таблиці 1.1– Переваги і недоліки існуючих програмних засобів моніторингу мережевого трафіку

Nikto [33]	Відкрите ПЗ для сканування веб-серверів на наявність потенційних проблем. Використовується для ідентифікації різних видів вразливостей.	Підтримка різних типів властивостей: Крос-сайт скриптінг (XSS), sql-ін'єкцій, вразливості веб-серверів. Має інтерфейс терміналу	Можлива велика кількість фальшивих позитивів і негативів. Залежність від вільних розробників.
Metasploit	потужний фреймворк для тестування на проникнення та експлойтації вразливостей в інформаційних системах.	Широкий функціонал аспектів тестування на проникнення, сканування, пошук вразливостей. Активна спільнота розробників.	Має погану репутацію, хоча і використовується в кібербезпеці Потребує глибоких технічних знань з мережевої безпеки. Потреба в постійному оновленні
Maltego [34]	Інструмент для збору та аналізу відкритої інформації про цільові об'єкти. Використовується для проведення розвідувальних досліджень.	Maltego дозволяє легко користуватись, для набуття навиків Надає широкий функціонал для різноманітних засобів збору інформації	Обмежена безкоштовна версія. Залежність від доступності даних і їхньої кількості.

Кінець таблиці 1.1– Переваги і недоліки існуючих програмних засобів моніторингу мережевого трафіку

Tcpdump [35]	Утиліта командного рядка для аналізу та відловлювання пакетів в мережах. Дозволяє перехоплювати та аналізувати мережевий трафік у реальному часі або зчитувати дані з раніше записаних файлів за допомогою фільтрів для відбору пакетів за різними критеріями.	Гнучкість фільтрації: надає можливість відбирати пакети за критеріями. Легкість користування через термінал. Підтримка різних форматів запису.	Відсутність графічного інтерфесу. Відсутність автоматичного аналізу пакетів, що вимагає ручну обробку даних. Обмежені можливості аналізу.
Hping3 [36]	Інструмент командного рядка, який використовується для генерації та надсилання пакетів TCP/IP, UDP та ICMP. Може використовуватися для тестування мереж, діагностики проблем з мережею та навантаження серверів.	Може генерувати різні пакети за різними параметрами і за різною швидкістю, розміром, інтервалом. Можливе тестування безпеки за різними категоріями.	Відсутність графічного інтерфейсу.Складність використання для новачків. Неінтуїтивний синтаксис команд.

1.5 Висновки до першого розділу

В першому розділі бакалаврської кваліфікаційної роботи було проведено аналіз предметної області згідно теми дослідження, зокрема було проаналізовано види та потреба існуючих методів моніторингу мережевого трафіку та активності кінцевих IoT-пристроїв з позиції системного адміністрування та протидії кіберзагрозам. Проведено аналіз найбільш типових кібератак, якими можуть бути уражені системи Інтернету речей. Виконано порівняльний аналіз найбільш відомих програмних засобів моніторингу мереж, виділено переваги та недоліки існуючих програмних засобів. За результатами розділу були сформульовані практичні задачі для вирішення в кваліфікаційній роботі.

2 ПРОЄКТУВАННЯ ПРОГРАМНОГО ЗАСОБУ МОНІТОРИНГУ МЕРЕЖЕВОГО ТРАФІКУ ПРИСТРОЇВ ІНТЕРНЕТУ РЕЧЕЙ

2.1 Аналіз вимог та розробка моделі варіантів використання програмного засобу

На основі аналізу предметної області, який був виконаний в першому розділі кваліфікаційної роботи, виділимо основні функціональні вимоги до програмного засобу з позиції адміністратора системи, який є головним користувачем програми.

Виділено такі головні вимоги до функціоналу програмного засобу:

- авторизація користувача;
- перегляд списку активних пристроїв;
- перегляд даних в реальному часі;
- перегляд даних статистики.

В результаті аналізу функціональних вимог до програмного засобу, була розроблена модель варіантів використання, яка зображена на рисунку 2.1.

Згідно моделі, головним користувачем програмного засобу є системний адміністратор (DevOps інженер), до задач якого входить підтримка, відновлення та моніторинг стану пристроїв Інтернет речей. В якості прикладу такої системи розглянемо систему збору передачі та збереження даних вимірювань фізичної величини розумними сенсорами на базі Інтернету речей.

Відповідно до розробленої моделі, дамо опис основним варіантам використання.

Для входу в систему, користувачу необхідно пройти процедуру авторизації, базовим сценарієм якої є, введення облікових даних (логіну і надійного пароля). З метою підвищення захищеності та уникнення несанкціонованого доступу, базовий сценарій авторизації користувача, може бути розширений за допомогою використання двохфакторної автентифікації.

Пройшовши процедуру авторизації, ідентифікований користувач з правами адміністратора отримує доступ до базових функцій засобу моніторингу. Базовим варіантом використання є перегляд даних, головний сценарій якого, починається зі списку(розумних сенсорів – кінцевих IoT-пристроїв системи). Варіант використання «перегляд даних» є базовим (батьківським) для таких конкретних реалізацій, як перегляд статистики і перегляд даних трафіку в реальному часі. Таке архітектурне рішення пов’язане з тим, що більшість вимог для цих двох варіантів використання є спільними та можуть бути сформульовані узагальнено.

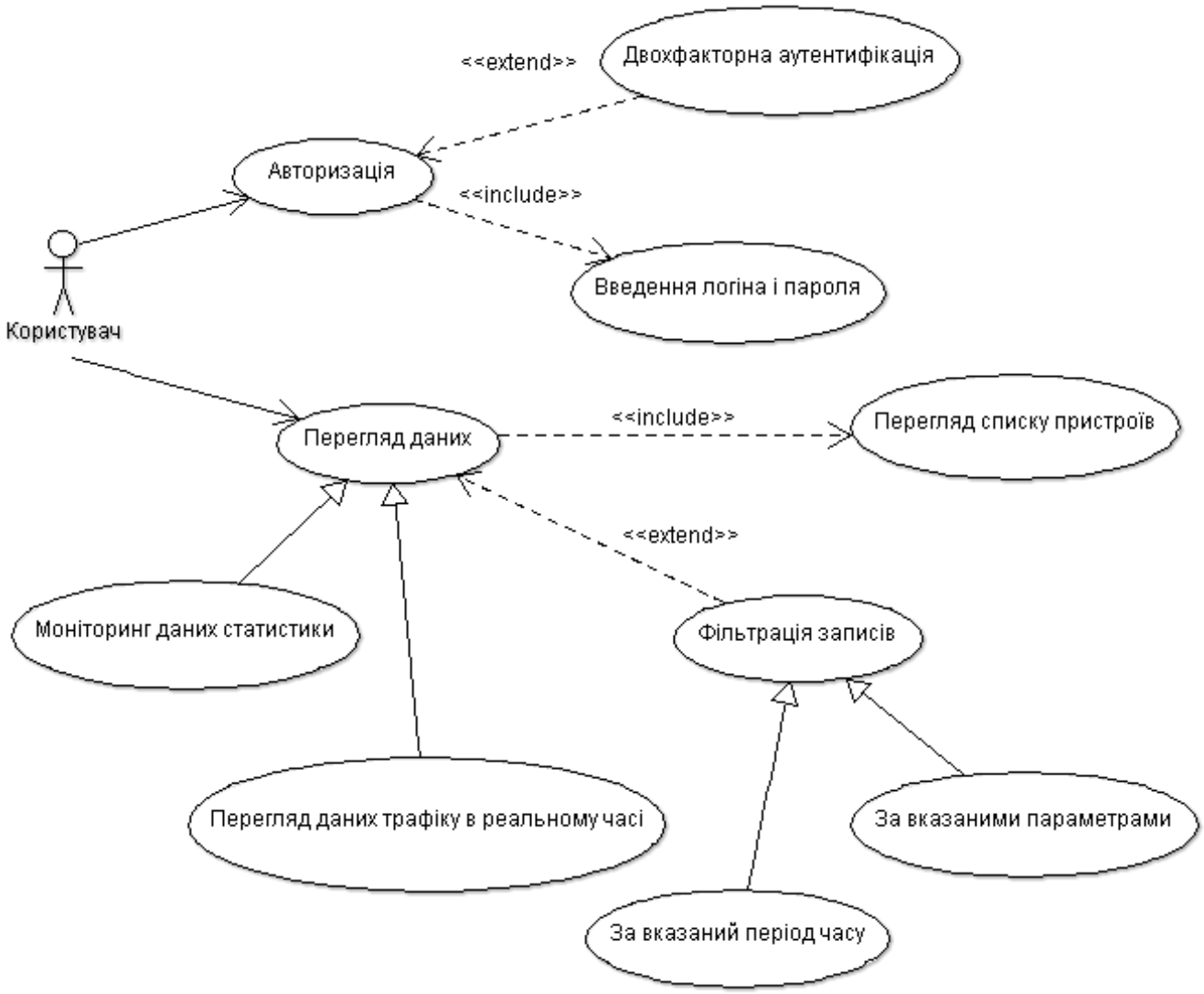


Рисунок 2.1 – Модель варіантів використання програмного засобу

Базовим сценарієм перегляду даних є виведення списку пристроїв з відображенням унікального ідентифікатора, стану, супровідної інформації у стислому вигляді. Для обраного пристрою (розумного сенсору) користувач може відкрити сторінку із графічними даним візуалізації параметрів трафіку. За необхідністю варіант використання може бути розширений за допомогою механізму фільтрації, за період часу і за вказаними користувачем параметрами.

Додатково виділимо базові нефункціональні вимоги, які безпосередньо не впливають на функціонал програмного засобу, але реалізація яких потенційно може покращити людино-машинну взаємодію:

- інтуїтивно зрозумілий інтерфейс,
- підтримка локалізації,
- адаптивний дизайн.

Вимоги до апаратної платформи:

- низьке споживання електроенергії,
- портативність та можливість тривалої автономної робота з батарейним живленням;
- підтримка безпроводових каналів зв’язку (803.11 Wi-Fi [38], 802.15.1 Bluetooth [39]).

2.2 Проєктування та моделювання мережі пристроїв Інтернету речей

Даний розділ присвячений проєктування і моделювання мережі – ключовому процесу розгортання застосувань на базі Інтернету речей. Згідно концепції IoT [40], система застосувань, що побудовані на її базі складається із деякого скінченного числа малопотужних розумних пристроїв з підключенням до мережі Інтернет. Структура та властивості, а також класифікація цих пристроїв була детально розглянута в розділі 1.2. З позиції системного адміністрування важливим є забезпечення кінцевих IoT-пристроїв безперебійним живленням та стабільним доступом до мережі Інтернет. Таким чином, перши

етапом розгортання IoT інфраструктури є створення локальної мережі, яка об’єднує всі кінцеві пристрої та надає їм підключення до мережі Інтернет через загальний шлюз. В такій мережі шлюз виконує функції трансляції адрес (Network Address Translate, NAT), мережевого екрану (Firewall), сервера доменних імен (Domain Name Server, DNS), динамічного конфігурування хвостів (Dynamic Host Configuration Protocol, DHCP) [41].

З метою сприяння кращому розумінню структури та концепції IoT з позиції контролю та адміністрування було виконано моделювання мережі за допомогою інструментального засобу візуального моделювання Cisco Packet Tracer. Схема мережі показана на рисунку 2.2.

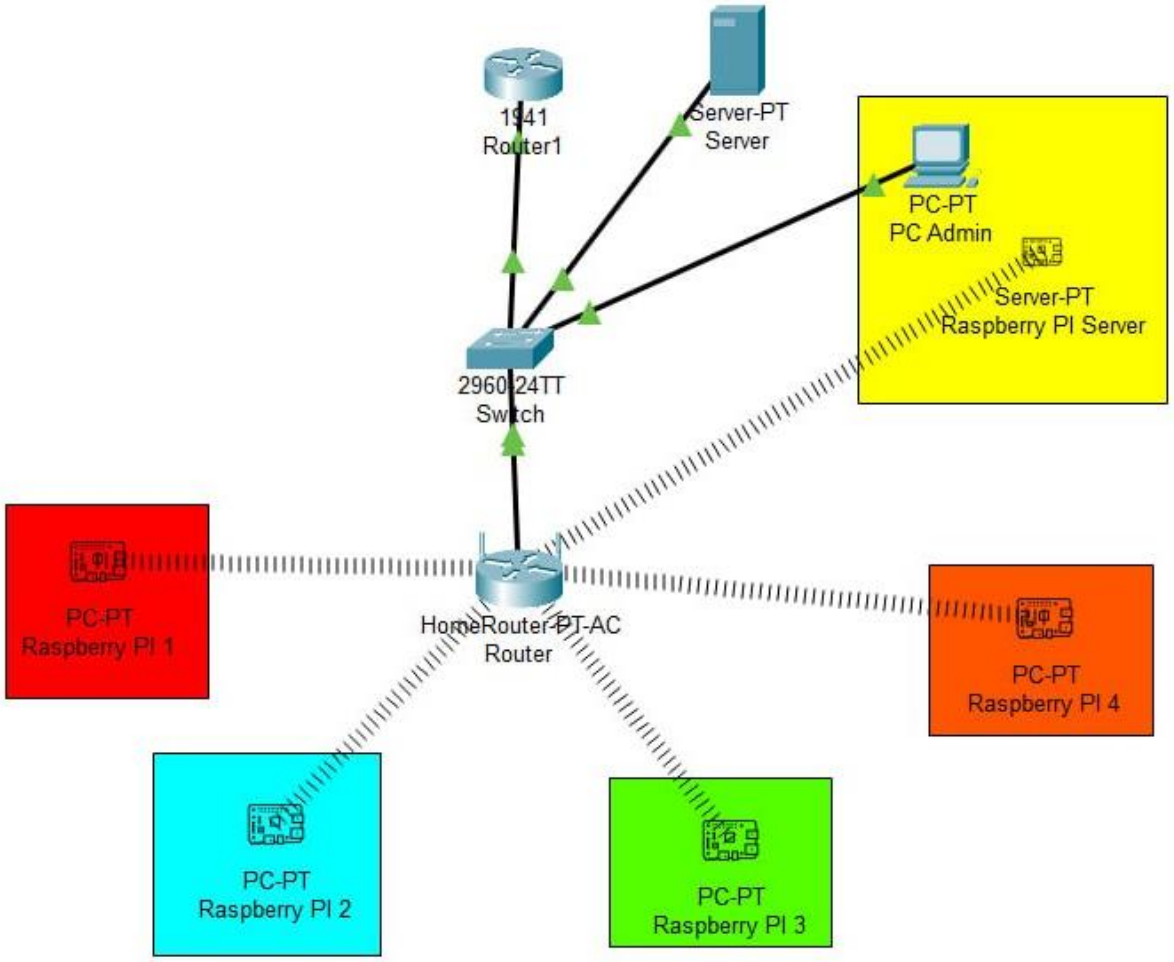


Рисунок 2.2 – IoT система на базі локальної мережі

В моделюванні були використані:

- п’ять комп’ютерів
- два сервери
- один домашній маршрутизатор
- один маршрутизатор 1941
- один комутатор 2960.

Іконки чотирьох комп’ютерів, були змінені на іконку Raspberry Pi[43], які виділені прямокутником, кожний з яких має різний колір, для визначення пристрою, за типом даних, способом моніторингу, місцем розташування або іншими відповідними критеріями, які буде ставити за пріоритет адміністратор. В якості місця зберіганням даних, використовується сервер з назвою «Server-Traffic». Цей сервер надає можливість адміністратору фільтрувати трафік за критеріями і переглядати активність мережі. Маршрутизатор з’єднує між собою сервер і пристрої, мережеві налаштування за протоколом DHCP, для створення підмережі в якій будуть виконуватись основні задачі. Комутатор 2960 з’єднує безпровідне підключенням з провідним, що дозволяє ідентифікувати підмережу і пристрої, які зможуть її ідентифікувати. Маршрутизатор 1941 надає доступ до Інтернету та засоби захисту. Сервер з назвою «Server» надає службу DHCP для автоматичного конфігурування мережевих налаштувань.

Комп’ютер PC-Admin використовується для віддаленого доступу до засобів моніторингу для адміністратора, і з якого також вносяться корегування або доповнення до налаштувань мережі.

Цей тип підключення, створюється для локальної мережі з метою збору даних з основних IoT-пристроїв, передачі їх на Server-Traffic для моніторингу.

У випадку, коли немає можливості реалізувати безпровідне підключення, є можливість підключити IoT-пристрій через кабель. Використавши такий варіант підключення можна забезпечити надійний і стабільний зв’язок, який не обмежується перешкодами. Але це робить підключення пристроїв обмеженим, на кількість вільних портів комутатора.

На рисунку 2.3 зображено підключення до зовнішньої мережі через комутатор в кросплатформенному інструменту візуального моделювання Cisco Packet Tracer.

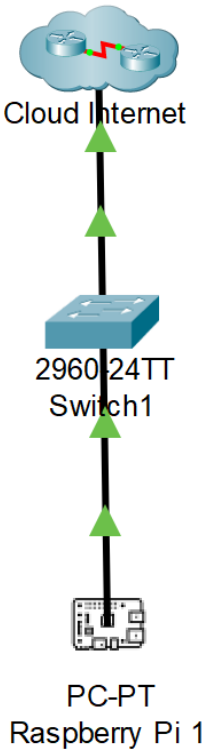


Рисунок 2.3 – Підключення через кабель

Cloud Internet надає IP-адресу для пристроїв в мережі. Комутатор отримує дані з пристроїв і Raspberry Pi збирає дані і надає їх через кабель або використовується, як точка, яка виконує операцію моніторингу трафіку в мережі.

Безпроводове підключення вимагає використання маршрутизатора, який також має функцію безпроводової точки доступу для підключення до Інтернету.

Такий варіант підключення, надає можливість підключати велику кількість пристроїв, з високою швидкістю. Завдяки таким можливостям, таке підключення є найкращим і обмежується тільки специфічним обладнанням і швидкістю передачі даних пристроїв, яку вони можуть надати.

На рисунку 2.4 зображено підключення до зовнішньої мережі через маршрутизатор з безпроводовою точкою доступу.

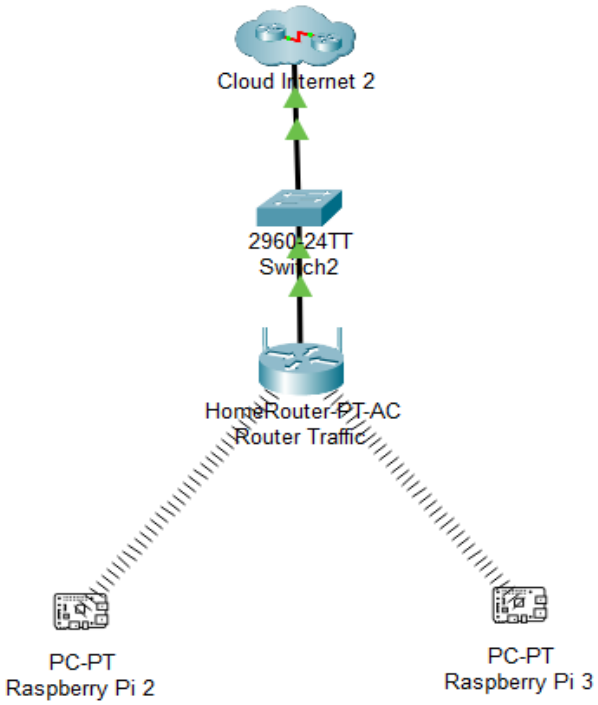


Рисунок 2.4 – Підключення через маршрутизатор з безпроводовою точкою доступу

2.3 Модель та опис даних для моніторингу

Для функціонування мережі, визначено критерії моніторингу, які дозволяють оптимізувати роботу адміністратора, виявляти проблеми і забезпечувати стабільну роботу моніторингу. В таблиці 2.1 розписані основні параметри загального трафіку.

Таблиця 2.1 містить інформацію про завантаження даних, вивантаження даних, швидкість завантаження і вивантаження, час виміру, назву даних, тип даних і приклад запису даних.

Стовпці описують поля та їхню класифікацію:

- номер: нумерація даних.

- параметри: описує тип даних, що міститься в записі.

- назва поля: містить назву параметру в певному описі, який зрозумілий для адміністратора.

- тип даних: описує тип даних, в якому вони вимірюються.

- приклад даних запису: містить приклад значення даних у полі.

Таблиця 2.1 надає основну інформацію про трафік даних, які можуть бути використані для аналізу і вдосконалення продуктивності мережі.

Таблиця 2.1 – Загальні параметри моніторингу трафіку

№	Параметри	Назва поля	Тип даних	Приклад запису
1	Завантажено МБ	TotalDownload	Double	10 Мбіт/с
2	Вивантажено МБ	TotalUpload	Double	5 Мбіт/с
3	Швидкість завантаження Мбіт/с	TotalDownloadSpeed	Double	10 Мбіт/с
4	Вивантаження Мбіт/с	TotalUploadSpeed	Double	20 Мбіт/с
5	Час виміру	TrafficLogTime	DateFine	2024-05-05 12:50:10:004

Таблиця 2.2 містить інформацію про параметри трафіку мережевих інтерфейсів.

Таблиця 2.2 – Параметри трафіку за інтерфейсами

Номер	Назва параметру	Назва поля	Тип даних	Приклад запису
1	Назва інтерфейсу	Interface	String	eth0
2	Mac-адреса	Mac	String	ab:cd:ef
3	Ip-адреса	IpAddress	String	192.168.0.10
4	Інтернет шлюз	Gateway	String	192.168.0.1
5	DNS Server	DNS Server	String	8.8.8.8
6	Опис (Примітка)	Notes	Varchar	Process 99%

Таблиця 2.2 містить інформацію про назву інтерфейсу, Mac-адресу, IP-адресу, Інтернет шлюз, DNS сервер і опис, за яким можна визначити завантаженість інтерфейсу.

Стовпці описують поля та їхню класифікацію:

- номер: Нумерація даних.
- параметри: Описує тип даних, що міститься в записі.
- назва поля: Містить назву параметру в певному описі, який зрозумілий для адміністратора.
- тип даних: Описує тип даних, в якому вони вимірюються.
- приклад даних запису: Містить приклад значення даних у полі.

Таблиця 2.2 надає основну інформацію про інтерфейс, за яким можна його класифікувати і виявляти несправності, які можуть виникнути.

- визначення неправильної IP-адреси.
- підключення Інтернет шлюза і DNS сервера
- завантаженість інтерфейсу.

Таблиця 2.3 містить інформацію про дані програмного інтерфейсу і їхні параметри за якими вони визначаються.

Таблиця 2.3 – Дані програмного інтерфейсу

Номер	Назва параметру	Назва поля	Тип даних	Приклад запису
1	Id процесу	PId	int	1234
2	Назва процесу	Process Home	String	svchost
3	Час створення	Cost time	Date time	2024-05-05
4	Опис (Примітка)	Gateway	varchar	192.168.0.1

Таблиця 2.3 містить інформацію про Id процесу, назву процесу, час створення і його опис.

Стовпці описують поля та їхню класифікацію:

- номер: Нумерація даних.

- параметри: Описує тип даних, що міститься в записі.

- назва поля: Містить назву параметру в певному описі, який зрозумілий для адміністратора.

- тип даних: Описує тип даних, в якому вони вимірюються.

- приклад даних запису: Містить приклад значення даних у полі.

За допомогою таблиці 2.3 можна визначити кожний процес, який створюється і класифікацію процесу. Ці параметри можуть оновлюватись і добавлятись в залежності від даних, які приймаються пристроями. Але основні параметри, такі як: Id процесу, Назва процесу, Час створення є початковою необхідністю, за якою визначають процес.

Таблиця 2.4 містить дані про Інтерфейс-пристрої і їхні основні параметри.

Таблиця 2.4 – Інтернет-пристрої

Номер	Назва параметру	Назва поля	Тип даних	Приклад запису
1	Id пристрою	PId	int	10
2	Назва	Dev Home	String	Сенсор температури
3	Опис (Примітка)	Notes	varchar	Працює з процесами

Таблиця 2.4 описує параметри, такі як: Id пристрою, назва пристрою і опис.

Стовпці описують поля та їхню класифікацію:

За допомогою таблиці 2.4 можна визначати кожний пристрій за Id, назвою і описом.

Таблиці 2.1-2.4 описують дані, які відповідають їхнім параметрам, для створення повноцінної бази даних. Рисунок 2.4 зображує створену базу даних, в якій інформація з кожної таблиці була використана для роботи один з одним.

База даних починається з IoT-пристроїв. В ній описані ID-пристрою, назву пристрою і опис пристрою.

На початку роботи бази даних, загального запису немає, бо вона ще пуста, але їхня кількість може бути необмежена, бо IoT пристрій може записувати багато разів інформацію. Загальний запис створює унікальний ID-запису, для відстеження кількості кілобайт, які були завантажені і вивантажені зі швидкістю передачі даних, яка була під час певного запису.

Програми, які можуть запускатись на пристрої, дуже багато. Для кожної програми, надається унікальний ID-програми, ID-пристрою, на якому програма на яку вона запущена, назву програми і час запуску програми на пристрої.

Програма, складається з необмеженої кількості процесів і у випадку аномальної роботи або хакерської атаки, є можливість зрозуміти, який процес використовується для атаки, а який для роботу пристрою. Кожному процесу надається унікальний ID з ім'ям і часом створення, за яким його можна ідентифікувати і відслідкувати.

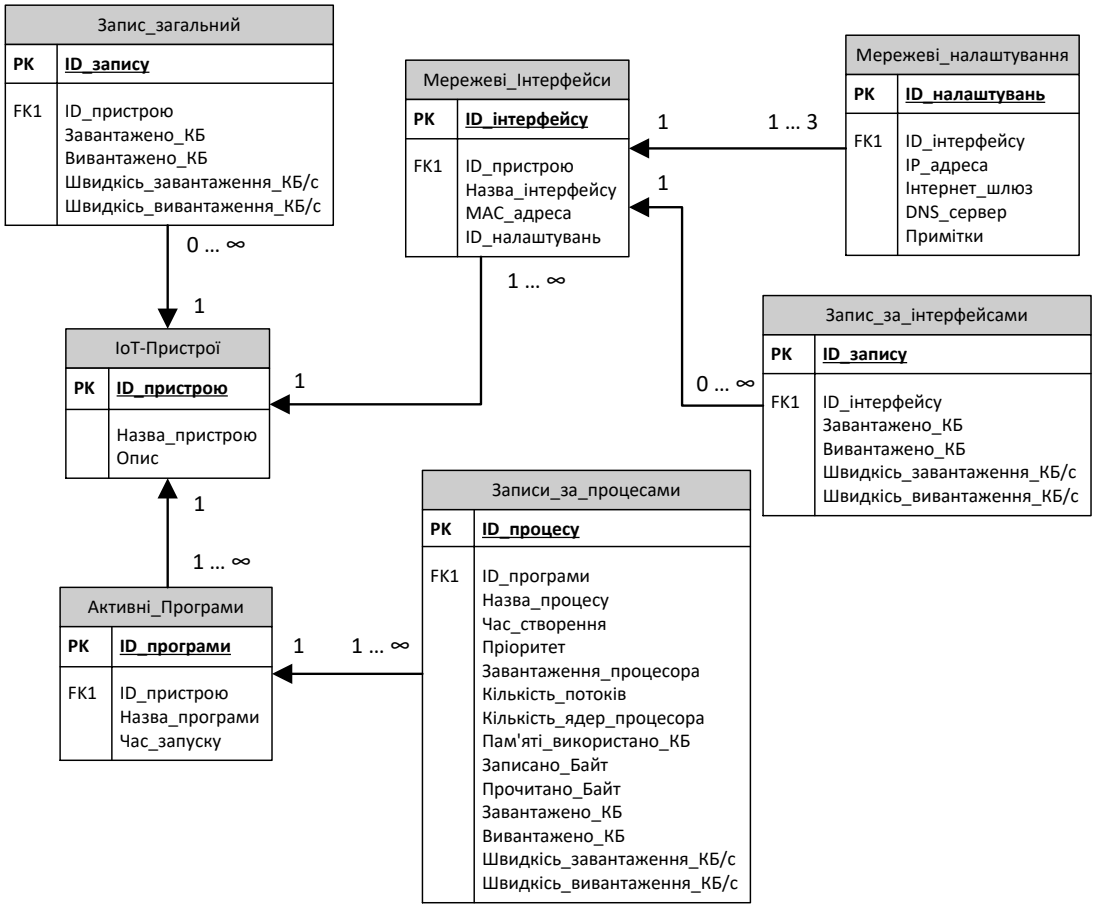


Рисунок 2.5 – Схема бази даних

Після класифікації процесу, можна визначити пріоритетність в роботі, завантаженість процесора, використання кількості потоків і кількості ядер процесора з використанням пам’яті. За процесом можна визначити, скільки було байт записано і прочитано, завантажено і вивантажено кілобайт з їхньою швидкістю завантаження і вивантаження в секунду.

Іот-пристрій має велику кількість мережевих інтерфейсів, які можуть бути підключатись фізично через Ethernet-кабель, бездротові, які підключаються через мережу і віртуальні, які створюються програмою.

2.4 Розробка бази даних

Нижче наведено програмний код на мові Python для роботи з БД, схема якої була розроблена в попередньому розділі:

Бібліотека sqlachemy – це інструментарій для роботи бази з базами даних в мові програмування Python. Це дозволяє спрощувати роботу з SQL-запитів, розширює функціонал Python з взаємодією БД і обробляти реляційні записи.

Опишемо детально код програми.

```
from sqlalchemy import Column, Integer, Float, DateTime, String, ForeignKey, Table
from sqlalchemy.orm import relationship, backref
from sqlalchemy.ext.declarative import declarative_base
Base = declarative_base()
```

В цій частині коду, типи даних імпортуються з інструментарію sqlalchemy для створення моделі бази даних і визначення таблиці. Клас Base, надає функціональність для всіх таблиць.

```
# IoT-пристрій
class Device(Base):
```

```
    __tablename__ = "device"
    device_id = Column(Integer, primary_key=True) # первинний ключ
    dev_name = Column(String) # назва притсрою
    dev_descr = Column(String) # опис пристрою
    general_logs = relationship("General_Log", backref=backref("device"))
    active_apps = relationship("Active_App", backref=backref("device"))
    network_interfaces = relationship("Network_Interface", backref=backref("device"))
```

Клас Іот-пристрій, надає унікальний ідентифікатор кожному пристрою, назву, опис і створює зв'язок між таблицями general_logs, active_apps, network_interfaces.

```
# запис загальний
class General_Log(Base):
    __tablename__ = "general_log"
    general_log_id = Column(Integer, primary_key=True)
    dev_id = Column(Integer, ForeignKey("device.device_id"))
    download_KB = Column(Float) # завантажено_кб
    upload_KB = Column(Float) # вивантажено_кб
    download_speed_KB_s = Column(String) # швидкість завантаження кб/с
    upload_speed_KB_s = Column(String) # швидкість вивантаження кб/с
```

Клас General_Log представляє загальні записи з унікальним ідентифікатором запису, ID пристрою і інформацією про завантаження і вивантаження даних.

```
# активна програма
```

```
class Active_App(Base):
    __tablename__ = "active_app"
    active_app_id = Column(Integer, primary_key=True)
    dev_id = Column(Integer,
ForeignKey("device.device_id"))
    app_name = Column(String) # назва пристрою
    start_time = Column(DateTime) # час запуску
    process_logs = relationship("Process_Log",
backref=backref("active_app"))
```

Клас Active_App представляє активну програму, яка має свій ID, ID-пристрою, назву самої програми, час запуску і має зв'язок з процесами.

```
# мережевий інтерфейс
class Network_Interface(Base):
    __tablename__ = "network_interface"
    network_interface_id = Column(Integer,
primary_key=True)
    dev_id = Column(Integer,
ForeignKey("device.device_id"))
    netw_interf_name = Column(String) # назва інтерфейсу
    mac_address = Column(String) # mac-адреса
    start_time = Column(DateTime) # час запуску
    network_if_logs = relationship("Network_Interface_Log",
backref=backref("network_interface"))
    network_settings = relationship("Network_Settings",
backref=backref("network_interface"))
```

лас Network_interface, описує дані мережевих інтерфейсів, для зв'язку між Network_Settings і Network_interface_log. В класі описаний ID інтерфейсу, ID пристрою до якого він належить, назва інтерфейсу, Mac-адресу інтерфейсу, час запуску інтерфейсу.

```
# записи за процесами
```

```
class Process_Log(Base):
    __tablename__ = "process_log"
    proc_log_id = Column(Integer, primary_key=True)
    active_app_id = Column(Integer,
ForeignKey("active_app.active_app_id"))
```

Клас Process_Log містить детальну інформацію про процеси, які можуть відтворюватись і класифікувати за різними ознаками. Кожний процес, має унікальні ID, ім’я процеса, час створення, пріоритетність, використання ядер, потоків і оперативної пам’яті тощо.

```
# запис загальний
class Network_Interface_Log(Base):
    __tablename__ = "network_interface_log"
    network_if_log_id = Column(Integer, primary_key=True)
    network_interface_id = Column(Integer,
ForeignKey("network_Interface.network_interface_id"))
```

Клас Network_Interface_Log записують основні дані про завантаження і вивантаження даних, швидкість завантаження і вивантаження даних, які передають через мережевий інтерфейс.

```
class Network_Settings(Base):
    __tablename__ = "network_settings"
    netw_set_id = Column(Integer, primary_key=True)
    ip_address = Column(String)
    gateway = Column(String)
    dns = Column(String)
    notes = Column(String)
    network_interface_id = Column(Integer,
ForeignKey("network_Interface.network_interface_id"))
```

Цей клас представляє мережеві налаштування інтерфейсу, що дозволяє визначити Ip-адресу, шлюз, DNS, нотатки тощо.

2.5 Висновок до другого розділу

В другому розділі бакалаврської кваліфікаційної роботи було визначено, головні вимоги до функціоналу програмного засобу. Відповідно до основних вимог була створена модель варіантів використання програмного засобу кінцевим користувачем – системним адміністратором. Додатково було визначено нефункціональні вимогами, реалізація яких в подальшому дозволить покращити взаємодію між користувачем і програмним засобом. В процесі проєктування були створенні схеми мереж в кросплатформенній програмі візуального моделювання мережі Cisco Packet Tracer та схема бази даних параметрів моніторингу трафіку, які були використані для створення програмного модуля для роботи з базою даних на мові Python.

3 РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ ПРОГРАМНОГО ЗАСОБУ МОНІТОРИНГУ МЕРЕЖЕВОГО ТРАФІКУ ПРИСТРОЇВ ІНТЕРНЕТУ РЕЧЕЙ

3.1 Опис реалізації модулів апаратного та програмного забезпечення програмно-технічного засобу

Для реалізації програмного засобу моніторингу мережевого трафіку пристроїв Інтернету речей потрібно визначити перелік інструментів, які будуть використані для основи програмного засобу і дозволять в майбутніх оновленнях не навантажувати засіб окремими доповненнями, з якими можливий ризик не сумісності.

Платформою для реалізації програмного засобу моніторингу мережевого трафіку пристроїв Інтернету речей, використовується мініатюрний одноплатний комп'ютер Orange Pi 3B.

Перелік інструментів, який використаний для реалізації програмного засобу моніторингу мережевого трафіку пристроїв Інтернету речей:

- мова програмування Python;
- фреймворк для розробки вебсистем Django;
- бібліотека для створення віртуального ізольованого середовища Virtualenv;
- бібліотека Flexmonster Pivot Table & Charts для зведення таблиць і графіків мови програмування JavaScipt;
- система керування базами даних SQLite;
- мова веб-розмітки HTML і мова стилю сторінок CSS;
- мова програмування JavaScript.

Orange Pi 3B – це одноплатний мікрокомп'ютер, який надає користувачеві запускати більшість проектів віддалено, за мінімальні витрати по ціні і енергоспоживанню. Такі мікрокомп'ютери використовуються студентами для навчання, програмістами, яким потрібний пристрій, для тестування і

компаніями, що займаються мікроелектронікою, яка в нинішній час розвивається.

Мікрокомп’ютери такого рівня використовуються в таких задач:

- розумний будинок, розумна парковка, розумні поливалки
- запуск Telegram-ботів для чатів або для проходження автентифікації
- запуск Веб-серверів або Веб-додатків
- обраховувати алгоритми, збирати і аналізувати дані
- створення хмарного середовища для перегляду відео і аудіо контенту,
- використання, як мережевого сховища для зберігання і обміну файлами

В таблиці 3.1 описані характеристики цього мікрокомп’ютера.

Таблиця 3.1 - Характеристики плати Orange Pi 3B

Процесор	Чотирьох ядерний, 64-розрядний процесор Rockchip RK3566 з тактовою частотою 1.8 Ггц з технологічним процесом 22 нм.
Графічний процесор	Вбудований графічний процесор ARM Mali G52 2EE з підтримкою OpenGL ES 1.1/2.0/3.2, OpenCL 2.0, Vulkan 1.1, підтримка 2D-прискорення.
NPU	Вбудований Ai-прискорювач RKNN NPU AI з продуктивністю 0.8Tops. Підтримка перетворення моделей архітектур Caffe/TensorFlow/TFLite/ONNX/PyTorch/Keras
Оперативна пам’ять	2/4/8 гігабайт стандарту LPDDR4/4X
Пам’ять програм	eMMC модуль від 16 до 256 гігабайт; SPI Flash:16 Мегабайт/32 Мегабайт; M.2 Sata або PCIe 2.0 NVME SSDs (Опціонально); Слот misroSD карти.
Відео вихід	1x HDMI 2.0 з підтримкою 4K, 1x MIPI DSI (2-смуговий інтерфейс) 1x eDP 1.3

Кінець таблиці 3.1 - Характеристики плати Orange Pi 3B

Аудіо вихід	3.5мм jack вхід/вихід
Мережевий інтерфейс	Wi-Fi5, BT5.0 з підтримкою BLE 10/100/1000 Мбіт Ehernet порт
Usb порти	1 x USB 2.0, 1 x USB 3.0 HOST, 2 x USB 2.0 HOST
Джерело живлення	Type-c 5В3А
Кнопки	1 x MaskROM, 1 x RESET, 1 x POWER
Підтримувані ОС	Android 11, Ubuntu 22.04, Ubuntu 20.04, Debian 11, Debian 12, OpenHarmony 4.0 Beta1, Orange Pi OS (Arch), Orange Pi OS (OH) на основі OpenHarmony

Такі характеристики обмежують використання комп'ютера в задачах, де потрібні потужні системи для обчислення великих даних. Зважаючи на це, певні задачі обчислюються набагато довше по часу, ніж на стандартному комп'ютері або ноутбук. Бувають випадки, коли для плати потрібно докуповувати специфічні деталі, такі як: камера, світлодіоди, датчики температур і тощо.

Через такі витрати, ціна стає вищою, але це все нівелюється компактністю, мобільністю, можливістю створювати кластери (об'єднання декількох комп'ютерів) для підвищення продуктивності і можливістю легко замінити у випадку браку або поломки пристрою.

Також основною перевагою таких комп'ютерів, є операційна система Linux. Ця операційна система використовується у всіх одноплатних комп'ютерах через мінімальні системні вимоги для запуску і можливістю користуватись нею через консоль. В певних

Orange Pi 3B буде використовуватись, як пристрій моніторингу мережевого трафіку.

Для реалізації проекту, потрібно обрати технології, які будуть використовуватись для оптимізації і розробки певних компонентів. Для

створення веб-сайту, для перегляду моніторингу даних, був використаний фреймворк Django.

Django – це веб-фреймворк на Python, який дозволяє розробникам створювати складні та масштабовані веб-додатки. Вперше був представлений у 2005 році, Django став одним з найпопулярніших інструментів для веб-розробки на Python, з активним використанням в різних галузях.

Django надає функцію вбудованого адміністративного інтерфейсу. Цей інтерфейс генерується на основі моделей даних. Це дозволяє легше керувати контентом без необхідності розробляти окремий інтерфейс.

Додатковими можливостями інтерфейсу є конфігурація і кастомізація, система автентифікації і прав доступу, інтернаціоналізація з локалізацією, безпека, масштабованість

Конфігурація і кастомізація дозволяє гнучко налаштовувати вигляд і функціональність, визначити поля, які будуть відображатись, додавати різні фільтри пошуку, сортування, а також створювати власні дії обробки даних.

Вбудована система автентифікації і прав доступу дозволяє налаштовувати різні рівні доступ для користувачів адміністративної панелі. Це дозволяє визначити, які моделі або поля будуть доступні для редагування з певними користувачами або групами.

Інтернаціоналізація з локалізацією дозволяє перекладати її інтерфейс різними мовами. Це дозволяє підтримувати проект розробникам, які знаходяться в різних країнах.

Virtualenv – це бібліотека, для створення ізольованих віртуальних середовищ. Надає можливість створення віртуальних середовищ, з кожним власним набором пакетів Python, інстальованих в каталогах і ними керувати. За бажанням, може ізолюватись від пакетів, що дозволяє використовувати, лише ті пакети, що були встановлені всередині віртуального середовища.

Такий інструмент дозволяє мати окремі версії Python та бібліотек для кожного проєкту, усуваючи проблеми між залежностями компонентів.

Бібліотека Flexmonster Pivot Table & Charts – це інструмент обробки, аналізу і візуалізації даних, у вигляді зведених таблиць і діаграм. Надає можливість для інтеграції у веб-додатки з широким набором функцій для аналізу великих обсягів даних.

Бібілотека Flexmonster підтримує різні формати даних:

- CSV: Текстові файли
- JSON: Структуровані дані в форматі JSON
- SQL: Дані з SQL-баз даних
- MongoDB: Документо-орієнтована база даних
- ElasticSearch: пошук і аналіз даних

Бібліотека Flexmonster Pivot Table & Charts – це інструмент обробки, аналізу і візуалізації даних, у вигляді зведених таблиць і діаграм. Надає можливість для інтеграції у веб-додатки з широким набором функцій для аналізу великих обсягів даних.

Flexmonster дозволяє працювати з різними видами фільтрації, з потужним API, яке повністю контролює зведену таблицю та діаграму. Надається інтеграція з різними мовами, часовими поясами, валютою і різними платформами, які використовуються для більш аналітичного рішення

Використовується Flexmonster в різних сферах аналітики, торгівлі і виробництві товару. Завдяки своїм можливостям весь процес оптимізується і зменшує витрати на весь аналіз даних.

Для розробки системної утиліти була обрана мова програмування Python, використання якої надає можливість ефективно створювати потужне ПЗ за рахунок наявності широкого спектру бібліотек.

Розроблена системна утиліта моніторингу складається з чотирьох основних програмних модулів, кожен з яких реалізує свою функцію у відповідності до поставленого технічного завдання:

1. Моніторинг загального використання мережі кінцевого IoT-пристрою.

2. Моніторинг використання мережі IoT-пристроєм з диференціалізацією за мережевими інтерфейсами.

3. Моніторинг мережевої активності процесів та використання обчислювальних ресурсів на IoT-пристрої.

Згідно архітектури програмного засобу, модулі вищеописаної системної утиліти встановлюються на одплатний міні-комп'ютер типу Raspberry Pi, Orange Pi, або інший сумісних пристрій на базі процесору з ARM архітектурою та під керуванням ОС на базі Linux.

Таким чином, компоненти системної утиліти дозволяють отримувати дані мережевого трафіку для перегляду в реальному часі, а також для подальшого зберігання і використання для аналізу.

Розглянемо модуль загального моніторингу використання мережі, для роботи якого спочатку засобами пакетного менеджера pip було інстальовано такі залежності: psutil, scapy та pandas.

Бібліотека Psutil забезпечує функціонал для моніторингу і збір інформації про систему і управління процесами. Надає засоби моніторингу ресурсів процесора, оперативної пам'яті, диску, мережі, сенсорів тощо, а також інструменти керування процесами (блокування, знищення, запуск, отримання даних тощо).

Бібліотека Scapy забезпечує роботу з мережею для обробки мережевих пакетів, які в збиранні і аналізі даних, мережевій діагностиці, тестуванні безпеки, тощо.

Pandas – бібліотека для аналізу та обробки даних. Використовується для структуруванні і іншими маніпуляціями з даними.

Після завантаження всіх бібліотек, було створено функцію, яка надає дані про: Завантаження, Вивантаження, Швидкість Завантаження, Швидкість вивантаження.

3.2 Розробка модулів загального моніторингу та моніторингу за інтерфейсами

Нижче наведено опис програмного коду для модулів загального моніторингу.

```
import time
import psutil
```

Ця частина коду призначена за імпорт необхідних модулів, для забезпечення доступу до функцій: управління часом time, збору системної інформації psutil.

```
UPDATE_DELAY = 1 # in seconds
```

В цій частині коду, була встановлена затримка в 1 секунду, для оновлення даних.

```
def get_size(bytes):
    """
    Returns size of bytes in a nice format
    """
    for unit in ['', 'K', 'M', 'G', 'T', 'P']:
        if bytes < 1024:
            return f"{bytes:.2f}{unit}B"
        bytes /= 1024
```

Функція get_size приймає один параметр bytes, який представляє розмір даних у байтах і повертає рядок з цим розміром в більш зручному форматі для користувача.

```
    for unit in ['', 'K', 'M', 'G', 'T', 'P']:
```

Цикл for, дозволяє пройти списку одиниць виміру. Записується це в зміну unit, яка по черзі приймає значення одиниць виміру.

```
if bytes < 1024:
    return f"{bytes:.2f}{unit}B"
bytes /= 1024
```

В цій частині коду відбувається перевірка даних на розмір. Якщо число менше за 1024, то воно повертає розмір у поточних одиницях виміру, відформатованих до двох десяткових знаків.

Якщо число дорівнює або більше за 1024, то воно ділиться на 1024 і конвертується в більшу одиницю виміру. Це дозволяє форматувати великі числові значення в більш читабельний формат, що спрощує їх сприйняття.

```
io = psutil.net_io_counters()
bytes_sent, bytes_recv = io.bytes_sent, io.bytes_recv
```

Функція psutil.net_io_counters(), отримує об’єкт, який містить інформацію про мережевий трафік. Цей об’єкт може мати велику кількість видів даних: передані і отримані байти, пакети, помилки, тощо.

```
bytes_sent, bytes_recv = io.bytes_sent, io.bytes_recv
```

В цій частині, витягуються два значення з об’єкта io. bytes_sent призначений за загальну кількість байтів, які відправлені з комп’ютера, а bytes_recv, за загальну кількість байтів, отриманих комп’ютером.

```
while True:
    time.sleep(UPDATE_DELAY)
    io_2 = psutil.net_io_counters()
    us, ds = io_2.bytes_sent - bytes_sent, io_2.bytes_recv
- bytes_recv
    print(f"Upload: {get_size(io_2.bytes_sent)}   "
          f", Download: {get_size(io_2.bytes_recv)}   "
          f",     Upload     Speed:     {get_size(us     /
UPDATE_DELAY)}/s   "
          f",     Download     Speed:     {get_size(ds     /
UPDATE_DELAY)}/s      ", end="\r")
    bytes_sent,     bytes_recv     =     io_2.bytes_sent,
io_2.bytes_recv
```

Цикл (Wheel True) починає нескінчений цикл з паузою, а (time.sleep) бере дані про паузу з UPDATE_DELAY, де встановлено затримку в 1 секунду.

Ця частина коду io_2 = psutil.net_io_counters() , викликається постійно для оновлення даних про поточну статистику мережевого трафіку. Також обчислюється швидкість відправки і швидкість отримання даних в us, ds = io_2.bytes_sent - bytes_sent, io_2.bytes_recv - bytes_recv

Потім функцією (print), виводиться інформація про загальну кількість відправлених даних, отриманих, швидкість відправлення і швидкість завантаження даних. Параметр end=”\r” дозволяє оновлювати інформацію без створення нових рядків.

```
bytes_sent, bytes_recv = io_2.bytes_sent, io_2.bytes_recv
```

Останній рядок коду оновлює значення з поточними для використання в наступній ітерації циклу. Це дозволяє обчислювати дані на основі змін між послідовними вимірюваннями.

Роботу функції загального моніторингу, було зображено на рисунку 3.1.

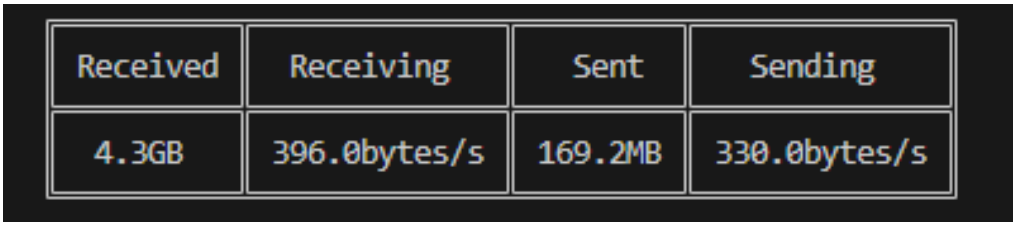


Рисунок 3.1 – Результат роботи функції загального моніторингу

Функція моніторингу по інтерфейсах дозволяє бачити не тільки інформацію, яка змінюється в цифрах, а більш детальний огляд, якщо в користувача є декілька мережевих інтерфейсів або підключено специфічний інтерфейс, такий як VirtualBox, тощо.

Робота коду:

```
import psutil
import time
import os
import pandas as pd
```

В цій частині коду було імпортованно декілька бібліотек. Psutil призначений для системної інформації, time для налаштування затримок між онолвеннями, os для очищення екарну залежно від операційої системи, а pandas, для створення табличних даних та їх зручного відображення.

```
def get_size(bytes):
    for unit in ['', 'K', 'M', 'G', 'T', 'P']:
        if bytes < 1024:
            return f"{bytes:.2f}{unit}B"
        bytes /= 1024
```

Всі данні записуються в unit, де записані основні параметри вимірювання даних.

Перевірка даних, якщо число менше за 1024, то воно повертає розмір у поточних одиницях виміру, відформатованих до двох десяткових знаків. Якщо більше або дорівнює, то ділить на 1024 і переводить у вище числове вимірювання.

```
io = psutil.net_io_counters(pernic=True)
```

В цій частині коду, використовується бібліотека psutil для отримання мережевої статистики. Аргумент pernic=True надає доступ до статистики кожного мережевого інтерфейсу в системі. Якщо встановити False, то функція поверне загальну статистику для всієї системи, об’єднавшим всі мережеві інтерфейси.

```
    for iface, iface_io in io.items():
        upload_speed,           download_speed           =
io_2[iface].bytes_sent       -       iface_io.bytes_sent,
io_2[iface].bytes_recv - iface_io.bytes_recv
```

Цикл for, який проходить через всі елементи словника io. Словник io містить статистику про мережеві інтерфейси, отриману за допомогою бібліотеки psutil. Ключ iface представляє собою ім’я мережевого інтерфейсу, а iface_io, значення, що містить статистику інтерфейсу об’єктом sent. Також обчислюється

швидкість передачі даних io_2[iface].bytes_sent, які проходять через інтерфейс iface_io.bytes_sent.

```
    io = io_2
    df = pd.DataFrame(data)
    df.sort_values("Download", inplace=True, ascending=False)
    os.system("cls") if "nt" in os.name else os.system("clear")
    print(df.to_string())
```

Відбувається оновлення зміни io, яка містить статистику мережевих операцій, новими даними io_2. Це робиться для наступної ітерації циклу, щоб дані оновлювались.

Створюється DataFrame, який є об’єктом pandas для представлення табличних данних. Після створення цей об’єкт буде відсортовано за стовпцем (Download) в порядку спадання.

Інтерфейс, який буде отримувати дані найбільше, буде першим, а інтерфейс з наймешним показником, буде останнім.

Рядок з os_system викликає команду, для очищення екрану залежно від операційної системи. Команда cls, використовується для Windows, а команда clear для Unix-подібних систем, таких як Linux або MacOs. Це дозволяє запускати код на різних операційних системах і отримувати повну інформацію, уникаючи певних дій, які повинні відбуватись на різних операційних системах.

Вкінці об’єкт DataFrame виводиться у вигляді тексту функцією print методом to_string.

Результат роботи функції моніторингу даних по інтерфейсах, було зображено на рисунку 3.2.

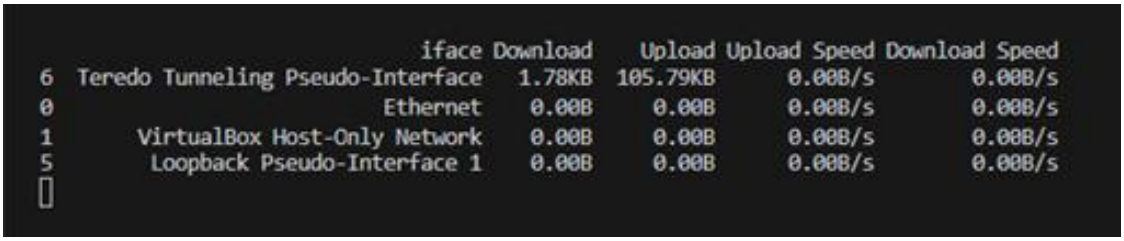


Рисунок 3.2 – Моніторинг даних трафіку за мережевими інтерфейсами

Код моніторингу процесів

```
from scapy.all import *
import psutil
from collections import defaultdict
import os
from threading import Thread
import pandas as pd
```

Імпорт необхідних модулів: Для захоплення і обробки мережевих пакетів імпортується Scapy.all. Psutil використовується для отримання інформації про процеси та з'єднання. Defaultdict використовується для зручного створення словників зі значеннями. os для системних команд, a Threads для роботи з потоками. Pandas для обробки та відображення даних у вигляді таблиці.

```
all_macs = {iface.mac for iface in ifaces.values()}
connection2pid = {}
pid2traffic = defaultdict(lambda: [0, 0])
global_df = None
is_program_running = True
```

Для MAC-адрес мережевих інтерфейсів використовується all_macs. Словник connection2pid використовується для зберігання відповідності між мережевими з'єднаннями та PID-процесів. Словником pid2traffic зберігає обсяг завантаження та вивантаження даних процесами. Для зберігання попередньої статистики трафіку використовується global_df. Глобальна змінна для контролю стану програми використовується is_program_running.

```
def process_packet(packet):
```

```
    global pid2traffic
    try:
and ports
        packet_connection = (packet.sport, packet.dport)
    except (AttributeError, IndexError):
        pass
    else:
our `connection2pid` global dictionary
        packet_pid = connection2pid.get(packet_connection)
        if packet_pid:
            if packet.src in all_macs:
our MAC address
upload
            else:
                pid2traffic[packet_pid][1] += len(packet)
```

Функція process_packet обробляє захоплені мережеві пакети. Пробує витягнути IP-адреси, порти джерела та призначення пакета.

Також ігнорує пакети без TCP/UDP шарів, визначає PID процесу, що відповідає за дане з'єднання і збільшує обсяг завантаження або вивантаження в залежності від пакеті, який може бути вихідним або вхідним.

```
def get_connections():
    global connection2pid
    while is_program_running:
        for c in psutil.net_connections():
            if c.laddr and c.raddr and c.pid:
                connection2pid[(c.laddr.port,
c.raddr.port)] = c.pid
                connection2pid[(c.raddr.port,
c.laddr.port)] = c.pid
        time.sleep(1)
```

Функція get_connections постійно оновлює словник, що відображає мережеві з'єднання на процеси. Ця функція використовує psutil для отримання списку всіх мережевих з'єднань. Для кожного з'єднання перевіряє наявність локальної та віддаленої адреси. Затримка в 1 секунду між ітераціями для зменшення навантаження на систему.

```
    os.system("cls") if "nt" in os.name else
os.system("clear")
    # print our dataframe
    print(printing_df.to_string())
    global_df = df
```

Ця частина коду, виконує очистку екранку в операційній системі. Якщо операційна система Windows, то виконається команда cls, у випадку операційної системи Linux або Mac OS, то команда clear. Після очищення екрану, виводиться інформація на екран і глобального оновлюється.

```
def print_stats():
    """Simple function that keeps printing the stats"""
    while is_program_running:
        time.sleep(1)
        print_pid2traffic()
```

Функція print_stats працює в нескінченному циклі викликаючи функцію print_pid2traffic з інтервалом в 1 секунду.

```
if __name__ == "__main__":
    printing_thread = Thread(target=print_stats)
    printing_thread.start()
    connections_thread = Thread(target=get_connections)
    connections_thread.start()
    print("Started sniffing")
    sniff(prn=process_packet, store=False)
    is_program_running = False
```

В останній частині коду всієї функції моніторингу по процесам, запускаються два потоки. Перший відображає статистику print_stats, а другий оновлює з'єднання get_connections. Після цього відбувається запуск процес захоплення пакетів за допомогою sniff(prn=process_packet, store=False, з функцією виклику для обробки кожного пакету. Результат роботи функції моніторингу даних по процесах було зображено на рисунку 3.3. Для функції моніторингу мережі по процесам, була можливість доповнення, яке дозволяє бачити кожний процес, який запущений від програми. Це дозволяє бачити, який процес використовує більше простору оперативної пам'яті або швидкості передачі даних, незалежно від іншої. На рисунку 3.4 зображено роботу модифікованої функції, коли кожний процес, який запущений від операційної системи Windows або браузера Google Chrome.

pid	name	create_time	Upload	Download	Upload Speed	Download Speed
17528	msedge.exe	2022-05-15 08:11:41.815511	118.22KB	626.72KB	2.89KB/s	87.41KB/s
12312	chrome.exe	2022-05-14 18:16:35.777317	15.52KB	199.48KB	1.33KB/s	30.36KB/s
14652	msedge.exe	2022-05-15 08:11:41.176197	10.56KB	25.44KB	3.90KB/s	408.00B/s
2308	svchost.exe	2022-05-14 18:14:24.516395	436.00B	1.38KB	0.00B/s	0.00B/s
21420	AdobeGCClient.exe	2022-05-18 10:12:02.422741	2.17KB	1.19KB	400.00B/s	406.00B/s

Рисунок 3.3 – Моніторинг даних по процесах

pid	name	cpu_usage	memory_usage	read_bytes	write_bytes	status	create_time	nice	n_threads	cores
15684	chrome.exe	0.0	643.83MB	42.93GB	4.79GB	running	2019-09-01 09:14:13	32	18	8
15164	explorer.exe	0.0	419.27MB	25.67GB	13.24GB	running	2019-09-01 09:13:22	32	160	8
2936	chrome.exe	0.0	395.21MB	36.71GB	53.56GB	running	2019-09-01 09:14:13	32	37	8
19812	chrome.exe	0.0	306.08MB	255.51MB	413.47MB	running	2019-09-01 09:46:01	32	18	8
19116	chrome.exe	0.0	302.96MB	44.55MB	83.12MB	running	2019-09-09 02:59:32	64	21	8
19476	chrome.exe	0.0	286.45MB	1.31GB	799.89MB	running	2019-09-01 09:14:14	32	16	8
2284	chrome.exe	0.0	278.95MB	216.56MB	262.83MB	running	2019-09-05 14:08:05	64	16	8
12100	python.exe	0.0	267.84MB	9.04MB	0.00B	running	2019-09-09 11:09:06	32	11	8
11412	chrome.exe	0.0	217.90MB	127.80MB	225.71MB	running	2019-09-02 22:31:39	64	18	8
15440	SkypeApp.exe	0.0	216.31MB	16.64MB	56.02MB	stopped	2019-09-01 09:13:25	32	44	8
9864	SearchUI.exe	0.0	160.31MB	260.27MB	153.55MB	stopped	2019-09-01 09:13:24	32	57	8
17096	chrome.exe	0.0	158.11MB	244.03MB	920.99MB	running	2019-09-01 09:57:14	64	17	8
7044	chrome.exe	0.0	152.77MB	8.03MB	2.95MB	running	2019-09-09 11:07:07	64	14	8
15784	chrome.exe	0.0	117.86MB	9.14GB	14.79GB	running	2019-09-01 09:14:13	32	16	8
6852	chrome.exe	0.0	90.52MB	4.21MB	5.54MB	running	2019-09-07 22:31:16	64	15	8
20796	chrome.exe	0.0	85.88MB	127.31MB	180.10MB	running	2019-09-07 13:13:38	64	14	8
15408	WWAHost.exe	0.0	80.60MB	535.31KB	8.00B	stopped	2019-09-01 09:19:09	32	28	8
19268	chrome.exe	0.0	80.17MB	8.09MB	10.87MB	running	2019-09-08 23:36:56	64	15	8
1712	chrome.exe	0.0	80.00MB	69.65MB	154.73MB	running	2019-09-07 11:17:51	64	16	8
3656	chrome.exe	0.0	74.92MB	43.88MB	92.43MB	running	2019-09-07 12:16:29	64	15	8
13556	chrome.exe	0.0	74.31MB	7.08MB	7.63MB	running	2019-09-06 23:45:50	32	13	8
5940	ctfmon.exe	0.0	72.78MB	134.58KB	0.00B	running	2019-09-01 09:13:25	128	11	8
11920	chrome.exe	0.0	70.77MB	1009.56KB	1.13MB	running	2019-09-09 10:46:31	32	15	8
17628	python.exe	0.0	65.67MB	8.57MB	1.12MB	running	2019-09-08 18:53:34	32	4	8
18116	AcroRd32.exe	0.0	62.93MB	41.10MB	385.21KB	running	2019-09-08 18:01:38	32	23	8

Рисунок 3.4 – Результат роботи функції моніторингу процесів

3.5 Висновки до третього розділу

В третьому розділі бакалаврської кваліфікаційної роботи було реалізовано опис апаратної платформи на базі одноплатного мікрокомп’ютера Orange Pi для розгортання програмного засобу.

Виконана програмна реалізації трьох модулів для моніторингу мережевого трафіку, кожний з яких використовується відповідно своїх функцію: загальний моніторинг переданих та прийнятих даних, а також відповідні швидкості передачі для Інтернет-пристрою; деталізація моніторингу параметрів за мережевими інтерфейсами та системними процесами.

ВИСНОВКИ

В ході виконання кваліфікаційної роботи бакалавра було розроблено прототип програмного засобу моніторингу мережевого трафіку для розмумних пристроїв Інтернету речей. В результаті проведеного аналізу відомих засобів моніторингу комп'ютерних систем та мереж була сформована специфікація вимог до програмного засобу та основні варіанти її використання кінцевим користувачем – системним адміністратором. Головним елементом програмного засобу є системна утиліта, яка встановлюється на Інтернет-пристрій з обмеженими обчислювальними ресурсами та керується операційною системою на балі Linux. До основних функції утиліти входить збір параметрів даних мережевого трафіку з деталізацією за мережевими інтерфейсами та процесами для кінцевих Інтернет пристроїв з подальшим записом параметрів а базу даних. Також у відповідності до технічного завдання, була розроблена схема бази даних та розробка її програмної моделі засобами мови Python та СКБД SQlite.

В першому розділі бакалаврської кваліфікаційної роботи було проведено аналіз предметної області згідно теми дослідження, зокрема було проаналізовано види та потреба існуючих методів моніторингу мережевого трафіку та активності кінцевих IoT-пристроїв з позиції системного адміністрування та протидії кіберзагрозам. Проведено аналіз найбільш типових кібератак, якими можуть бути уражені системи Інтернету речей. Виконано порівняльний аналіз найбільш відомих програмних засобів моніторингу мереж, виділено переваги та недоліки існуючих програмних засобів. За результатами розділу були сформульовані практичні задачі для вирішення в кваліфікаційній роботі.

В другому розділі бакалаврської кваліфікаційної роботи було визначено, головні вимоги до функціоналу програмного засобу. Відповідно до основних вимог була створена модель варіантів використання програмного засобу кінцевим користувачем – системним адміністратором. Додатково було визначено нефункціональні вимогами, реалізація яких в подальшому дозволить покращити

взаємодію між користувачем і програмним засобом. В процесі проєктування були створенні схеми мереж в кросплатформенній програмі візуального моделювання мережі Cisco Packet Tracer та схема бази даних параметрів моніторингу трафіку, які були використані для створення програмного модуля для роботи з базою даних на мові Python.

В третьому розділі бакалаврської кваліфікаційної роботи було реалізовано опис апаратної платформи на базі одноплатного мікрокомп'ютера Orange Pi для розгортання програмного засобу.

Виконана програмна реалізації трьох модулів для моніторингу мережевого трафіку, кожний з яких використовується відповідно своїх функцію: загальний моніторинг переданих та прийнятих даних, а також відповідні швидкості передачі для Інтернет-проистрою; деталізація моніторингу параметрів за мережевими інтерфейсами та системними процесами.

Подальша робота над проєктом пов'язана з розробкою інтерфейсу користувача, тестуванням та удосконаленням системної утиліти моніторингу та розгортання прототипу засобу.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. How to Make a Network Usage Monitor in Python. URL: https://thepythoncode.com/article/make-a-network-usage-monitor-in-python. (дата звернення: 21.06.2024).

2. Моніторинг мережі. Протоколи, найкращі практики, інструменти 2021. URL: https://eska.global/blog/setevoj-monitoring-protokoly-luchshie-praktiki-instrumenty-2020 (дата звернення: 21.06.2024).

3. IoT Monitor Traffic: Unveiling a Smarter Approach to Monitoring Traffic. URL: https://www.temok.com/blog/iot-monitor-traffic/ (дата звернення: 21.06.2024).

4. Network Security Monitoring. URL: https://foresite.com/blog/network-security-monitoring/. (дата звернення: 21.06.2024).

5. Deeks, M. A Review on Botnet Attacks. Preprints 2023, 2023070366. DOI: http://dx.doi.org/10.20944/preprints202307.0366.v1. (дата звернення: 21.06.2024).

6. Falowo O. I., Ozer M., Li C. and Abdo J. B. Evolving Malware and DDoS Attacks: Decadal Longitudinal Study, *in IEEE Access*. vol. 12. pp. 39221-39237. 2024. DOI: https://doi.org/10.1109/ACCESS.2024.3376682. (дата звернення: 21.06.2024).

7. Faris H., Al-Zoubi A. M., Heidari A. A., Aljarah I., Mafarja M., Hassonah M. A., and Fujita H. An intelligent system for spam detection and identification of the most relevant features based on evolutionary random weight networks. *Information Fusion.* 2019. 67–83. DOI: https://doi.org/10.1155/2022/186288. (дата звернення: 21.06.2024).

8. Wang, Z.; Zhang, Y.; Chen, Y.; Liu, H.; Wang, B.; Wang, C. A Survey on Programmable Logic Controller Vulnerabilities, Attacks, Detections, andForensics. *Processes*. 2023. 918. DOI: https://doi.org/10.3390/pr11030918. (дата звернення: 21.06.2024).

9. Stellios I., Kotzanikolaou P., Psarakis M., Alcaraz C., Lopez J. A Survey of IoT-Enabled Cyberattacks: Assessing Attack Paths to Critical Infrastructures and Services, *in IEEE Communications Surveys & Tutorials*, pp. 3453-3495. 2018. DOI: https://doi.org/10.1109/COMST.2018.2855563. (дата звернення: 21.06.2024).

10. Alhussen A., Engin A. Avoiding data loss and corruption for file transfers with Fast Integrity Verification. *Journal of Parallel and Distributed Computing*. 2021. P. 33-44. DOI: https://doi.org/10.1016/j.jpdc.2021.02.002. (дата звернення: 21.06.2024).

11. Avijit M. Man-in-the-middle-attack: Understanding in simple words. *Cyberspace: Jurnal Pendidikan Teknologi Informasi* 2.2. 2019. P. 109-134. DOI: http://dx.doi.org/10.22373/cj.v2i2.3453. (дата звернення: 21.06.2024).

12. Zhang C., Zhou S., Chain BM. Hybrid epidemics—A case study on computer worm conficker. *PloS one* 10.5. 2015). DOI: 10.1371/journal.pone.0127478. (дата звернення: 21.06.2024).

13. Sinanović H., Mrdovic S. Analysis of Mirai malicious software. *25th International Conference on Software, Telecommunications and Computer Networks*. 2017. URL: https://ieeexplore.ieee.org/abstract/document/8115504/authors#authors. (дата звернення: 21.06.2024).

14. Chen YH., Chang A., Huang CW. Using learning time as metrics: an artificial intelligence driven risk assess framework to evaluate DDoS cyber attack. *Journal of Intelligent & Fuzzy Systems* 40.4. 2021. P. 7691-7699. URL: https://www.wired.com/story/github-ddos-memcached/. (дата звернення: 21.06.2024).

15. Herzog S. Revisiting the Estonian cyber attacks: Digital threats and multinational responses. *Journal of Strategic Security* 4.2 .2011. P. 49-60. URL: https://www.jstor.org/stable/26463926?seq=6. (дата звернення: 21.06.2024).

16. Pitney AM, Penrod S., Foraker ., Bhunia S. A systematic review of 2021 microsoft exchange data breach exploiting multiple vulnerabilities. *7th international conference on smart and sustainable technologies* 2022. DOI: https://doi.org/10.23919/SpliTech55088.2022.9854268. (дата звернення: 21.06.2024).

17. Marelli M. The SolarWinds hack: Lessons for international humanitarian organizations. *International Review of the Red Cross* 104.919. 2022 P. 1267-1284.

URL: https://ieeexplore.ieee.org/abstract/document/8115504 (дата звернення: 21.06.2024).

18. Kang C., Abbas F., Oh H. Protection scheme for IoT devices using introspection. 2015 6th *International Conference on the Network of the Future*. 2015. DOI: 10.1109/NOF.2015.7333292 (дата звернення: 21.06.2024).

19. Hwang K., Dongarra J., Fox G. Distributed and cloud computing: from parallel processing to the internet of things. Morgan kaufmann, 2013.

20. Zhu Q., Wang R., Chen Q., Liu Y., Qin W. IOT Gateway: BridgingWireless Sensor Networks into Internet of Things. 2010 *IEEE/IFIP International Conference on Embedded and Ubiquitous Computing*. 2010. P. 347-352, DOI: 10.1109/EUC.2010.58.. (дата звернення: 21.06.2024).

21. Ammar M., Crispo B., Tsudik G. Simple: A remote attestation approach for resource-constrained iot devices. 2020 ACM/IEEE *11th International Conference on Cyber-Physical Systems*. 2020.DOI: https://doi.org/10.1109/ICCPS48487.2020.00036. (дата звернення: 21.06.2024).

22. Akinlolu A., Adedolyn O., Ademola A., Olukayode A. A Comparative Study of Operating Systems: Case of Windows, UNIX, Linux, Mac, Android and iOS. 2020. URL: https://www.researchgate.net/profile/Adedoyin-Odumabo/publication/372400705_A_Comparative_Study_of_Operating_Systems_Case_of_Windows_UNIX_Linux_Mac_Android_and_iOS/links/64b41d62c41fb852dd7b65e1/A-Comparative-Study-of-Operating-Systems-Case-of-Windows-UNIX-Linux-Mac-Android-and-iOS.pdf. (дата звернення: 21.06.2024).

23. Wireshark User's Guide. URL: https://www.wireshark.org/ docs/wsug_html_chunked/. (дата звернення: 21.06.2024).

24. Nmap Reference Guide. URL: https://nmap.org/book/man.html. (дата звернення: 21.06.2024).

25. Metasploit Documentation. URL: https://docs.metasploit.com/. (дата звернення: 21.06.2024).

26. Tenable Nessus. URL: https://docs.tenable.com/Nessus.htm#Tenable-Nessus. (дата звернення: 21.06.2024).

27. OpenVAS. URL: https://greenbone.github.io/docs/latest/. (дата звернення: 21.06.2024).

28. Kali Docs. Offical Documentation. URL: https://www.kali.org/docs/ (дата звернення: 21.06.2024).

29. John the Ripper password cracker. URL: https://www.openwall.com/john/doc/. (дата звернення: 21.06.2024).

30. Wifite Tool Documentation. URL: https://www.kali.org/tools/wifite/. (дата звернення: 21.06.2024).

31. Kismet. URL: https://www.kismetwireless.net/docs/. (дата звернення: 21.06.2024).

32. Nikto. URL: https://www.cirt.net/Nikto2 (дата звернення: 21.06.2024).

33. Maltego guides. URL: https://docs.maltego.com/support/home. (дата звернення: 21.06.2024).

34. TCPDUMP MAN PAGE. URL: https://www.tcpdump.org/manpages/tcpdump.1.html.(дата звернення: 21.06.2024).

35. Hping3 – Linux man page. URL: https://linux.die.net/man/8/hping3. (дата звернення: 21.06.2024).

36. Basic Features of Orange Pi PC. URL: http://www.orangepi.org/orangepiwiki/index.php/Orange_Pi_PC. (дата звернення: 21.06.2024).

37. Raj, Jennifer S., and Mr C. Vijesh Joe. "Wi-Fi network profiling and QoS assessment for real time video streaming." IRO Journal on Sustainable Wireless Systems 3.1 (2021): 21-30. URL: https://d1wqtxts1xzle7.cloudfront.net/76164280/03-libre.pdf?1639301910=&response-content-disposition=inline%3B+filename%3DWi_Fi_Network_Profiling_and_QoS_Assessme.pdf&Expires=1718957287&Signature=D4o7AZoMNCpEuqHsWHfd0YhhQ0WGWd4gEG6kEdeu0CKNeafJ4QoPPA~kLf~nmVlBFdWgZPZ1bhJoCuPEM3ShvgCakJ54-

RfgAWsW0p-tf0u8ds7hiFTe6qMvRHuyfAGmYDfbPlUvqyFpkSPNQe1C25b05deaXhenbeCTPam95NKJla2hk0t0Ki81YQ3JvmXRhaqr~ZcSwRHeEXCOcNEVZsrCxHzPFwXkKJfSf1939wmpgRK6eMMkCR6J4yZFoFtlV8~5Ph9ts~KxDbYcjiz~uhp3TyVU1GEfhsiTm2ebJIcs8eWvadLtW0AqXbMA8dgAyGhz~xtrvWGT5OqoSQIR6g__&Key-Pair-Id=APKAJLOHF5GGSLRBV4ZA. (дата звернення: 21.06.2024).

38. Eridani D., Rochim A., Cesara F. Comparative performance study of ESP-NOW, Wi-Fi, bluetooth protocols based on range, transmission speed, latency, energy usage and barrier resistance. 2021. *international seminar on application for technology of information and communication*. 2021. DOI: https://doi.org/10.1109/iSemantic52711.2021.9573246. (дата звернення: 21.06.2024).

39. Reibel J-J. Networking: Book, Independently published. English 2024. 314p.

40. Chou Eri-c. Mastering Python Networking - Third Edition: Your one-stop solution to using Python for network automation, programmability, and DevOps 4rd: Packt Publishing; 4rd ed.edition. English. 595p.

41. Комп'ютерні мережі, системне адміністрування та кібербезпека. Матеріали модульного середовища ХНУ. URL. https://msn.khmnu. edu.ua/course/view.php?id=7709. (дата звернення: 21.06.2024).

42. Orange Pi 3B. URL: http://www.orangepi.org/html/hardWare/ computerAndMicrocontrollers/service-and-support/Orange-Pi-3B.html. (дата звернення: 21.06.2024).

43. Django documentation. URL: https://docs.djangoproject.com/en/5.0/. (дата звернення: 21.06.2024).

44. Virtualenv documentation. URL: https://virtualenv.pypa.io/en/latest/. (дата звернення: 21.06.2024).

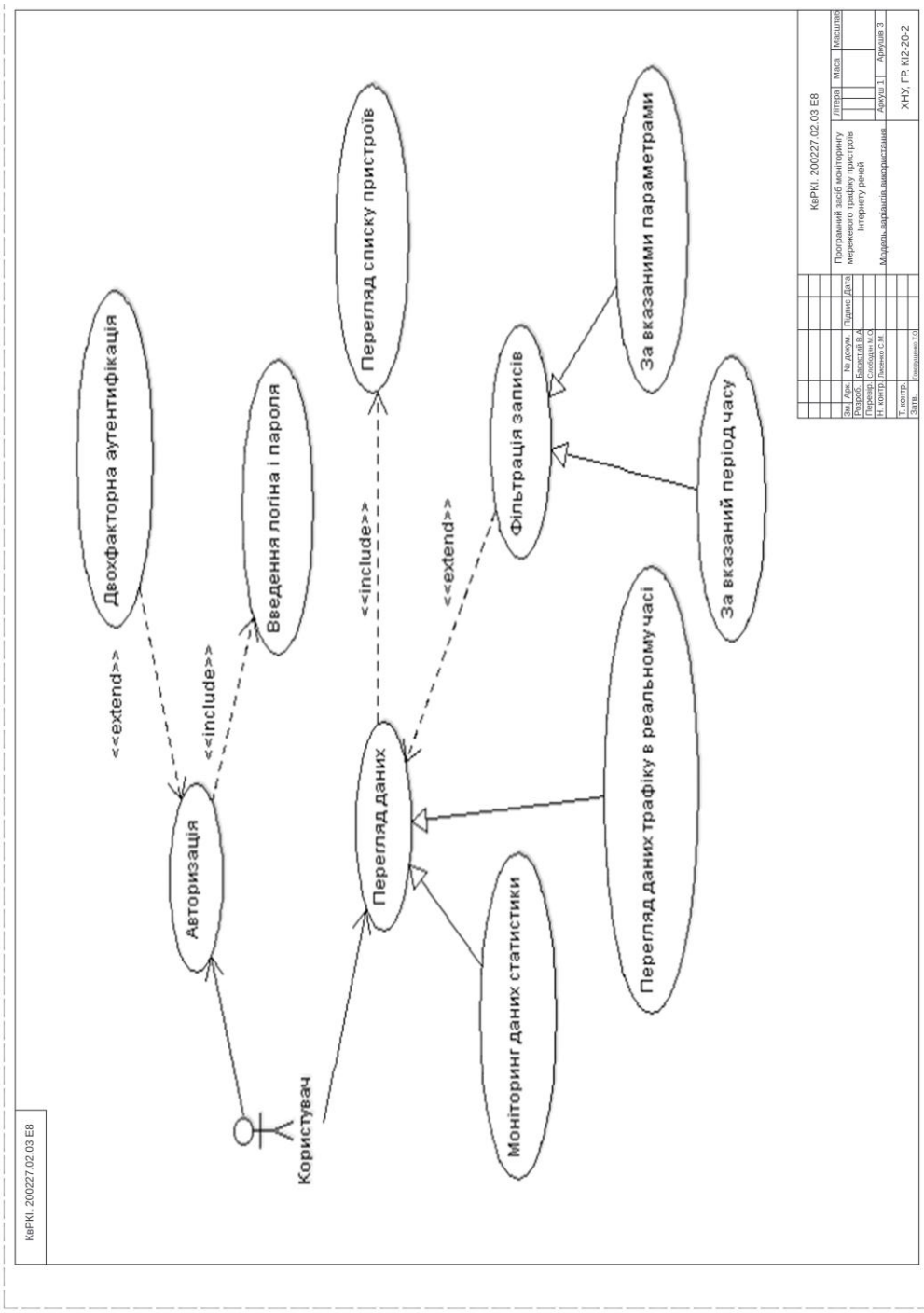
45. Flexmonster documentation. URL: https://www.flexmonster.com/doc/. (дата звернення: 21.06.2024).

46. SQLite Documentation. URL: https://www.sqlite.org/docs.html. (дата звернення: 21.06.2024).

Додаток А

(обов’язковий)

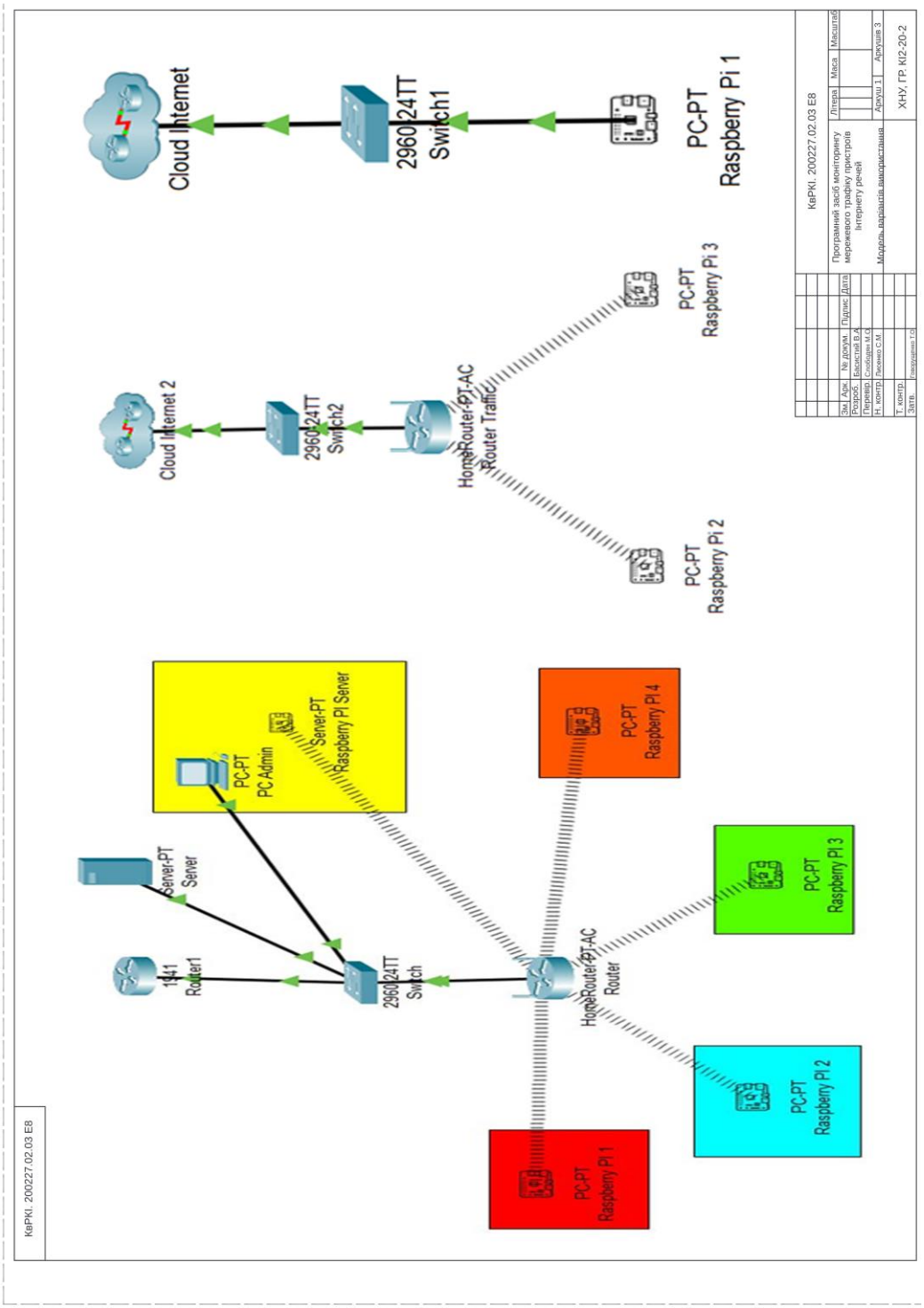
Копія креслення «Модель варіантів використання»



Додаток Б

(обов’язковий)

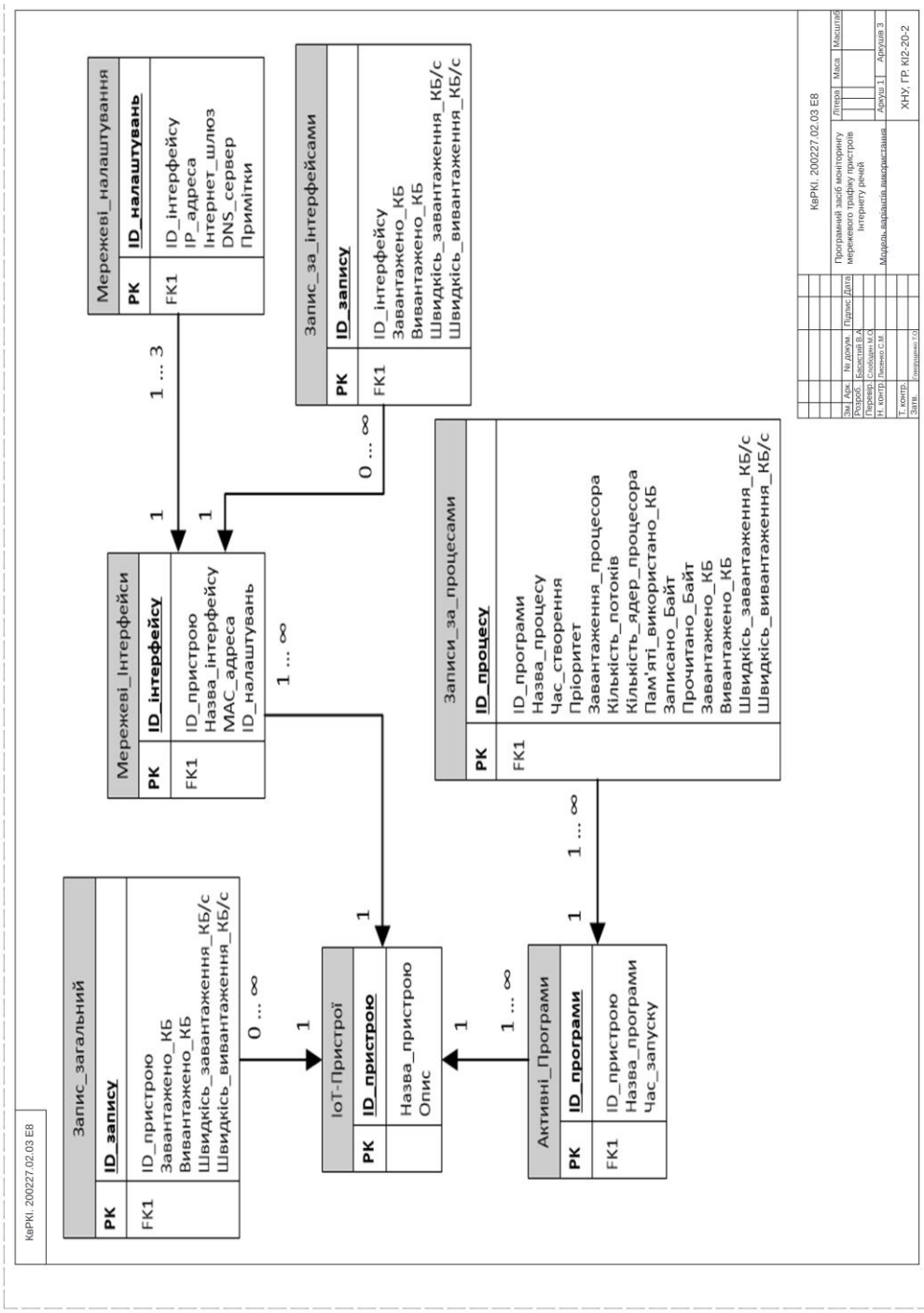
Копія креслення «Схема мережі IoT»



Додаток В

(обов’язковий)

Копія креслення «Схема бази даних»



Додаток Г

(обов'язковий)

Код бази даних

```
from sqlalchemy import Column, Integer, Float, DateTime, String, ForeignKey,
Table
from sqlalchemy.orm import relationship, backref
from sqlalchemy.ext.declarative import declarative_base

Base = declarative_base()

# IoT-пристрій
class Device(Base):
    __tablename__ = "device"
    device_id = Column(Integer, primary_key=True) # первинний ключ
    dev_name = Column(String) # назва притсрою
    dev_descr = Column(String) # опис пристрою
    general_logs = relationship("General_Log", backref=backref("device"))
    active_apps = relationship("Active_App", backref=backref("device"))
    network_interfaces = relationship("Network_Interface",
backref=backref("device"))

# запис загальний
class General_Log(Base):
    __tablename__ = "general_log"
    general_log_id = Column(Integer, primary_key=True)
    dev_id = Column(Integer, ForeignKey("device.device_id"))
    download_KB = Column(Float) # завантажено_кб
    upload_KB = Column(Float) # вивантажено_кб
    download_speed_KB_s = Column(String) # швидкість завантаження кб/с
    upload_speed_KB_s = Column(String) # швидкість вивантаження кб/с

# активна програма
class Active_App(Base):
    __tablename__ = "active_app"
    active_app_id = Column(Integer, primary_key=True)
    dev_id = Column(Integer, ForeignKey("device.device_id"))
    app_name = Column(String) # назва пристрою
    start_time = Column(DateTime) # час запуску
    process_logs = relationship("Process_Log", backref=backref("active_app"))
```

```
# мережевий інтерфейс
class Network_Interface(Base):
    __tablename__ = "network_interface"
    network_interface_id = Column(Integer, primary_key=True)
    dev_id = Column(Integer, ForeignKey("device.device_id"))
    netw_interf_name = Column(String) # назва інтерфейсу
    mac_address = Column(String) # mac-адреса
    start_time = Column(DateTime) # час запуску
    network_if_logs            =            relationship("Network_Interface_Log",
backref=backref("network_interface"))
    network_settings              =              relationship("Network_Settings",
backref=backref("network_interface"))

# записи за процесами
class Process_Log(Base):
    __tablename__ = "process_log"
    proc_log_id = Column(Integer, primary_key=True)
    active_app_id = Column(Integer, ForeignKey("active_app.active_app_id"))
    system_id = Column(Integer) # ID в системі
    proc_name = Column(String) # назва процесу
    create_time = Column(DateTime) # час створення
    priority = Column(Integer) # пріоритет процесу
    cpu_usage = Column(Integer) # завантаження процесу
    thread_num = Column(Integer) # кількість потоків
    core_num = Column(Integer) # кількість ядер процесора
    mem_usage_kb = Column(Float) # використано пам'яті, кб
    wrote_kb = Column(Float) # записано на диск, кб
    read_kb = Column(Float) # прочитано з диска, кб
    download_KB = Column(Float) # завантажено_кб
    upload_KB = Column(Float) # вивантажено_кб
    download_speed_KB_s = Column(String) # швидкість завантаження кб/с
    upload_speed_KB_s = Column(String) # швидкість вивантаження кб/с

# запис загальний
class Network_Interface_Log(Base):
    __tablename__ = "network_interface_log"
    network_if_log_id = Column(Integer, primary_key=True)
    network_interface_id                    =                    Column(Integer,
ForeignKey("network_Interface.network_interface_id"))
    download_KB = Column(Float) # завантажено_кб
    upload_KB = Column(Float) # вивантажено_кб
    download_speed_KB_s = Column(String) # швидкість завантаження кб/с
```

```
    upload_speed_KB_s = Column(String) # швидкість вивантаження кб/с


class Network_Settings(Base):
    __tablename__ = "network_settings"
    netw_set_id = Column(Integer, primary_key=True)
    ip_address = Column(String)
    gateway = Column(String)
    dns = Column(String)
    notes = Column(String)
    network_interface_id                    =                    Column(Integer,
ForeignKey("network_Interface.network_interface_id"))
```

Додаток Д

(обов’язковий)

Код програмного засобу

```
import psutil
import time

UPDATE_DELAY = 1 # in seconds

def get_size(bytes):
    """
    Returns size of bytes in a nice format
    """
    for unit in ['', 'K', 'M', 'G', 'T', 'P']:
        if bytes < 1024:
            return f"{bytes:.2f}{unit}B"
        bytes /= 1024

# get the network I/O stats from psutil
io = psutil.net_io_counters()
# extract the total bytes sent and received
bytes_sent, bytes_recv = io.bytes_sent, io.bytes_recv

while True:
    # sleep for `UPDATE_DELAY` seconds
    time.sleep(UPDATE_DELAY)
    # get the stats again
    io_2 = psutil.net_io_counters()
    # new - old stats gets us the speed
    us, ds = io_2.bytes_sent - bytes_sent, io_2.bytes_recv - bytes_recv
    # print the total download/upload along with current speeds
    print(f"Upload: {get_size(io_2.bytes_sent)}   "
          f", Download: {get_size(io_2.bytes_recv)}   "
          f", Upload Speed: {get_size(us / UPDATE_DELAY)}/s   "
          f", Download Speed: {get_size(ds / UPDATE_DELAY)}/s      ", end="\r")
    # update the bytes_sent and bytes_recv for next iteration
    bytes_sent, bytes_recv = io_2.bytes_sent, io_2.bytes_recv
```

```
import psutil
import time
import os
import pandas as pd

UPDATE_DELAY = 1 # in seconds

def get_size(bytes):
    """
    Returns size of bytes in a nice format
    """
    for unit in ['', 'K', 'M', 'G', 'T', 'P']:
        if bytes < 1024:
            return f"{bytes:.2f}{unit}B"
        bytes /= 1024

# get the network I/O stats from psutil on each network interface
# by setting `pernic` to `True`
io = psutil.net_io_counters(pernic=True)

while True:
    # sleep for `UPDATE_DELAY` seconds
    time.sleep(UPDATE_DELAY)
    # get the network I/O stats again per interface
    io_2 = psutil.net_io_counters(pernic=True)
    # initialize the data to gather (a list of dicts)
    data = []
    for iface, iface_io in io.items():
        # new - old stats gets us the speed
        upload_speed,  download_speed  =  io_2[iface].bytes_sent  -
iface_io.bytes_sent, io_2[iface].bytes_recv - iface_io.bytes_recv
        data.append({
            "iface":                iface,                "Download":
get_size(io_2[iface].bytes_recv),
            "Upload": get_size(io_2[iface].bytes_sent),
```

```
            "Upload      Speed":      f"{get_size(upload_speed      /
UPDATE_DELAY)}/s",
            "Download     Speed":     f"{get_size(download_speed     /
UPDATE_DELAY)}/s",
        })
    # update the I/O stats for the next iteration
    io = io_2
    # construct a Pandas DataFrame to print stats in a cool tabular
style
    df = pd.DataFrame(data)
    # sort values per column, feel free to change the column
    df.sort_values("Download", inplace=True, ascending=False)
    # clear the screen based on your OS
    os.system("cls") if "nt" in os.name else os.system("clear")
    # print the stats
    print(df.to_string())



from scapy.all import *
import psutil
from collections import defaultdict
import os
from threading import Thread
import pandas as pd

# get the all network adapter's MAC addresses
all_macs = {iface.mac for iface in ifaces.values()}
# A dictionary to map each connection to its correponding process ID
(PID)
connection2pid = {}
# A dictionary to map each process ID (PID) to total Upload (0) and
Download (1) traffic
pid2traffic = defaultdict(lambda: [0, 0])
# the global Pandas DataFrame that's used to track previous traffic
stats
```

```
global_df = None
# global boolean for status of the program
is_program_running = True

def get_size(bytes):
    """
    Returns size of bytes in a nice format
    """
    for unit in ['', 'K', 'M', 'G', 'T', 'P']:
        if bytes < 1024:
            return f"{bytes:.2f}{unit}B"
        bytes /= 1024


def process_packet(packet):
    global pid2traffic
    try:
        # get the packet source & destination IP addresses and ports
        packet_connection = (packet.sport, packet.dport)
    except (AttributeError, IndexError):
        # sometimes the packet does not have TCP/UDP layers, we just
ignore these packets
        pass
    else:
        # get the PID responsible for this connection from our
`connection2pid` global dictionary
        packet_pid = connection2pid.get(packet_connection)
        if packet_pid:
            if packet.src in all_macs:
                # the source MAC address of the packet is our MAC
address
                # so it's an outgoing packet, meaning it's upload
                pid2traffic[packet_pid][0] += len(packet)
            else:
                # incoming packet, download
```

```
                pid2traffic[packet_pid][1] += len(packet)


def get_connections():
    """A function that keeps listening for connections on this
machine
    and adds them to `connection2pid` global variable"""
    global connection2pid
    while is_program_running:
        # using psutil, we can grab each connection's source and
destination ports
        # and their process ID
        for c in psutil.net_connections():
            if c.laddr and c.raddr and c.pid:
                # if local address, remote address and PID are in
the connection
                # add them to our global dictionary
                connection2pid[(c.laddr.port, c.raddr.port)] = c.pid
                connection2pid[(c.raddr.port, c.laddr.port)] = c.pid
        # sleep for a second, feel free to adjust this
        time.sleep(1)


def print_pid2traffic():
    global global_df
    # initialize the list of processes
    processes = []
    for pid, traffic in pid2traffic.items():
        # `pid` is an integer that represents the process ID
        # `traffic` is a list of two values: total Upload and
Download size in bytes
        try:
            # get the process object from psutil
            p = psutil.Process(pid)
        except psutil.NoSuchProcess:
```

```
            # if process is not found, simply continue to the next
PID for now
            continue
        # get the name of the process, such as chrome.exe, etc.
        name = p.name()
        # get the time the process was spawned
        try:
            create_time = datetime.fromtimestamp(p.create_time())
        except OSError:
            # system processes, using boot time instead
            create_time = datetime.fromtimestamp(psutil.boot_time())
        # construct our dictionary that stores process info
        process = {
            "pid": pid, "name": name, "create_time": create_time,
"Upload": traffic[0],
            "Download": traffic[1],
        }
        try:
            # calculate the upload and download speeds by simply
subtracting the old stats from the new stats
            process["Upload Speed"] = traffic[0] - global_df.at[pid,
"Upload"]
            process["Download Speed"] = traffic[1] -
global_df.at[pid, "Download"]
        except (KeyError, AttributeError):
            # If it's the first time running this function, then the
speed is the current traffic
            # You can think of it as if old traffic is 0
            process["Upload Speed"] = traffic[0]
            process["Download Speed"] = traffic[1]
        # append the process to our processes list
        processes.append(process)
    # construct our Pandas DataFrame
    df = pd.DataFrame(processes)
    try:
```

```
            # set the PID as the index of the dataframe
            df = df.set_index("pid")
            # sort by column, feel free to edit this column
            df.sort_values("Download", inplace=True, ascending=False)
        except KeyError as e:
            # when dataframe is empty
            pass
        # make another copy of the dataframe just for fancy printing
        printing_df = df.copy()
        try:
            # apply the function get_size to scale the stats like
'532.6KB/s', etc.
            printing_df["Download"]                                    =
printing_df["Download"].apply(get_size)
            printing_df["Upload"]                                      =
printing_df["Upload"].apply(get_size)
            printing_df["Download    Speed"]    =    printing_df["Download
Speed"].apply(get_size).apply(lambda s: f"{s}/s")
            printing_df["Upload     Speed"]     =     printing_df["Upload
Speed"].apply(get_size).apply(lambda s: f"{s}/s")
        except KeyError as e:
            # when dataframe is empty again
            pass
        # clear the screen based on your OS
        os.system("cls") if "nt" in os.name else os.system("clear")
        # print our dataframe
        print(printing_df.to_string())
        # update the global df to our dataframe
        global_df = df


def print_stats():
    """Simple function that keeps printing the stats"""
    while is_program_running:
        time.sleep(1)
```

```
        print_pid2traffic()



if __name__ == "__main__":
    # start the printing thread
    printing_thread = Thread(target=print_stats)
    printing_thread.start()
    # start the get_connections() function to update the current
connections of this machine
    connections_thread = Thread(target=get_connections)
    connections_thread.start()
    # start sniffing
    print("Started sniffing")
    sniff(prn=process_packet, store=False)
    # setting the global variable to False to exit the program
    is_program_running = False
```

Оновлений код:

```
import psutil
from datetime import datetime
import pandas as pd
import time
import os


def get_size(bytes):
    """
    Returns size of bytes in a nice format
    """
    for unit in ['', 'K', 'M', 'G', 'T', 'P']:
        if bytes < 1024:
            return f"{bytes:.2f}{unit}B"
        bytes /= 1024
```

```
def get_processes_info():
    # the list the contain all process dictionaries
    processes = []
    for process in psutil.process_iter():
        # get all process info in one shot
        with process.oneshot():
            # get the process id
            pid = process.pid
            if pid == 0:
                # System Idle Process for Windows NT, useless to see
anyways
                continue
            # get the name of the file executed
            name = process.name()
            # get the time the process was spawned
            try:
                create_time                                          =
datetime.fromtimestamp(process.create_time())
            except OSError:
                # system processes, using boot time instead
                create_time                                          =
datetime.fromtimestamp(psutil.boot_time())
            try:
                # get the number of CPU cores that can execute this
process
                cores = len(process.cpu_affinity())
            except psutil.AccessDenied:
                cores = 0
            # get the CPU usage percentage
            cpu_usage = process.cpu_percent()
            # get the status of the process (running, idle, etc.)
            status = process.status()
            try:
```

```
                # get the process priority (a lower value means a
more prioritized process)
                nice = int(process.nice())
            except psutil.AccessDenied:
                nice = 0
            try:
                # get the memory usage in bytes
                memory_usage = process.memory_full_info().uss
            except psutil.AccessDenied:
                memory_usage = 0
            # total process read and written bytes
            io_counters = process.io_counters()
            read_bytes = io_counters.read_bytes
            write_bytes = io_counters.write_bytes
            #  get  the  number  of  total  threads  spawned  by  this
process
            n_threads = process.num_threads()
            # get the username of user spawned the process
            try:
                username = process.username()
            except psutil.AccessDenied:
                username = "N/A"

        processes.append({
            'pid': pid, 'name': name, 'create_time': create_time,
            'cores':   cores,   'cpu_usage':   cpu_usage,   'status':
status, 'nice': nice,
            'memory_usage':  memory_usage,  'read_bytes':  read_bytes,
'write_bytes': write_bytes,
            'n_threads': n_threads, 'username': username,
        })

    return processes
```

```
def construct_dataframe(processes):
    # convert to pandas dataframe
    df = pd.DataFrame(processes)
    # set the process id as index of a process
    df.set_index('pid', inplace=True)
    # sort rows by the column passed as argument
    df.sort_values(sort_by, inplace=True, ascending=not descending)
    # pretty printing bytes
    df['memory_usage'] = df['memory_usage'].apply(get_size)
    df['write_bytes'] = df['write_bytes'].apply(get_size)
    df['read_bytes'] = df['read_bytes'].apply(get_size)
    # convert to proper date format
    df['create_time']  =  df['create_time'].apply(datetime.strftime,
args=("%Y-%m-%d %H:%M:%S",))
    # reorder and define used columns
    df = df[columns.split(",")]
    return df

if __name__ == "__main__":
    import argparse
    parser  =  argparse.ArgumentParser(description="Process  Viewer  &
Monitor")
    parser.add_argument("-c", "--columns", help="""Columns to show,
                                                available        are
name,create_time,cores,cpu_usage,status,nice,memory_usage,read_bytes
,write_bytes,n_threads,username.
                                                Default            is
name,cpu_usage,memory_usage,read_bytes,write_bytes,status,create_tim
e,nice,n_threads,cores.""",

default="name,cpu_usage,memory_usage,read_bytes,write_bytes,status,c
reate_time,nice,n_threads,cores")
    parser.add_argument("-s",      "--sort-by",      dest="sort_by",
help="Column   to   sort   by,   default   is   memory_usage   .",
default="memory_usage")
```

```
    parser.add_argument("--descending",          action="store_true",
help="Whether to sort in descending order.")
    parser.add_argument("-n", help="Number of processes to show,
will show all if 0 is specified, default is 25 .", default=25)
    parser.add_argument("-u", "--live-update", action="store_true",
help="Whether to keep the program on and updating process
information each second")

    # parse arguments
    args = parser.parse_args()
    columns = args.columns
    sort_by = args.sort_by
    descending = args.descending
    n = int(args.n)
    live_update = args.live_update
    # print the processes for the first time
    processes = get_processes_info()
    df = construct_dataframe(processes)
    if n == 0:
        print(df.to_string())
    elif n > 0:
        print(df.head(n).to_string())
    # print continuously
    while live_update:
        # get all process info
        processes = get_processes_info()
        df = construct_dataframe(processes)
        # clear the screen depending on your OS
        os.system("cls") if "nt" in os.name else os.system("clear")
        if n == 0:
            print(df.to_string())
        elif n > 0:
            print(df.head(n).to_string())
        time.sleep(0.7)
```

UNICHECK
by Turnitin

Ім'я користувача:
Кафедра КІ

ID перевірки:
1016376072

Дата перевірки:
19.06.2024 18:58:22 EEST

Тип перевірки:
Doc vs Internet + Library

Дата звіту:
19.06.2024 18:58:57 EEST

ID користувача:
100005591

Назва документа: **Басистий_Програмний засіб моніторингу мережевого трафіку пристроїв Інтернету речей**

Кількість сторінок: **76** Кількість слів: **13245** Кількість символів: **100486** Розмір файлу: **2.05 MB** ID файлу: **1016184091**

Виявлено модифікації тексту (можуть впливати на відсоток схожості)

13.9%
Схожість

Найбільша схожість: **6.97%** з Інтернет-джерелом (https://www.thepythoncode.com/article/make-a-network-usage-mon...

12.2% Джерела з Інтернету	750	Сторінка 78
2.65% Джерела з Бібліотеки	174	Сторінка 81

1.36% Цитат

Цитати	2	Сторінка 82

Не знайдено жодних посилань

0%
Вилучень

Немає вилучених джерел

Модифікації

Виявлено модифікації тексту. Детальна інформація доступна в онлайн-звіті.

Замінені символи	2
Підозріле форматування	22 сторінки

Anti-Plagiarism v-15.257

Максимальне співпадіння з одним документом 1.0%

Словники перевірки: en_US, ru_RU, ua_UA. **Помилок в документах: 16%**

	Документ		Сумарний збіг по Базі Даних	
ID: 131626 Назва: БКР Програмний засіб моніторингу мережевого трафіку пристроїв Інтернету речей Додано в БД: 2024-06-19 Автора: В. А. Басистий Керівники: М. О. Слободян Консультанти: Опоненти:	Символи	Лексеми	Символи	Лексеми
	67327	925	1475 (2%)	18 (2%)

Джерело плагіату

ID	Опис	Наявність плагіату в документі	
		Символи	Лексеми

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

РЕЦЕНЗІЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ

Дипломник: Басистий Віталій Анатолійович

Тема: Програмний засіб моніторингу мережевого трафіку пристроїв Інтернету речей

Спеціальність: 123 «Комп'ютерна інженерія»

Обсяг кваліфікаційної роботи:

Кількість листів креслень 3 Кількість сторінок записки 73

1. Короткий зміст роботи та прийнятих рішень: Метою роботи є розробка програмного засобу моніторингу мережевого трафіку, який підвищує ефективність роботи системного адміністратора щодо підтримки застосувань Інтернету речей.

2. Висновок про відповідність роботи дипломному завданню: Робота повністю відповідає поставленому завданню.

3. Характеристика виконання кожного розділу, ступінь використання останніх досягнень науки і техніки і передових методів роботи:

В ході виконання роботи було проведено аналіз вимог до програмного засобу моніторингу мережевого трафіку пристроїв Інтернету речей; дано загальну характеристику та розробити структурно схеми кінцевого пристрою IoT на базі розумного сенсору; розроблено схему мережі IoT, де відображено підключення IoT-пристроїв до локальної мережі та до мережі Інтернет; розроблено модель даних та схему бази даних для моніторингу трафіку; розроблено програмний засіб для отримання параметрів трафіку Інтернет-пристроїв з подальшим записом їх в базу даних.

4. Позитивні сторони роботи: висока практична цінність роботи.

5. Негативні сторони роботи: відсутність реалізації графічного інтерфейсу користувача.

6. Оцінка графічного оформлення та пояснювальної записки роботи: Пояснювальна записка оформлена коректно, згідно діючих стандартів оформлення документації.

7. Відгук про роботу в цілому: Робота виконана на належному технічному рівні.

8. Інші зауваження: _____

9. Оцінка дипломної роботи: відмінно

Рецензент (прізвище, ім'я, по батькові, посада, місце роботи) Сегох М.В. др. філософії, зав. каф. КібБезпеки

"20" червня 2024 р. _____ (підпис)

Завідувачу кафедри КІІС
д-р.техн.наук, проф. Говорущенко Т. О.

Басистого Віталія Анатолійовича

ПІБ здобувача вищої освіти

ФІТ, 4 курсу, групи КІ2-20-2

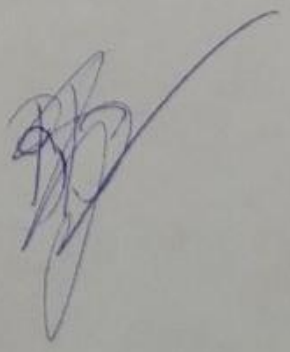
ЗАЯВА

З правилами чинного Положення «Про систему забезпечення академічної доброчесності у Хмельницькому національному університеті» від 01.07.2022, згідно з яким виявлення плагіату є підставою для відмови в допуску кваліфікаційної роботи до захисту та застосування заходів дисциплінарної та академічної відповідальності, ознайомлений (а). Про використання програмно-технічних засобів для перевірки кваліфікаційних робіт здобувачів вищої освіти на плагіат оповіщений(а) та надаю свою згоду на обробку та збереження університетом моєї роботи в інституційному репозитарії університету.

Також надаю університету право на передачу моєї роботи для обробки та збереження в базах даних програмно-технічних засобів (Unicheck та Anti-Plagiarism) та використання роботи для виявлення плагіату в інших роботах, які перевіряються програмно-технічними засобами та користувачами, що мають доступ до цих програмно-технічних засобів, виключно в обмежених цілях для виявлення плагіату в текстах робіт.

Робота для перевірки університетом надається в друкованому та електронному варіанті. Електронна версія моєї роботи збігається (ідентична) з друкованою.

18 червня 2024 року



РІШЕННЯ ЕКСПЕРТНОЇ КОМІСІЇ

КАФЕДРИ КОМП'ЮТЕРНОЇ ІНЖЕНЕРІЇ ТА ІНФОРМАЦІЙНИХ СИСТЕМ

ПРО ДОПУСК КВАЛІФІКАЦІЙНОЇ РОБОТИ ДО ЗАХИСТУ

Підтверджуємо ознайомлення з результатом звіту подібності щодо роботи, генерованого системою виявлення текстових збігів/ідентичності/схожості:

Назва: Програмний засіб моніторингу мережевого трафіку пристроїв Інтернету речей

Автор: Басистий Віталій Анатолійович

Спеціальність: 123 – Комп'ютерна інженерія

Освітня програма: освітньо-професійна

Науковий керівник: Слободян Максим Олегович

Після аналізу звіту подібності зроблено такий висновок:

№	Висновок	Позначка про відповідність
1	Запозичення, виявлені в роботі, є законними і не є плагіатом. Робота приймається до захисту.	відповідає
2	Виявлені запозичення не є плагіатом, розміщені в розділах, які не описують безпосередньо авторське дослідження, але кількість цитат перевищує обсяг, виправданий поставленою метою роботи Робота приймається до захисту, але має бути відкоригована. Відкоригований варіант має бути поданий на кафедру за 2 дні до захисту, разом із заявою щодо самостійності виконання письмової роботи та ідентичності друкованої та електронної версії роботи	
3	Виявлені запозичення не є плагіатом, але частково розміщені в розділах, які описують безпосередньо авторське дослідження, а кількість цитат перевищує обсяг, виправданий поставленою метою роботи. В зв'язку з цим мета роботи та поставлені завдання не були досягнені. Робота може бути допущена до захисту (наступного року) після того як буде відкоригована та допрацьована і успішно пройде повторну перевірку на академічний плагіат.	
4	Робота містить навмисні текстові спотворення, передбачувані спроби укриття запозичень або інші прояви академічного плагіату. Робота містить фабрикацію або фальсифікацію даних. Робота не допускається до захисту.	

Підтвердження:

Запозичення, виявлені в роботі, є законними і не є плагіатом, оскільки:

1) максимальна схожість з одним Інтернет-джерелом становить 6,97% і стосуються виключно фрагментів програмного коду типових прикладів, які наводяться в технічній документації та навчальний ресурсах для розробників, співпадіння.

2) всі зафіксовані системою ознаки модифікації тексту відносяться до оформлення лістингів програмного коду, де використовуються нерегламентовані комбінації латинських літер, спеціальних символів та цифр.

Сумарний обсяг всіх запозичень, визначений системою виявлення збігів/ ідентичності/схожості, складає 13.9% і адресується до 924 першоджерела, що, з урахуванням наведених обгрунтувань, відповідає характеру унікального проєкту і свідчить на користь кваліфікаційної роботи.

Керівник роботи _____ М. О. Слободян

Гарант ОП _____ С. М. Лисенко

Завідувач кафедри КІІС _____ Т. О. Говорущенко