Хмельницький національний університет
Факультет інформаційних технологій
Кафедра кібербезпеки

КВАЛІФІКАЦІЙНА РОБОТА

Колби Сергія Сергійовича

на здобуття ступеня вищої освіти Бакалавра

Система аналізу
активності користувачів локальної мережі

Галузь знань 12 – Інформаційні технології
Спеціальність 125 – Кібербезпека
Освітня програма Кібербезпека

Шифр КРБКБ.200110.20.01.11 ПЗ

Виконав студент 4 курсу група КБ-20-1 Сергій КОЛБА
Керівник канд. техн. наук, доцент Юрій КЛЬОЦ
Нормоконтролер старший викладач Сергій МОСТОВИЙ

До захисту допускаю:
Завідувач кафедри кібербезпеки Юрій КЛЬОЦ

19 06 2024 р.

Хмельницький 2024

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет Інформаційних технологій

Кафедра Кібербезпеки

Рівень вищої освіти Бакалавр

Галузь знань 12 – Інформаційні технології

Спеціальність 125 – Кібербезпека

Освітня програма Кібербезпека

ЗАТВЕРДЖУЮ

Завідувач кафедри кібербезпеки

Юрій КЛЬОЦ

15 лютого 2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Колбі Сергію Сергійовичу

1 Тема роботи Система аналізу активності користувачів локальної мережі

Керівник роботи Кльоц Юрій Павлович

Затверджено наказом ректора університету від 15 лютого 2024 № 8

2 Строк подання студентом кваліфікаційної роботи на кафедру

3 Вихідні дані до роботи розробити програмне забезпечення для збору даних про активність користувачів у локальній мережі та передачу інформації у базу даних з можливістю пошуку за певними критеріями та аналізом їх активності

4 Зміст пояснювальної записки (перелік питань, які потрібно розробити)
1. Визначення необхідного апаратного та програмного забезпечення. 2. Визначення способу отримання логів з мережних пристроїв. 3. Створення програмного забезпечення та бази даних. 3. Приклад функціонування розробленого програмного забезпечення. 4. Аналіз активності на основі отриманих даних

5 Перелік графічного матеріалу (із зазначенням обов'язкових креслень)
1. Фізична модель бази даних. 2. Даталогічна модель бази даних.
3. Загальна схема функціональності програми.

6 Консультанти розділів кваліфікаційної роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Нормоконтроль	Мостовий С.В., старший викладач кафедри кібербезпеки		

7 Дата видачі завдання 16 лютого 2024 р.

КАЛЕНДАРНИЙ ПЛАН

Назва етапів (розділів) кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
Вибір і затвердження теми кваліфікаційної роботи	16.02.2024	
Ознайомлення з предметною областю	20.02.2024	
Дослідження існуючих рішень	25.02.2024	
Постановка задачі	15.03.2024	
Визначення загальних принципів рішення задачі	23.03.2024	
Деталізація принципів рішення задачі	04.04.2024	
Розробка проєктних рішень	18.04.2024	
Апробація проєктних рішень	10.05.2024	
Оформлення пояснювальної записки згідно вимог	20.05.2024	
Оформлення графічної частини	3.06.2024	
Захист КР	Червень	

Студент _____ Сергій КОЛБА

Керівник кваліфікаційної роботи _____ Юрій КЛЬОЦ

АНОТАЦІЯ

Курсовий проект: Система аналізу активності користувачів локальної мережі

Автор роботи: Колба С.С.

Керівник роботи: Кльоц Ю. П.

Обсяг: 66 с., 28 рис., 2 додатка, 40 джерело.

Графічна частина: 12 презентаційних слайдів

АНАЛІЗ АКТИВНОСТІ, БАЗА ДАНИХ, ЗБІР ДАНИХ, ЛОГИ, ЛОГУВАННЯ АКТИВНОСТІ, ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ

Мета курсового проєкту: розробити програмне забезпечення для збору даних про активність користувачів у локальній мережі та передачу інформації у базу даних з можливістю пошуку за певними критеріями та аналізу їх активності.

Для досягнення цих цілей було проведено аналіз предметної області, ознайомлення із необхідними термінами і поняттями, розгляд засобів віддаленого збору логів. Також була проведена поетапна розробка логування в мережі, що включає у себе налаштування маршрутизатора Mikrotik, проєктування та розробка бази даних та програмного забезпечення. Завершальним етапом став огляд функціональності логування та програмного забезпечення, а також пояснення загального алгоритму функціонування програми.

Загалом, було обрано обладнання та програмне забезпечення для збору логів, налаштувано обладнання для забезпечення проходження дозволеного та блокування забороненого трафіку. Розроблено базу даних та програмне забезпечення із можливістю збору, перегляду та аналізу активності користувачів, а також протестована функціональність програмне забезпечення.

12.06.2024 _____

ANNOTATION

Course project: System for analyzing the activity of local network users

Author of the work: Kolba Sergii Sergiiovich.

Supervisor: Klots Yurii Pavlovych.

Amount: 66 pages, 28 figures, 2 appendices, 40 sources.

Graphic part: 12 presentation slides.

ACTIVITY ANALYSIS, DATABASE, DATA COLLECTION, LOGS, ACTIVITY LOGGING, SOFTWARE

Course project objective: to develop software for collecting data on user activity on a local network and transferring information to a database that can be searched by certain criteria and analyzed.

To achieve these goals, an analysis of the subject area was conducted, familiarization with necessary terms and concepts was undertaken, and the means of remote log collection were explored. Additionally, a phased development of network logging was carried out, involving configuration of Mikrotik router settings, database design and development, and software development. The final stage involved reviewing the operation of logging and software, along with explaining the overall functioning algorithm of the program.

In summary, hardware and software were selected for log collection, equipment was configured to allow permitted traffic and block prohibited traffic, and a database and software were developed to gather, view, and analyze user activity, with the software's functionality also tested.

12.06.2024 _____

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ, ТЕРМІНІВ

IP – Internet Protocol, інтернет протокол

MAC – Media Access Control, Hardware Address, фізична адреса

CIDR – Classless Inter-Domain Routing, Безкласова маршрутизація

DNS – Domain Name System, система доменних імен

DHCP – Dynamic Host Configuration Protocol, протокол динамічної конфігурації вузла

NAT – Network Address Translation, перетворення мережевих адрес

TCP – Transmission Control Protocol, протокол управління передачею

UDP – User Datagram Protocol, протокол датаграм користувача

LAN – Local Area Network, локальна мережа

ICMP – Internet Control Message Protocol, міжмережевий протокол керуючих повідомлень

GUI – graphical user interface, графічний інтерфейс користувача

SIEM – Security information and event management, інформація про безпеку та управління подіями

SRM – Supplier Relationship Management, Управління взаємовідносинами з постачальниками

IoT – Internet of Things, інтернет речі

SDR – Software-defined radio, програмно обумовлена радіосистема

SAAS – Software as a service, програма як послуга

SNMP – Simple Network Management Protocol, простий протокол мережевого управління

IPMI – Intelligent Platform Management Interface, інтелектуальний інтерфейс управління платформою

CRM – Customer relationship management, управління відносин з клієнтами

ВСТУП

Системи логування відіграють важливу роль у виявленні та запобіганні загрозам безпеці. Вони дозволяють виявляти аномалії у мережевій активності, що можуть свідчити про спроби несанкціонованого доступу або кібератаки, а також проводити аудит безпеки завдяки детальним записам про дії користувачів і події в системі. Відстеження змін у конфігураціях системи також є важливим аспектом, що допомагає виявляти несанкціоновані зміни.

Крім того, ефективні системи логування сприяють покращенню продуктивності мережі. Вони дозволяють моніторити використання ресурсів, таких як процесорний час, пам'ять та мережевий трафік, що допомагає вчасно виявляти та усувати проблеми, пов'язані з перевантаженнями або недостатніми ресурсами. Аналіз продуктивності різних компонентів мережі дозволяє оптимізувати їх роботу та планувати масштабування.

У цій кваліфікаційній роботі реалізовано налаштування маршрутизатора Mikrotik, а саме: доступності трафіку, правил брандмауера, реалізація відправлення зібраних даних на віддалений сервер з метою занесення інформації в базу даних. Також було розроблено фізичну та датологічну модель бази даних та її створення. Ще одним проектом розробки був додаток, який приймає данні з маршрутизатора та заносить їх у базу даних, звідки і проектує інформацію на графічний інтерфейс.

Метою даної кваліфікаційної роботи реалізація можливості збору даних активності користувачів у локальній мережі та передачу інформації у базу даних з можливістю пошуку за певними критеріями, що допоможе у аналізуванні трафіку та наданні інформації для створення статистики активності.

Таким чином, ефективність систем логування у мережі полягає у їх здатності забезпечувати високий рівень безпеки, оптимізувати продуктивність, відповідати нормативним вимогам та підтримувати оперативне управління.

1 ДОСЛІДЖЕННЯ ПРЕДМЕТНОЇ ОБЛАСТІ. АНАЛІЗ ЗАВДАННЯ ТА ПОШУК ТЕОРЕТИЧНОЇ ІНФОРМАЦІЇ

1.1 Ознайомлення та аналіз предметної області

Відповідність нормативним вимогам також є важливим аспектом ефективності систем логування. Багато організацій зобов'язані дотримуватись нормативних вимог щодо зберігання та обробки даних. Системи логування забезпечують наявність аудиторських записів, що дозволяє відповідати вимогам аудиту, а також збереження даних протягом визначених періодів, необхідних для відповідності регулятивним стандартам.

Окрім цього, системи логування полегшують управління мережею та вирішення проблем. Вони допомагають швидко виявляти і діагностувати проблеми у мережі, що зменшує час простою і підвищує стабільність системи. Збережені логи дозволяють проводити історичний аналіз, що може бути корисним для прогнозування проблем і планування оновлень або змін у мережі.

Важливою функцією таких систем є можливість генерації звітів та сповіщень у реальному часі. Це дозволяє оперативно реагувати на інциденти та мінімізувати потенційні ризики. Наприклад, система може сповістити адміністратора про підозрілу активність, таку як спроба доступу до конфіденційних даних у неробочий час або незвичний обсяг переданих даних.

Інтеграція систем аналізу активності з іншими засобами безпеки, такими як антивірусне програмне забезпечення, міжмережеві екрани та системи виявлення вторгнень, дозволяє створити комплексну систему захисту мережі. Це забезпечує багаторівневий підхід до безпеки, де різні засоби працюють разом для виявлення та нейтралізації загроз. Більш досконалі детектори сприяють виявленню характеристик змінних частин логів, включаючи особливості числових і категоріальних полів [1].

1.2 Необхідні терміни та поняття

Файли логування являють собою спеціалізовані типи файлів даних, що містять інформацію про події, які відбулися в певний момент часу, збираючи її з різних джерел. Ці файли зберігають дати, часові позначки та інші метадані, що стосуються події або доступних на той момент даних. Майже всі програмні додатки та системи, включаючи операційні системи і веб-сервери, створюють файли журналів, що дозволяє зрозуміти діяльність системи та взаємодію користувачів з нею. Наприклад, логування доступу на веб-сервері може записувати IP-адреси відвідувачів, час їхніх візитів та переглянуті сторінки. Ця інформація допомагає власникам сайтів покращувати продуктивність, оптимізувати створення контенту і знижувати витрати. Адміністратори мереж використовують файли логів для моніторингу несанкціонованих входів або змін паролів, вирішення питань безпеки і забезпечення доступу до конфіденційних даних тільки авторизованим користувачам.

Логи часто записуються у файли Syslog, які є найпоширенішим стандартом логування у програмних фреймворках та операційних системах. Ці файли можуть бути структурованими або неструктурованими і містять метадані для покращення пошуку та швидшої роботих[2].

1.2.1 Визначення необхідного апаратного та програмного забезпечення

Матеріально-технічне забезпечення охоплює фізичні пристрої, такі як комп'ютери, сервери, маршрутизатори та інше обладнання, яке використовується для створення і функціонування системи. Програмне забезпечення, в свою чергу, включає набір програм, що встановлюються на ці пристрої для виконання різних функцій та операцій, зокрема операційних систем, додатків, програм для моніторингу та управління мережею.

Щодо матеріально-технічного забезпечення, яке стосується даної кваліфікаційної роботи, необхідно розглянути комутатор, який включає функції моніторингу та логування подій на своєму пристрої. Один з найбільш популярних комутаторів від MikroTik - це модель CRS328-24P-4S+RM (рисунок 1.1). Він оснащений 24 портами Gigabit Ethernet з підтримкою технології PoE+ та 4 портами SFP+. Цей комутатор працює під управлінням операційної системи RouterOS, яка надає широкий спектр можливостей для налаштування та управління мережевим обладнанням. Крім того, CRS328-24P-4S+RM має вбудований захист від перевантажень та коротких замикань, а також підтримує стандарти безпеки мережевого обладнання, забезпечуючи високий рівень надійності та безпеки мережі.

В університеті для доступу до інтерфейсів комутаторів MikroTik і перегляду логів підключення до певних портів та списку користувачів використовується програмне забезпечення Winbox. Для налаштування базових параметрів на пристроях Mikrotik потрібна програма Winbox, у якій також здійснюються всі подальші операції [4].

Winbox – це інтерфейс для управління RouterOS, операційною системою на основі Linux, що використовується для налаштування апаратних маршрутизаторів Mikrotik Router Board (рисунок 1.2). Завдяки Winbox можна здійснювати моніторинг та управління мережею. RouterOS є універсальною системою, яка дозволяє реалізовувати мережеві проекти малого та середнього масштабу. Обладнання від Mikrotik забезпечує можливість створення високоякісних WiFi-мереж[4]. Усі функції інтерфейсу WinBox максимально віддзеркалюють функції консолі. Деякі розширені та критичні для системи налаштування неможливі з WinBox, як-от зміна MAC-адреси в інтерфейсі [5].

WinBox працює за принципом клієнт-сервер, де клієнтське програмне забезпечення встановлюється на комп’ютері адміністратора, а серверна частина – на пристрої MikroTik. Програма використовує власний протокол для зв'язку з маршрутизаторами, що забезпечує високу швидкість і безпеку передачі даних.

У підсумку, WinBox є потужним та зручним інструментом для адміністрування мережевих пристроїв MikroTik. Завдяки своїм широким можливостям і простоті використання, він допомагає адміністраторам забезпечувати надійність, безпеку та ефективність роботи мережі, що робить його незамінним у сучасних мережевих середовищах.

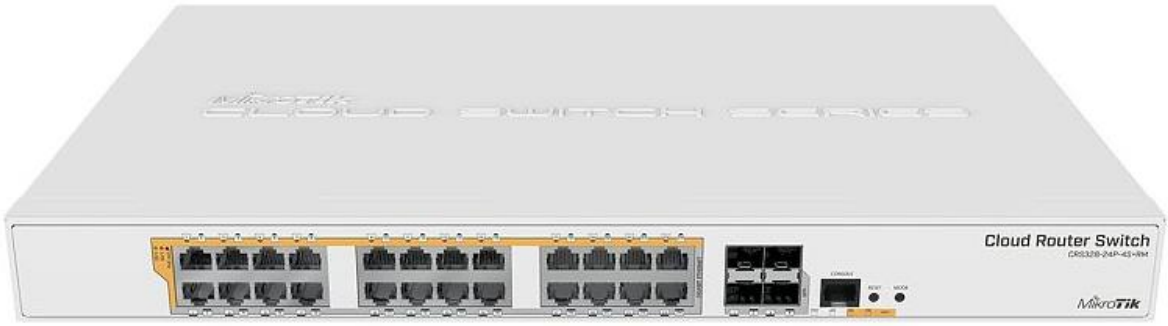


Рисунок 1.1 – Комутатор MikroTik CRS328-24P-4S+RM

Visual Syslog Server для Windows – це безкоштовна програма з відкритим вихідним кодом для отримання та перегляду повідомлень syslog (рисунок 1.3).

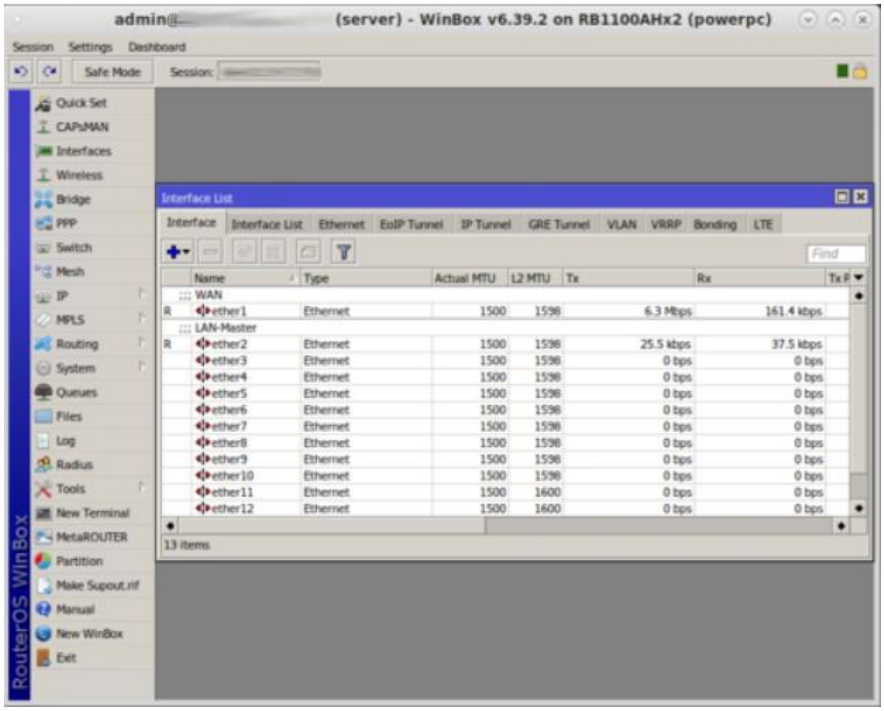


Рисунок 1.2 – Інтерфейс Winbox

Корисна при налаштуванні маршрутизаторів та систем. Вона має режим перегляду повідомлень у реальному часі, перемикання на отримання нових повідомлення, має допоміжне кольорове підсвічування, корисне фільтрування повідомлень та налаштування сповіщень та дії [6].

Безкласова доменна маршрутизація (CIDR) – це метод IP-адресації, що забезпечує гнучке управління IP-простором, уникаючи обмежень класової системи адресації. CIDR дозволяє ефективно використовувати обмежені IP-ресурси, оскільки підмережі можуть мати різні маски, відповідно до потреб мережі[7].

У CIDR IP-адреса і маска підмережі записуються разом, наприклад: 172.20.0.1/16. Число після «/» називається довжиною префікса і показує кількість бітів, встановлених на 1 у масці підмережі, решта бітів встановлюється на 0. Наприклад, префікс /16 відповідає масці 255.255.0.0.

IP pool - це діапазон IP-адрес, який призначений для використання у мережі. Він складається з певної кількості IP-адрес, які можуть бути надані пристроям у мережі для їхньої ідентифікації та комунікації.

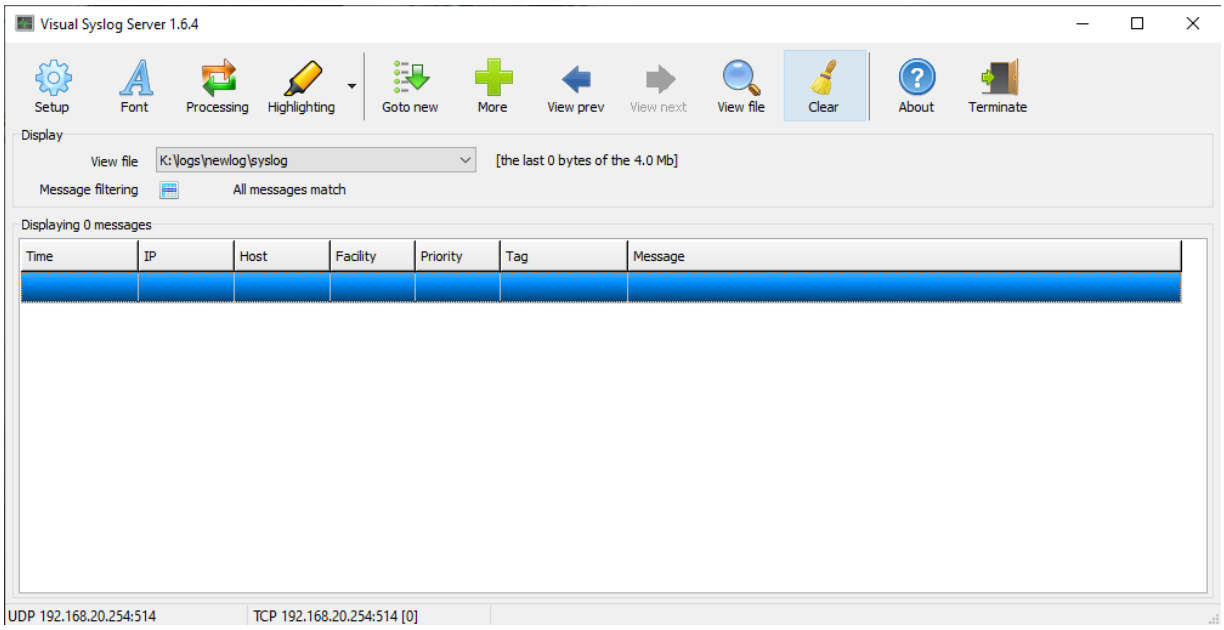


Рисунок 1.3 – Вигляд програми Visual Syslog Server

Брандмауер (Firewall) – це мережевий безпековий пристрій, що контролює весь вхідний та вихідний трафік мережі, приймаючи, блокуючи або перенаправляючи його згідно з встановленими правилами безпеки. По суті, брандмауер діє як бар'єр, що відокремлює приватну внутрішню мережу від загальнодоступного Інтернету [8].

1.2.2 NAT та DHCP

Трансляція мережевих адрес (NAT) – це процес, що дозволяє одній унікальній IP-адресі представляти групу комп'ютерів (рисунок 1.4). При NAT мережевий пристрій, як-от маршрутизатор або брандмауер, призначає публічну IP-адресу комп'ютерам у приватній мережі. Таким чином, NAT дозволяє одному пристрою виступати посередником між локальною приватною мережею та Інтернетом. Основною метою є економія загальнодоступних IP-адрес, підвищуючи безпеку і знижуючи витрати. NAT зберігає IP-адреси, дозволяючи приватним мережам виходити в Інтернет, перетворюючи адреси приватної мережі на дозволені глобальні адреси перед передачею пакетів між мережами. [9].

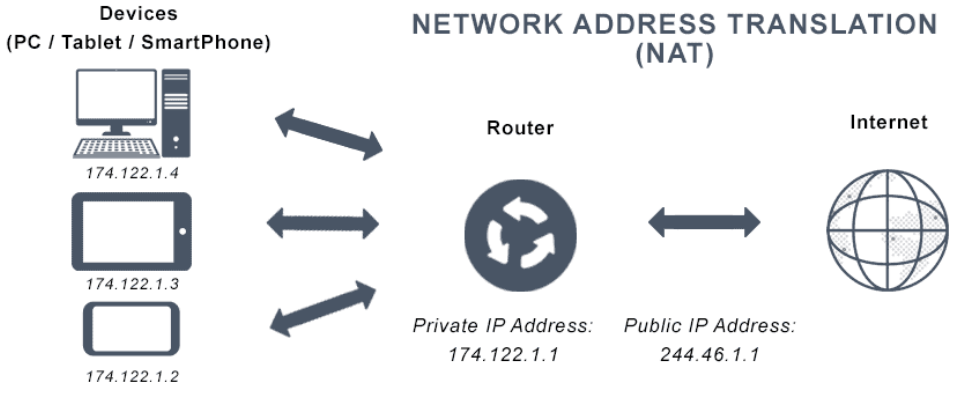


Рисунок 1.4 - Приклад трансляції мережевих адрес (NAT)[9]

NAT зберігає IP-адреси, дозволяючи приватним IP-мережам, виходити в Інтернет. Перш ніж NAT пересилає пакети між мережами, які з'єднує, він перетворює адреси приватної внутрішньої мережі на дозволені глобальні і унікальні адреси.

Протокол динамічної конфігурації хоста (DHCP) – це мережевий протокол, що використовується в IP-мережах для автоматичного призначення IP-адрес та інших мережевих параметрів пристроям, підключеним до мережі за допомогою архітектури клієнт-сервер. [10].

DHCP-клієнт – це Інтернет-хост, який використовує DHCP для отримання параметрів конфігурації, наприклад IP-адреси. Для його налаштування перейдемо у вкладку IP – DHCP Client. Було створено нового клієнта і обрано інтерфейс ether1, який буде отримувати зовнішню динамічну IP-адресу із пулу який ми описали в Address List.

DHCP-сервер є пристроєм у мережі, який призначає IP-адреси клієнтським пристроям із заздалегідь визначеного пулу адрес. Коли новий пристрій підключається до мережі, він надсилає запит на DHCP-сервер для отримання необхідних мережевих налаштувань, включаючи IP-адресу, маску підмережі, адресу шлюзу та серверів DNS. Сервер обробляє запит і забезпечує пристрій потрібними конфігураційними даними для налаштування мережі [11].

1.3 Засоби віддаленого збору логів

Засоби віддаленого збору логів є критично важливими для забезпечення безпеки, моніторингу та аналізу мережевої активності в сучасних IT-системах. Вони дозволяють централізувати збирання, зберігання та аналіз логів з різноманітних джерел, таких як сервери, мережеві пристрої, додатки та інші компоненти інфраструктури. Ці засоби дозволяють підприємствам та організаціям ефективно моніторити свою інфраструктуру, вчасно виявляти та реагувати на

інциденти безпеки, а також проводити глибокий аналіз та оптимізацію роботи своїх систем.

1.3.1 Zabbix

Zabbix – це корпоративне, розподілене рішення для моніторингу, з відкритим кодом, призначене для відстеження продуктивності та доступності мережевих пристроїв, серверів, сервісів та інших IT-ресурсів. Zabbix є гнучким рішенням, яке може моніторити як прості автономні додатки (рисунок 1.5), так і великомасштабні середовища.

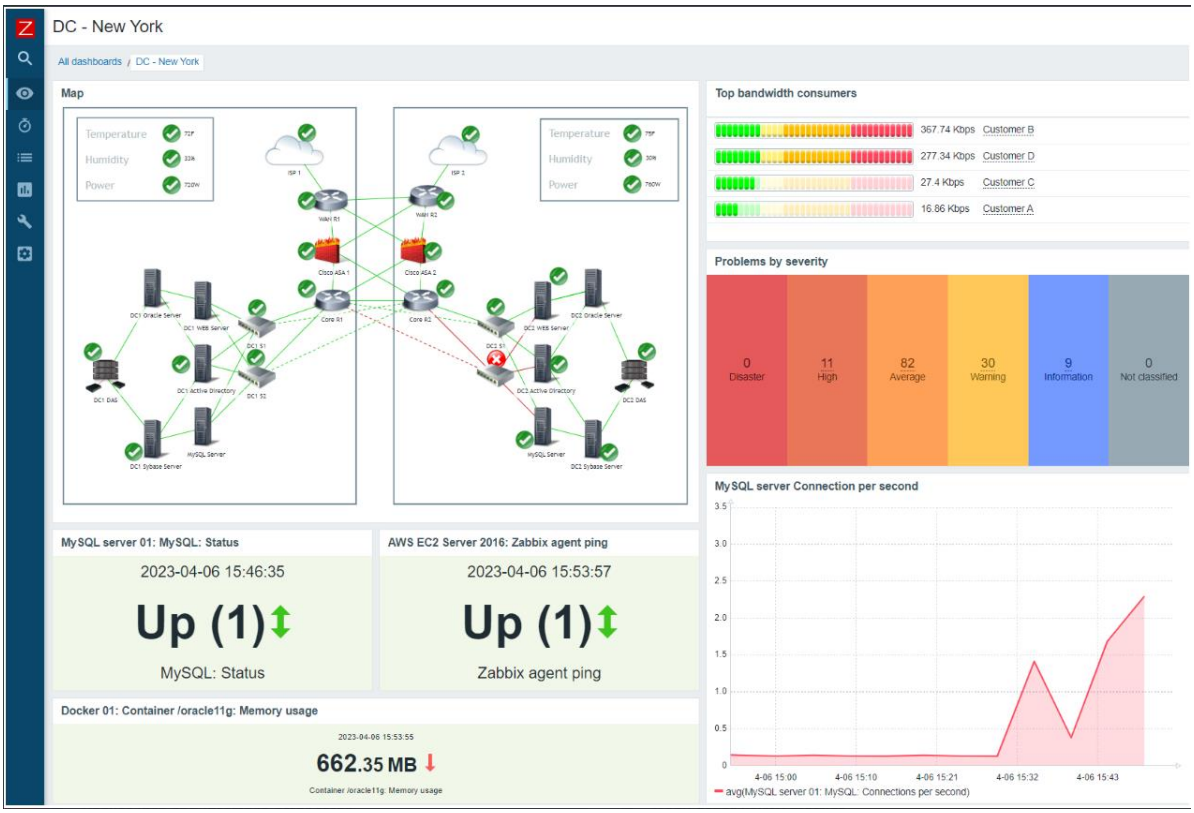


Рисунок 1.5 – Інтерфейс Zabbix [12]

Завдяки функції автоматичного виявлення ресурсів, Zabbix може знаходити мережеві сутності, серверні ресурси та підключати або відключати пристрої.

Використовуючи готові інтеграції (шаблони), можна моніторити будь-що – від найнижчого рівня пристроїв до сервісів SAAS.

Для збору метрик Zabbix використовує агентський або безагентський підхід, що дозволяє збирати дані з будь-яких джерел, включаючи пристрої, датчики, операційні системи, віртуалізаційні платформи, контейнерні платформи, такі як Docker та Kubernetes, хмарні інфраструктури, бази даних, веб-сторінки, Java екосистеми, сервери додатків, кінцеві точки API, бізнес-додатки та багато інших.

Zabbix забезпечує високу продуктивність в реальному часі для виявлення проблем та проведення аналізу першопричин, корелюючи наявні та нові проблеми.

Система також підтримує інциденти, оповіщення та повідомлення: користувачі отримують сповіщення, коли в екосистемі виникає проблема, з використанням різних каналів зв'язку, таких як Microsoft Teams, електронна пошта або текстові повідомлення.

Для візуалізації зібраних даних та моніторингових подій Zabbix надає можливість створювати графіки, списки, географічні карти та топологічні карти мереж.

Завдяки підтримці мультиорендарства та розподіленого моніторингу, одне рішення Zabbix може використовуватися для кількох дата-центрів, відділів та організацій, включаючи віддалені локації за фаєрволами з можливістю віддаленого виконання команд.

Zabbix відрізняється надзвичайною гнучкістю: користувачі можуть адаптувати систему до своїх потреб, використовуючи вбудовані функції, включаючи передачу метрик і подій через HTTP, звітність, аудит, безпеку, обчислення SLA сервісів та багато іншого [13].

Вона підтримує сценарії, засновані на моніторингу, та автоматичне виявлення. Централізований моніторинг лог-файлів здійснюється через веб-інтерфейс, який також використовується для адміністрування та налаштування. Система надає можливості для звітності та аналізу тенденцій, а також підтримує моніторинг відповідності рівня обслуговування (SLA).

Високопродуктивні агенти (zabbix-agent) доступні практично для всіх платформ. Комплексна реакція на події забезпечується підтримкою SNMP v1, 2, 3, SNMP пасток та IPMI. З коробки підтримується моніторинг JXM (Java Management Extensions) додатків, а також виконання запитів до різних баз даних без необхідності використання додаткових скриптів. Розширення можливостей програми здійснюється через виконання зовнішніх скриптів. Система також включає гнучкі шаблони та групи, а також надає можливість створювати карти мереж [14].

1.3.2 Rsyslog

Rsyslog є швидкою системою обробки логів, яка відзначається високою продуктивністю, великим набором функцій безпеки та модульним дизайном (рисунок 1.6). Початково розроблений як звичайний syslogd, rsyslog перетворився на мультифункціональний інструмент у сфері логування, що здатний приймати дані з різноманітних джерел, змінювати їх формат та направляти результати до різних призначень [15].

Rsyslog має ряд передових функцій, таких як фільтрація, і підтримує як TCP, так і UDP для передачі повідомлень. Він може обробляти логи, пов'язані з поштою, авторизацією, повідомленнями ядра та іншими [16].

1.3.3 Splunk

Splunk — це платформа SIEM, призначена для збору, зберігання, аналізу, моніторингу та аналізу інформації. Вона здатна працювати з різноманітними джерелами даних, включаючи віртуальні та фізичні хости, різні пристрої IoT, хмару та CRM-системи (рисунок 1.7).

Рисунок 1.6 – Робота із rsyslog [17]

Продукти Splunk містять штучний інтелект, який обробляє інформацію в режимі реального часу та розбиває отриману інформацію на значення та поля. Весь аналіз і візуалізація інформації здійснюється в автоматичному режимі або за допомогою спеціальних модулів [18].

Splunk - це платформа для великих даних, яка спрощує завдання збору та управління великими обсягами машиногенерованих даних і пошуку інформації в них. Ця технологія використовується для бізнес-аналітики, веб-аналітики, управління додатками, відповідності та безпеки [19].

Splunk також має модульну архітектуру, що дозволяє розширювати його функціональність за допомогою додаткових модулів та додатків. Це робить платформу гнучкою та адаптивною до потреб різних організацій, незалежно від їхнього розміру та галузі.

Завдяки своїм широким можливостям, Splunk використовується в різних сферах, включаючи IT-операції, безпеку, бізнес-аналітику та інтернет речей. Він

допомагає компаніям підвищувати ефективність роботи, покращувати безпеку та приймати обґрунтовані рішення на основі даних.

1.3.4 Wireshark

Wireshark — це безкоштовний інструмент із відкритим кодом, який аналізує мережевий трафік у реальному часі в операційних системах Windows, Mac, Unix і Linux. Він фіксує пакети даних, що проходять через мережевий інтерфейс (наприклад, Ethernet, LAN або SDR), і перетворює ці дані в корисну інформацію для IT-фахівців і команд з кібербезпеки (рисунок 1.8).

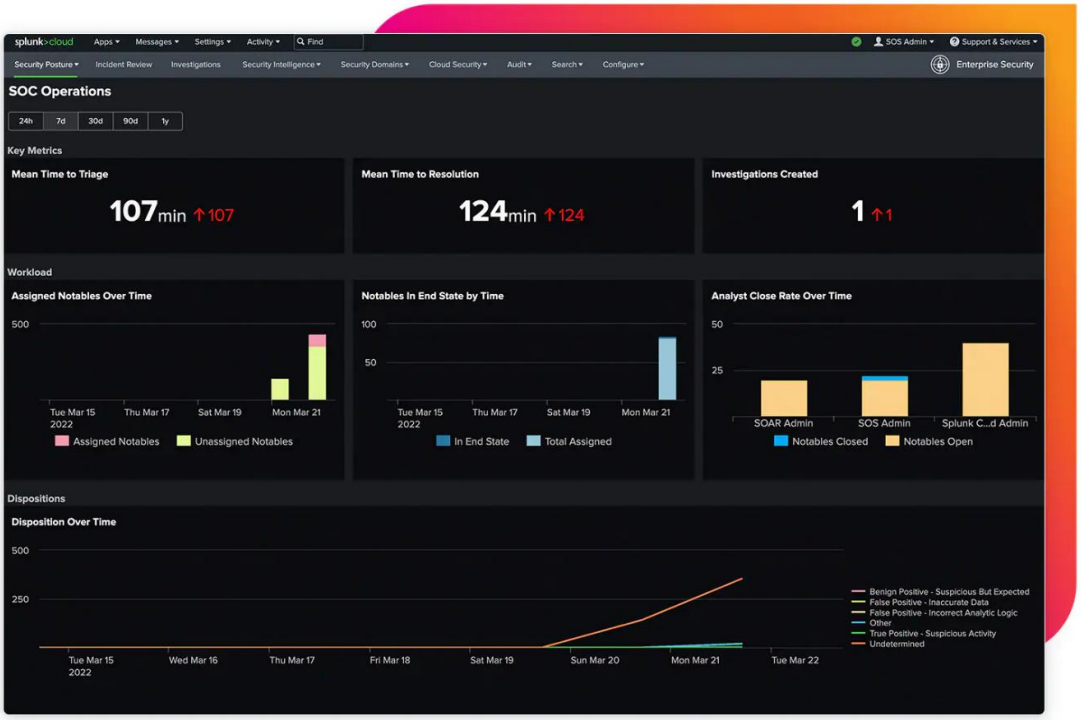


Рисунок 1.7 – Робота із Splunk [20]

Це широко відомий і потужний інструмент для аналізу пакетів, що дає можливість мережевим адміністраторам досліджувати проблеми затримок і ідентифікувати потенційні загрози [22].

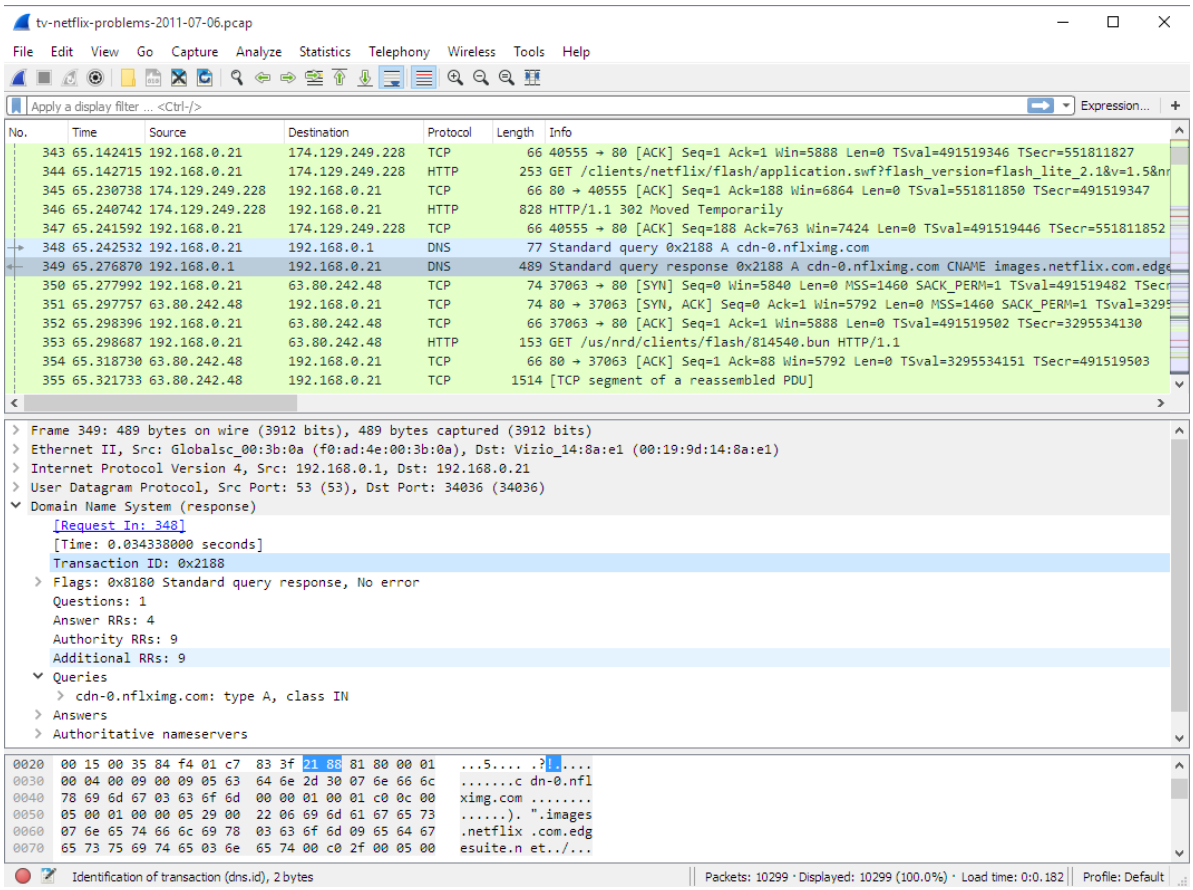


Рисунок 1.8 – Приклад запису пакетів в Wireshark [21]

Wireshark належить до категорії програмного забезпечення захоплення пакетів (також називається аналізатором мережевого протоколу, аналізатором протоколу та аналізатором мережі). Вони перехоплюють мережевий трафік, щоб зрозуміти дії, що обробляються, і зібрати корисну інформацію. Wireshark (раніше відомий як Ethereal) надає низку різних фільтрів відображення для перетворення кожного захопленого пакета в читабельний формат. Це дозволяє користувачам визначити першопричину проблем із кібербезпекою та навіть виявити потенційну діяльність кіберзлочинців.

Сніфер дозволяє IT-фахівцям швидко й ретельно діагностувати безпеку мережі, але якщо Wireshark потрапить у чужі руки, його можна використовувати для кібератак і розвідувальних кампаній [23]. Застосовуючи Wireshark, можна діагностувати та усувати проблеми типових програм, що використовуються в корпоративній мережі. Крім того, можливо вимірювання параметрів мережі,

виявлення мережевих проблем, які вони спричиняють, і ефективне їх вирішення[24].

1.4 Постановка задачі

В даній кваліфікаційній роботі потрібно розробити програмне забезпечення для збору даних про активність користувачів у локальній мережі та передачу інформації у базу даних з можливістю пошуку за певними критеріями та аналізу їх активності

Потрібно вибрати обладнання, обрати програмне забезпечення для збору логів, налаштувати обладнання для забезпечення проходження дозволеного та блокування забороненого трафіку. Розробити базу даних та програмне забезпечення із можливістю збору, перегляду та аналізу активності користувачів та протестувати програмне забезпечення.

2. ПОЕТАПНА РОЗРОБКА ЛОГУВАННЯ МЕРЕЖІ

2.1 Налаштування маршрутизатора Mikrotik

Для налаштування базових налаштувань у Mikrotik знадобилася програма WinBox. Всі подальші дії виконуються у ній.

Після відкриття програми, у переліку була MAC-адреса нашого маршрутизатора до якого було виконано підключення із базовими даними для входу, а саме «admin» у рядку логіну і без паролю. Для запобігання вільного доступу до нашого пристрою, змінено логін та пароль у першу чергу. У цьому допомогло створення нового користувача, надійного паролю та надання йому всіх прав адміністратора і деактивація акаунту адміна.

Далі призначено зовнішню та внутрішню IP-адресу для Mikrotik. Для створення локальної IP-адреси, перейшовши у вкладки IP – Addresses, було введено CIDR, адресу локальної підмережі та інтерфейс. Із списку обрано bridge1 локальної мережі, про його створення описано нижче.

Для зовнішньої мережі проведено ті самі дії, проте вводимо CIDR від провайдера або організації, якщо це має бути сегмент підмережі, і адресу зовнішньої мережі, а із списку інтерфейсів обираємо ether1, який буде вхідним у наш маршрутизатор(рисунок 2.1).

Для того, щоб вказати маршрутизатору DNS-сервер, перейшовши у вкладки IP – DNS, клацнули стрілку вниз навпроти поля Servers та внесли DNS компанії Google, а навпроти динамічного сервера DNS нашої організації. Також поставимо галочку на кнопці «All Remote Requests». Це потрібно для того, щоб комп’ютери в локальній мережі змогли надсилати DNS-запити.

Створення Bridge для локальної мережі. На продуктах MikroTik RouterOS ми маємо можливість створити конфігурацію мосту. Міст служить для об'єднання 2 або більше інтерфейсів в один широкомовний домен або в один сегмент. Мости можуть поєднувати кілька типів інтерфейсів в один сегмент, наприклад бездротові інтерфейси та інтерфейси Ethernet [25].

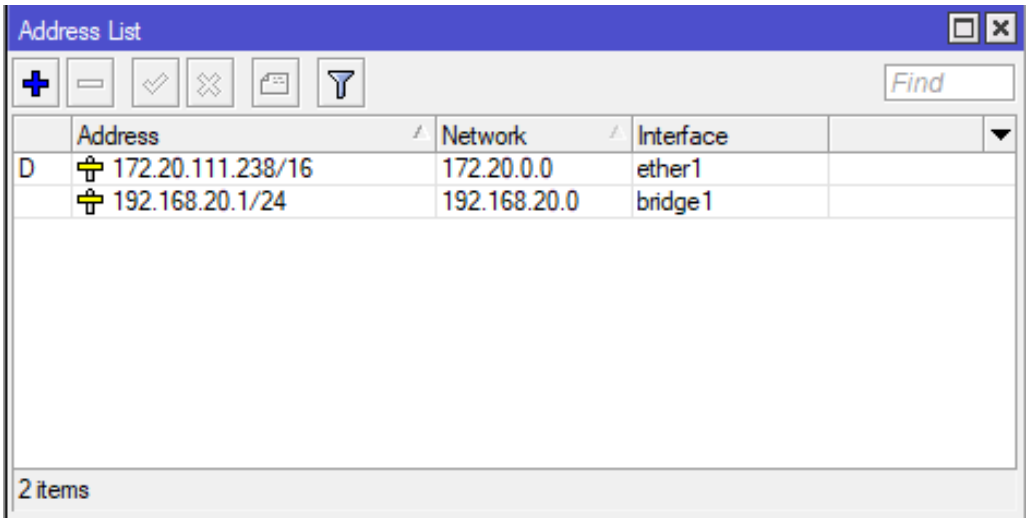


Рисунок 2.1 – Налаштована зовнішня та внутрішня адреси маршрутизатора

Було створено новий Bridge. Після підтвердження та переходу у вкладку Ports, по черзі додано інтерфейси із ether2 по ether 24, тобто всі окрім ether1(рисунок 2.2).

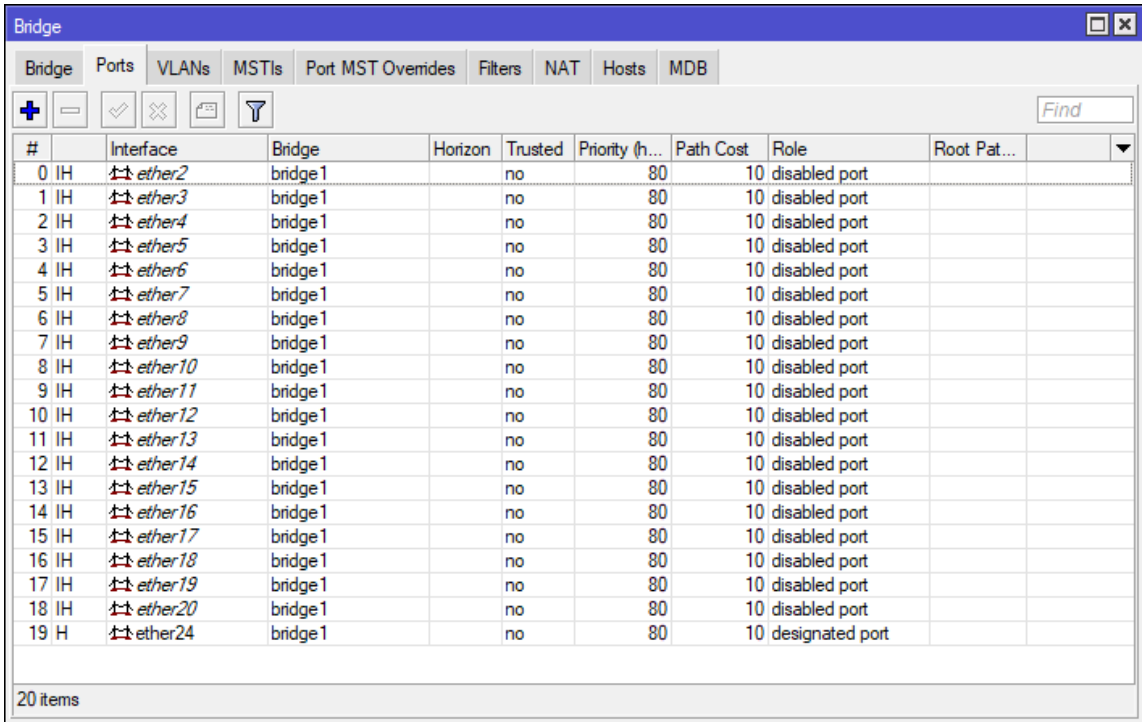


Рисунок 2.2 – Перелік інтерфейсів у bridge1

Створено пул, за яким ми зможемо отримувати локальну IP-адресу від DHCP-сервера. Зайшовши в IP – Pool, дали назву і там було записано локальний пул в діапазоні 192.168.20.2 – 192.168.20.254(рисунок 2.3).

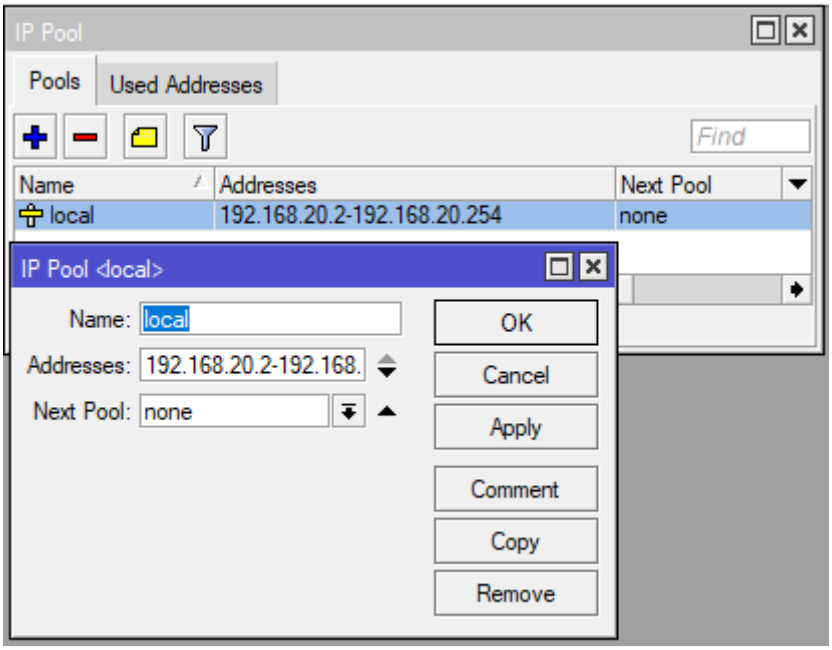


Рисунок 2.3 – Локальний пул адрес маршрутизатора

2.1.1 Firewall

В категорії IP – Firewall – Filter Rules, створено певний правил для брандмауера, які забезпечують безпеку. В першу чергу дозволено отримання ICMP протоколу, що дозволить перевіряти наявність маршрутизатора у мережі та його взаємодію з ним. Цей пункт у нас буде під номером 0 у правилах.

Коротко розпишемо решту пунктів(рисунок 2.4):

а) пропуск трафіку через інтерфейс ether1;

б) правило на логування tcp протоколу на 443 порт у всіх портах, які включені у bridge1;

в) прийом трафіку через інтерфейс ether1;

г) відкидати пакети від всіх адрес, окрім адрес, які є у Enable_address серед Address Lists.

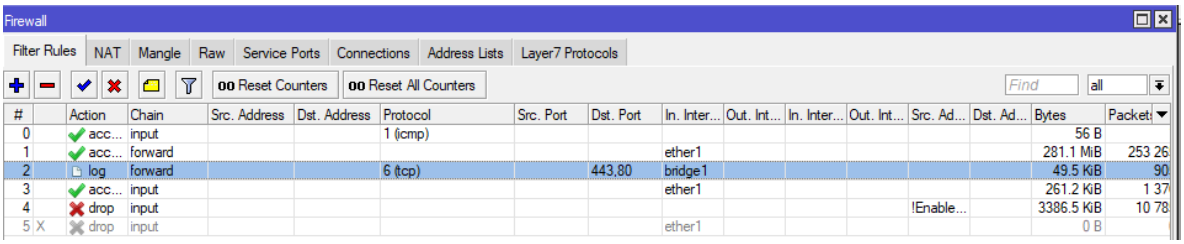


Рисунок 2.4 – Список правил Firewall

Наступним кроком було створення списку доступних адрес, з яких можливий прийом трафіку і підключення(рисунок 2.5).

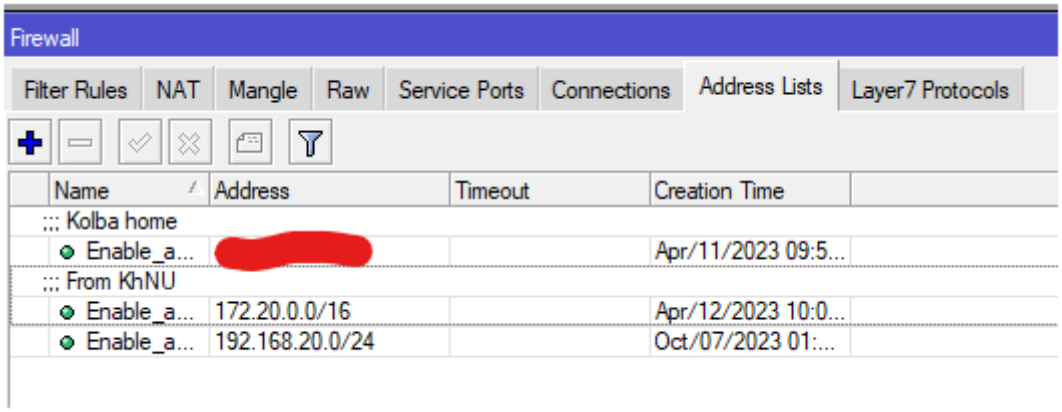


Рисунок 2.5 – Перелік списку доступних адрес в маршрутизаторі

Перша адреса, яка вказана на рисунку 2.5, необхідна для дозволу зовнішньої IP- адреси віддаленого маршрутизатора на підключення до даного пристрою, який ми і налаштували. Наступна підмережа IP-адрес була додана із можливістю забезпечення інтернет-трафіку маршрутизатора з глобальної мережі. Остання підмережа IP-адрес є локальною частиною нашого маршрутизатора.

Налаштування NAT проводились у вкладці IP – Firewall – NAT. У полі Chain обрано srcnat, а у полі Out Interface – ether1. Далі у вкладці Action у полі із цією є назвою обираємо masquerade.

Правило "masquerade" у firewall MikroTik використовується для NAT, зокрема для забезпечення доступу приватних мережних пристроїв до Інтернету через публічну IP-адресу маршрутизатора. Це дозволяє приховати реальні IP-адреси внутрішніх пристроїв і використовувати лише одну публічну адресу для всієї комунікації з Інтернетом. Це забезпечує те, що відповідь на запити буде адресована правильно, тобто до публічної IP-адреси маршрутизатора, яка потім буде перенаправлена до відповідного приватного пристрою в мережі.

2.1.2 DHCP-клієнт і DHCP-сервер

Для завершення налаштування було додано DHCP-клієнт та DHCP-сервер.Для налаштування цих функцій у вкладках IP – DHCP Server було створено сервер. Наступним шляхом було налаштування. Спершу обрано інтерфейс, у нашому випадку це bridge1. Наступним кроком – введено локальну підмережу 192.168.20.0/24. Вказано шлюз – 192.168.20.1. Обирано локальний пул адрес, який вказаний в IP Pool. Введено DNS сервер – 192.168.20.1 та час на який сервер записуватиме у себе видану IP-адресу(рисунок 2.6).

2.1.3 Logging і Visual Syslog Server

Для отримання логів від активності інших користувачів в інтернеті з маршрутизатора на певний сервер, було використано програму Visual Syslog Server, яку можна встановити за посиланням https://sourceforge.net/projects/syslogserverwindows/ .

Початковим етапом налаштування логів буде перехід у вкладки System – Logging – Actions. Після переходу, вибравши поле remote, було обрано тип дій із логами, в нашому випадку remote. Вказана IP-адреса та порт сервера, на який

відправлятимуться записи про активність, в даному випадку IP-адреса 192.168.20.254, а порт – 514. Далі нажавши на галочку навпроти BSD Syslog та в Syslog Facility обрали 22 (local6). Пізніше перейшли в кладку Rules у тій же Logging та створили нове правило із назвою Firewall, таким же коментарем та обраною дією remote (Рисунок 2.7).

Змінили також правило info, щоб події firewall не записувалися у логи маршрутизатора.

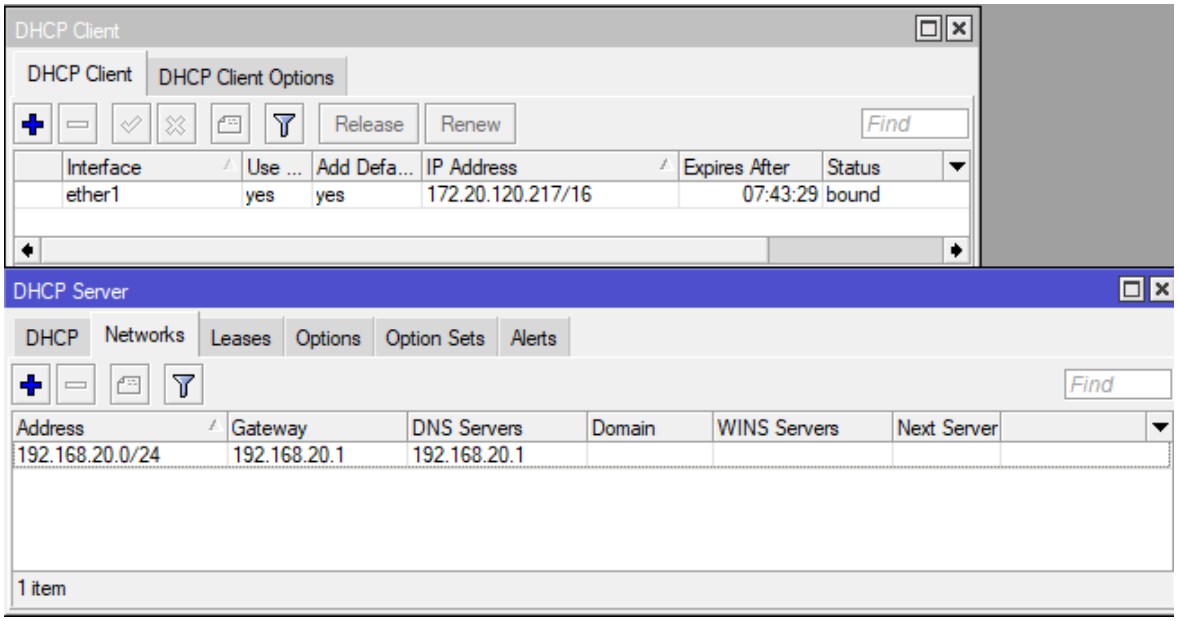


Рисунок 2.6 – Налаштований DHCP Client та DHCP Server

Наступним кроком було створення правила в брандмауері маршрутизатора на логування tcp пакетів, які прямують через порти 80 та 443, через які встановлюються http та https з'єднання, це правило є в наявності на рисунку 4.

Після запуску програми Visual Syslog Server та переходу в налаштування (Setup) у полях TCP/UDP listener interface and port вводимо дані, які ми вказували у дії логування remote, тобто IP-адресу та порт сервера, на якому запущена програма (рисунок 2.8). На вкладці Files ми обрали шлях для збереження логів, задали розмір файлу, за яким буде створено новий файл та їх кількість у даній папці.

Ці логи можуть включають інформацію про вхідні та вихідні з'єднання, заборонені спроби доступу, а також інші події, які стосуються обробки пакетів мережі. Вони можуть містити дані про джерело та призначення, типи пакетів, порти, протоколи, час і дату, а також іншу інформацію, необхідну для аналізу мережевої активності. Аналіз логів брандмауера може також бути корисним для вивчення тенденцій у мережевому трафіку, виявлення неефективних правил брандмауера та вдосконалення стратегій безпеки мережі.

Крім того, ці дані можуть слугувати основою для створення звітів про стан мережевої безпеки, допомагаючи приймати обґрунтовані рішення щодо покращення захисних механізмів. Інформація з логів сприяє не тільки швидкому вирішенню проблем, але й довгостроковому плануванню розвитку мережевої інфраструктури, забезпечуючи її стабільність та надійність.

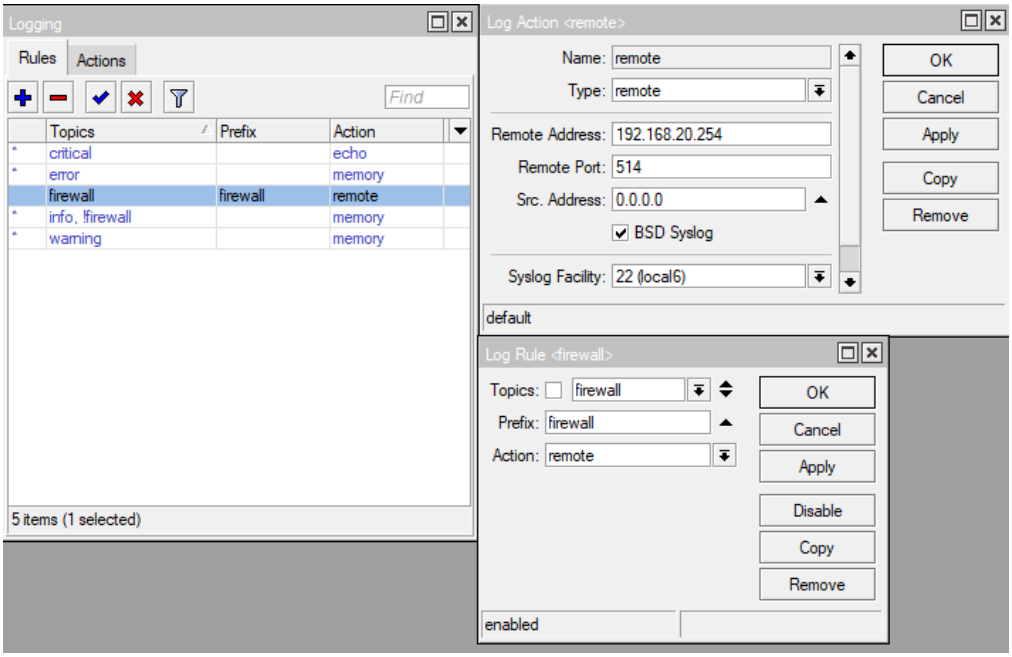


Рисунок 2.7 – Налаштування правил логування

Це особливо актуально в умовах зростаючої кількості кіберзагроз та необхідності впровадження прогресивних стратегій кібербезпеки.

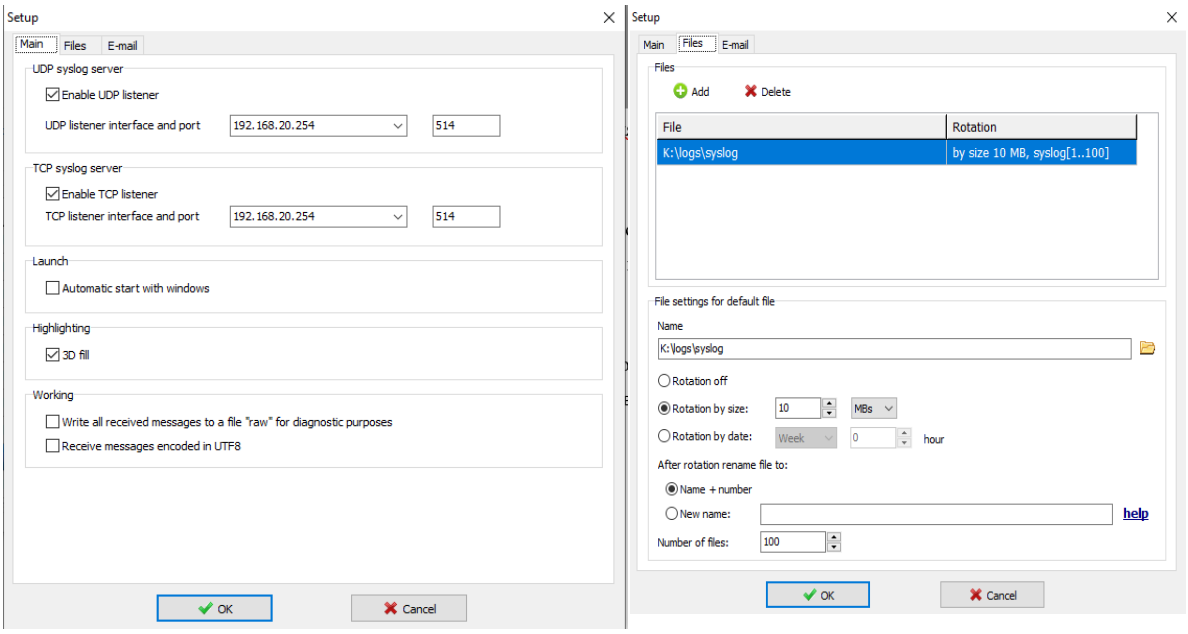


Рисунок 2.8 – Налаштування в програмі Visual Syslog Server

Після цього логи TCP-протоколу, які проходять через брандмауер, відображалися у програмі (рисунок 2.9).

2.2 Проєктування та розробка бази даних

База даних у даній кваліфікаційній роботі створена за допомогою раніше згаданого Microsoft SQL Server Management Studio 19 (рисунок 2.10).

Зазвичай при розробці використовують такі моделі БД, як:

– інфологічна модель;

– даталогічна модель;

– фізична модель.

Інфологічні моделі даних використовуються на ранніх стадіях проектування для опису структур даних у процесі розробки додатка.

Даталогічна модель є моделлю логічного рівня, яка відображає логічні зв’язки між елементами даних незважаючи на їхній зміст та середовище зберігання

[26].

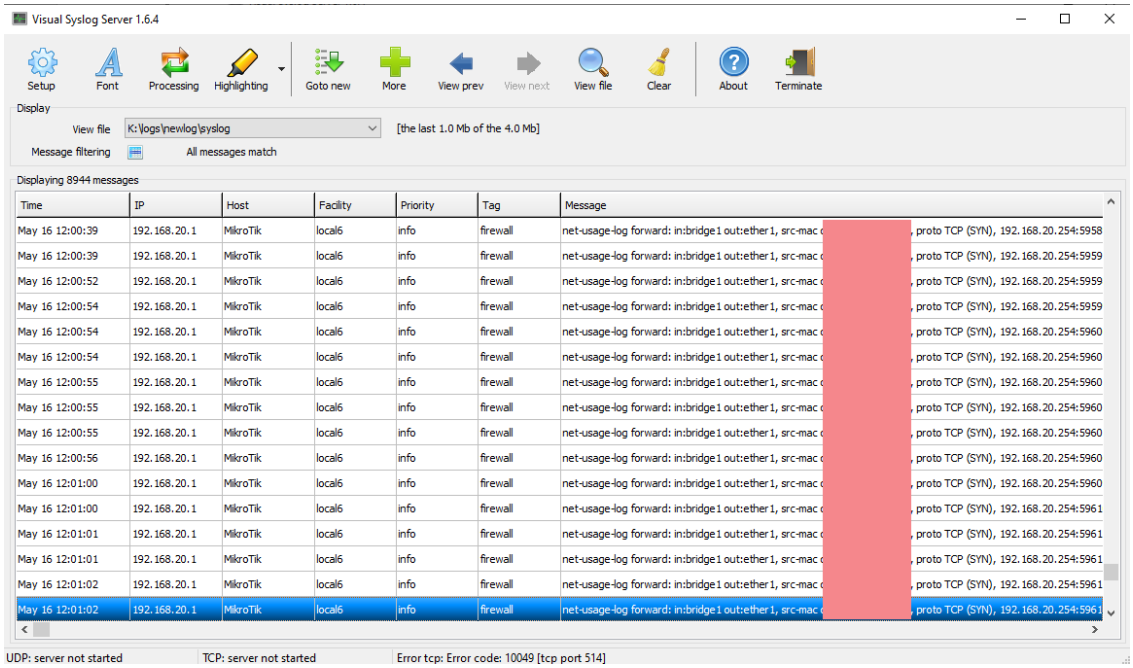


Рисунок 2.9 – Відображення логів брандмауера, які проходять через маршрутизатор у програмі Visual Syslog Server

Фізична модель – це подання дизайну даних як реалізовано у системі керування базами даних.

За структурою організації в БД розрізняють такі моделі баз даних:

– реляційна;

– ієрархічна;

– мережна.

Ієрархічна модель – зберігає інформацію послідовно: один елемент є головним, а інші йому підпорядковуються.

Реляційна модель – логічна модель даних, яка складається з сукупності взаємопов’язаних двохвимірних таблиць.

Мережна модель – має складові структури, як і реляційна, але відрізняється характером відношення між ними так, як існує можливість встановлення додаткових зв’язків [27].

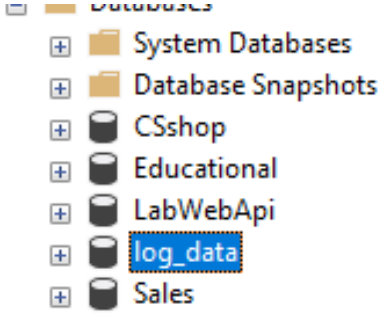


Рисунок 2.10 – Створена база даних log_data

При отриманні лог файлів з маршрутизатора, ми будемо записувати у базу даних наступні критерії з отримуваного повідомлення:

– IP-адреса маршрутизатора;

– назва маршрутизатора;

– дата та час повідомлення;

– вид локального повідомлення;

– тип повідомлення;

– джерело повідомлення;

– коментар;

– правило;

– вхідний інтерфейс;

– вихідний інтерфейс;

– MAC-адрес джерела;

– протокол;

– IP-адреса та порт відправника;

– IP-адреса та порт отримувача;

– довжина пакета.

Створено три таблиці: router (рисунок 2.11), message (рисунок 2.12) та users (рисунок 2.13).

На рисунку 2.11 видно, що id_router – це первинний ключ та ідентифікатор

роутера у базі. Ip_router – IP адреса маршрутизатора з якого зчитуються логи. name_router – назва маршрутизатора. out_Interface – вихідний інтерфейс через який здійснюється прийом даних у мережу.

```
CREATE TABLE router (
    id_router INT PRIMARY KEY IDENTITY(1,1),
    ip_router VARCHAR(50) NOT NULL,
    name_router VARCHAR(100),
    out_Interface VARCHAR(50)
);
```

Рисунок 2.11 – Створена таблиця router

У таблиці massage (рисунок 2.12) id_massag – як і id_router первинний ключ, проте це вже ідентифікатор повідомлення. Time_massag – дата та час отримання повідомлення. Tipe_local_messag – вид локального повідомлення. Tipe_messag – тип повідомлення. Source_messag – джерело повідомлення. protocol_messag – протокол. Length_messag – довжина повідомлення.

```
CREATE TABLE message (
    id_messag INT PRIMARY KEY IDENTITY(1,1),
    time_massag DATETIME NOT NULL,
    tipe_local_messag VARCHAR(50),
    tipe_messag VARCHAR(50),
    source_messag VARCHAR(50),
    protocol_messag VARCHAR(50),
    length_messag INT
);
```

Рисунок 2.12 – Створена таблиця message

Наступним створено таблицю users (рисунок 2.13). Id_List – первинний ключ та ідентифікатор користувача. Id_router – зовнішній ключ, який посилається на дані id_router, що у таблиці router. Id_messag – має таке саме значення, що і id_router, проте воно посилається на id_messag, що у таблиці message. In_interface – вихідний інтерфейс з якого йде трафік від користувачів. Comment – коментар. Mac_source – MAC-адреса джерела/користувача трафіку. Ip_and_port_sender – IP-адреса та порт

відправника, тобто користувача. Ip_and_port_receiver - IP-адреса та порт отримувача, тобто кому були призначені пакети.

```
CREATE TABLE users (
    id_List INT PRIMARY KEY IDENTITY(1,1),
    id_router INT FOREIGN KEY REFERENCES router(id_router),
    id_messag INT FOREIGN KEY REFERENCES message(id_messag),
    in_interface VARCHAR(50),
    comment VARCHAR(50),
    mac_source VARCHAR(50),
    ip_and_port_sender VARCHAR(50),
    ip_and_port_receiver VARCHAR(50)
);
```

Рисунок 2.13 – Таблиця users

Ці таблиці пов’язані, тому можна створити фізичну (рисунок 2.14) та даталогічну (рисунок 2.15) моделі бази даних.

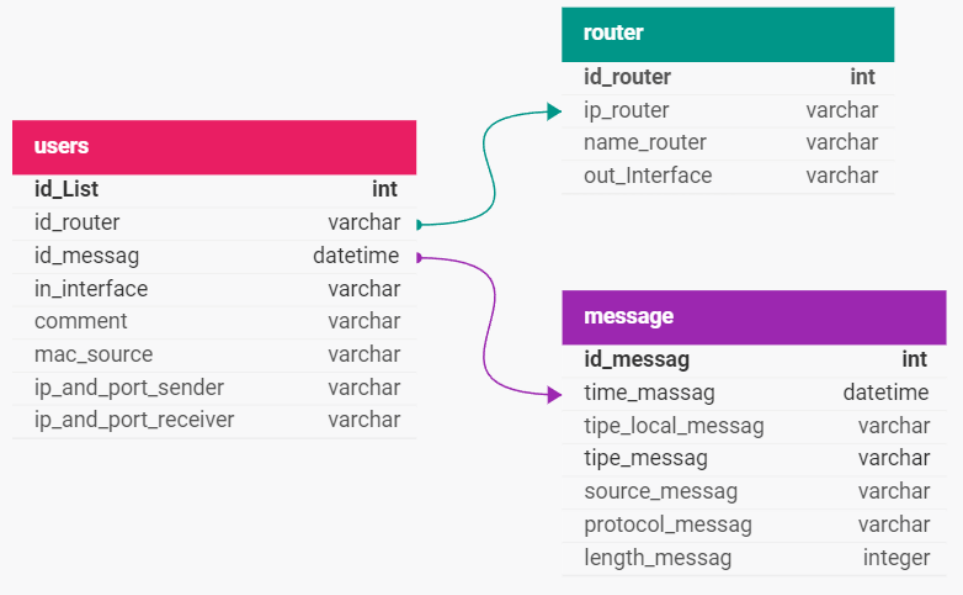


Рисунок 2.14 – Фізична модель бази даних

Фізична модель бази даних визначає спосіб, яким дані будуть зберігатися та організовані на рівні операційної системи та апаратного забезпечення. Вона встановлює структуру таблиць, індексів, відносин між ними, а також методи зберігання та доступу до даних. Фізична модель також включає визначення схеми

розподілу даних, вибір форматів файлів, визначення стратегій збереження даних на носіях та оптимізації доступу до них.

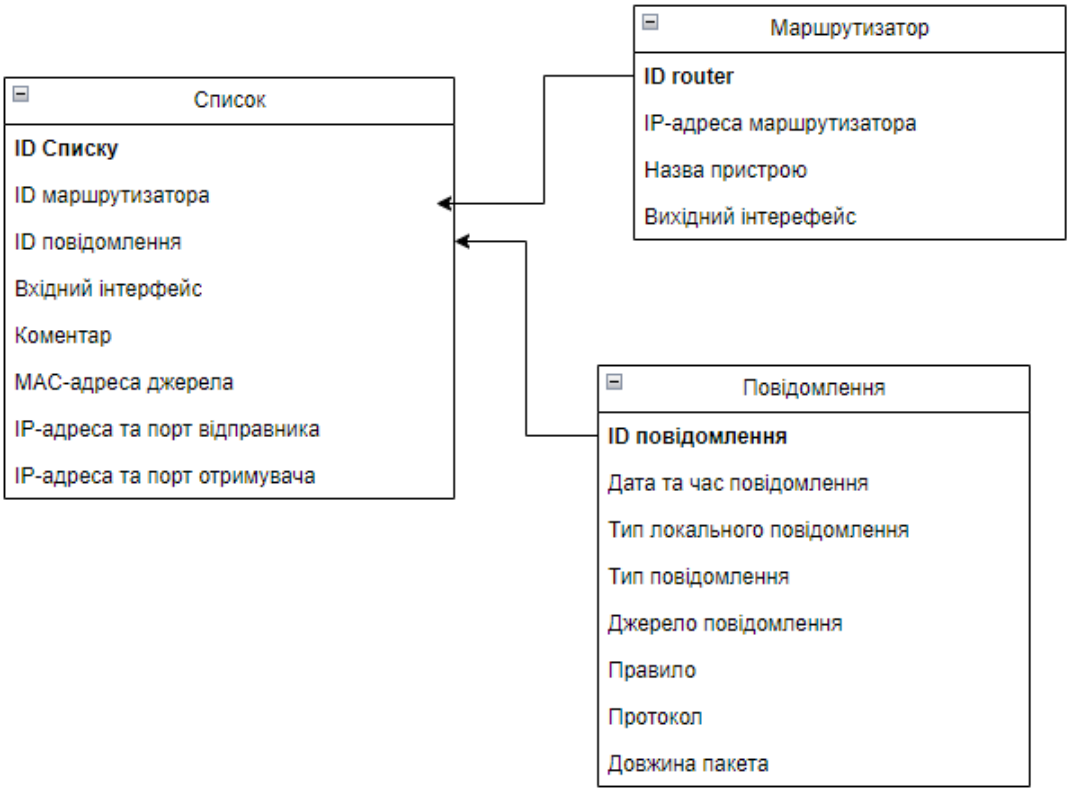


Рисунок 2.14 – Даталогічна модель бази даних

2.3 Проєктування та розробка програмного забезпечення

2.3.1 Вибір мови програмування

В даній кваліфікаційній роботі розробка програмного забезпечення виконувалася на мові Python.

Мова програмування Python розпочала свій шлях з версії 0.9.0 у лютому 1991 року. Вже на той момент вона мала можливість обробки винятків, функцій та базових типів даних, а також володіла механізмом обміну функціональністю між класами та іншими функціями. З цього моменту почалося її постійний розвиток.

Версія 1.0 була випущена у січні 1994 року. Також у тому ж році з'явився comp.lang.python, що став важливим форумом для дискусій та співпраці в спільноті користувачів Python. Версія 1.2 була останньою, що вийшла під керівництвом Центру математики та інформатики в Амстердамі(CWI). Починаючи з 1995 року, розробку мови продовжила Корпорація національних науково-дослідних ініціатив (CNRI).

У CNRI було видано кілька версій мови, включаючи 1.6. Від неї почалося формування вимог користувачів щодо створення версії Python, сумісної з ліцензією BPL. Це привело до випуску версії 1.6.1, яка містила кілька дрібних змін та нову ліцензію, що дозволила подальшому розвитку мови.

У 2000 році команда Гвідо ван Россума перейшла до BeOpen.com, де було створено BeOpen PythonLabs. Це призвело до випуску версії 2.0. Незабаром після цього Гвідо разом зі своєю командою перейшов до Digital Creations. Після випуску версії 2.1 всі права на інтелектуальну власність були передані в Python Software Foundation (PSF) [28].

Python значною мірою зосереджений на читабельності, узгодженості та якості. Однорідність мови забезпечує високу читабельність, що є надзвичайно важливим в сучасних умовах, коли програмування більше схоже на колективну працю, ніж на індивідуальну діяльність [29]. Ця мова програмування має багато переваг. До прикладу, вона має чистий і зрозумілий синтаксис, що робить її легкою мовою для навчання навіть для новачків. Це дозволяє швидко приступити до написання коду та зменшує час, необхідний для навчання. Завдяки своїй читабельності, Python сприяє написанню чистого та зрозумілого коду. Це полегшує підтримку та розширення коду в майбутньому. Працює на багатьох платформах, включаючи Windows, macOS та Linux, що робить його універсальним вибором для розробки.

Є ще декілька причин, чому ми обрали цю мову програмування. Python спрощує розробку GUI завдяки своїм зрозумілим бібліотекам та модулям. Він надає потужні інструменти для роботи з текстовими даними, такі як регулярні

вирази (модуль re) і потужні засоби для обробки дат та часу (модуль datetime), що є важливим для нашої задачі з парсингом логів. Python має чудову підтримку для роботи з різними базами даних через модулі, такі як «pyodbc» для підключення до SQL Server, що ми використовували у нашій програмі.

2.3.2 Вибір основних програмних бібліотек

Вбудована бібліотека Python поставляється з модулем logging, який містить багато корисних функцій порівняно з бібліотеками журналювання за замовчуванням в інших мовах програмування. Він популярний серед розробників, добре задокументований, його функціональність можна розширити за допомогою сторонніх модулів. Нижче наведено деякі функції, які він пропонує: форматування записів журналу; надсилання журналів до кількох місць призначення, починаючи від стандартного виводу, файлів, електронних листів, сокетів і HTTP; складна фільтрація; визначення власних рівнів; можна розширити за допомогою інших модулів для підтримки структурованого журналювання, журналювання красивого друку [30].

Наступною вбдованою бібліотекою є модуль tkinter – це стандартна бібліотека, яка використовується для створення графічного інтерфейсу користувача комп’ютерних програм. Бібліотека tkinter надає нам багато вбудованих графічних компонентів інтерфейсу (їх також називають “віджетами”), які можна використовувати для створення програм для різних операційних систем [31].

Серед її переваг: простота та легкість у використанні; незалежність від платформи; розширюваність; має велику кількість налаштовуваних компонентів, таких як кнопки, надписи, текстові поля.

Наступна бібліотека – re. Регулярні вирази, часто позначені як RE, встановлюють набір шаблонів для визначення рядків, які відповідають певним

критеріям; функції у цьому модулі дозволяють перевіряти, чи співпадає певний рядок з вказаним регулярним виразом (або, навпаки, чи відповідає вказаний регулярний вираз певному рядку) [32].

Datetime – надає класи для обробки часу і дати різними способами, підтримується і стандартний спосіб представлення часу, проте більший акцент зроблено на простоту маніпулювання датою, часом і їх частинами. [33]

За допомогою модулю Python з відкритим кодом pyodbc, який реалізує специфікацію Python DB API 2.0, ми можемо підключити Python до бази даних ODBC. Ця специфікація розроблена, щоб забезпечити узгоджений інтерфейс для різних баз даних і допомогти розробникам писати програми, які можуть працювати з різними базами даних без істотних змін коду. Модуль ODBC Python дозволяє логічно організувати код, що полегшує його розуміння та використання. Також він дозволяє підключатися до джерел даних з Python у Windows, macOS і Linux як для 32-бітних, так і для 64-бітних платформ.

Щоб встановити модуль pyodbc на свій комп’ютер потрібно ввести команду pip install pyodbc в інтерактивному режимі Python. Щоб під’єднатися до бази даних і отримати дані за допомогою мови програмування Python, потрібно лише кілька функцій, а саме:

- connect() – для створення підключення до бази даних;
- cursor() – для створення курсору з підключення;
- execute() – для виконання оператора select;
- fetchone() – для отримання рядків із запиту [34].

Matplotlib – це бібліотека для створення графіків у Python та її застосування в числовому обчисленні з NumPy. Вона має об'єктно-орієнтований інтерфейс програмування додатків (API), що дозволяє вбудовувати графіки в програми за допомогою різноманітних бібліотек інтерфейсу користувача, таких як Tkinter, wxPython, Qt або GTK [35].

2.3.3 Структура програмного забезпечення

Щоб розбити цей код на більш зрозумілі і керовані компоненти, було створено структуру проекту з декількома файлами(рисунок 2.15). Це дозволить розділити логіку парсингу логів, взаємодію з базою даних та інтерфейс користувача.

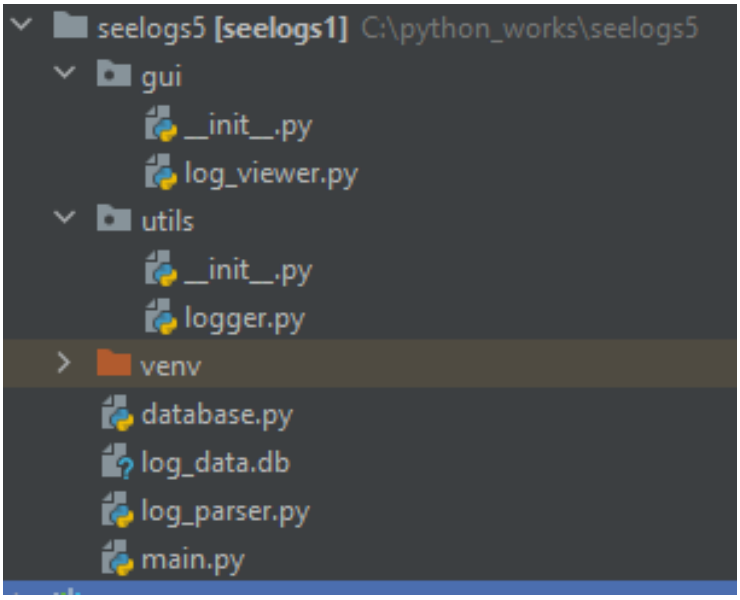


Рисунок 2.15 – Структура проєкту

__init__.py – файли зустрічаються двічі, оскільки вони потрібні для позначення каталогів gui та utils як пакетів у Python. Це дозволяє імпортувати модулі з цих каталогів, використовуючи синтаксис імпорту пакетів.

Gui/log_viewer – містить клас LogViewerApp, який відповідає за GUI програми.

utils/logger.py – файл для налаштування логування.

Database.py – має в наявності функції для взаємодії з базою даних: збереження і отримання даних.

Log_parser.py – володіє класом LogParser, який відповідає за парсинг логів.

Main.py – Головний файл, який ініціалізує програму і створює головне вікно.

2.4 Код програмного забезпечення

2.4.1 Файл log_parser.py

На початку було імпортовано модулі «re», «datetime», «logging» відповідно для роботи з регулярними виразами, датою та часом, а також логуванням.

Рядок коду «logger = logging.getLogger(__name__)» створює об'єкт логера для поточного модуля, використовуючи ім'я модуля (__name__).

Надалі буде надаватись головні частини коду і короткий опис під ним.

```
class LogParser:
    def __init__(self, filepath=None):
        self.filepath = filepath
        self.parsed_data = []
```

LogParser – це клас для парсингу лог-файлів. Він використовує регулярні вирази для виділення необхідної інформації з кожного рядка логу і зберігає розпарсені дані у списку. Логування використовується для відслідковування процесу парсингу і повідомлення про успіхи та помилки.

__init__(self, filepath=None) – це конструктор класу LogParser. Він використовується для ініціалізації об'єкта класу.

Його функції:

– приймає необов'язковий параметр «filepath», який визначає шлях до файлу з логами;

– ініціалізує атрибут «self.filepath» значенням «filepath», як порожній список, який буде використовуватися для зберігання розпарсених даних.

```
def parse_log(self, filepath=None):
        self.filepath = filepath or self.filepath
        if not self.filepath:
            raise ValueError("No file path provided for parsing.")

        current_year = datetime.now().year
        with open(self.filepath, 'r') as file:
            for line in file:
```

```
                parsed_line = self._parse_line(line, current_year)
                if parsed_line:
                    self.parsed_data.append(parsed_line)
                    logger.debug(f"Parsed line: {parsed_line}")
                else:
                    logger.warning(f"Failed to parse line: {line}")
```

«parse_log(self, filepath=None)» – це основний метод для парсингу лог-файлу. Його функції:

– приймає необов'язковий параметр «filepath», який, якщо передано, оновлює атрибут «self.filepath».

– перевіряє, чи правильно задано шлях до файлу, інакше викидає «ValueError».

– відкриває файл за вказаним шляхом і читає його порядково.

– для кожного рядка викликає метод «_parse_line» для парсингу і додає успішно розпарсені дані до «self.parsed_data».

– логує інформацію про успішний парсинг або попередження у випадку невдачі.

```
def _parse_line(self, line, current_year):
    pattern = (r'(?P<router_ip>\d+\.\d+\.\d+\.\d+)\t'
               r'(?P<date>\w+ \d+ \d+:\d+:\d+)\t'
               r'MikroTik\t(?P<tipe_local_messag>\w+)\t'
               r'(?P<tipe_messag>\w+)\t'
               r'(?P<source_messag>\w+)\t'
               r'(?P<comment>net-usage-log) forward: in:(?P<in_iface>[^ ]+)
out:(?P<out_iface>[^,]+), '
               r'src-mac (?P<src_mac>[^,]+), proto (?P<protocol_messag>[^ ]+)
\([^)]+\), '
               r'(?P<src_ip>\d+\.\d+\.\d+\.\d+):(?P<src_port>\d+)-
>(?P<dst_ip>\d+\.\d+\.\d+\.\d+):(?P<dst_port>\d+), len
(?P<length_messag>\d+)')
    match = re.search(pattern, line)
    if match:
        data = match.groupdict()
        data['time_massag'] = datetime.strptime(f"{current_year}
{data['date']}", '%Y %b %d %H:%M:%S')
        return data
    else:
        return None
```

«_parse_line(self, line, current_year)» – приватний метод для парсингу окремого рядка лог-файлу.

Його функції:

– приймає рядок логу та поточний рік як параметри;

– визначає регулярний вираз для парсингу рядка;

– застосовує регулярний вираз до рядка для виділення необхідних частин (IP-адреси, дата, тип повідомлення тощо) і якщо він співпав з рядком, перетворює виділені частини в словник, а якщо ні – повертає None;

– форматує дату і час, додаючи поточний рік;

– повертає розпарсені дані у вигляді словника.

Взагальному, конструктор «__init__» створює об'єкт парсера і готує його для роботи, метод «parse_log» відповідає за основний процес читання та парсингу лог-файлу, обробляючи кожен рядок окремо, а приватний метод _parse_line виконує детальний парсинг кожного рядка, використовуючи регулярні вирази для виділення потрібної інформації.

Таким чином, ці методи працюють разом для забезпечення ефективного парсингу лог-файлів, де кожен з них виконує чітко визначену роль у цьому процесі.

2.4.2 Файл logger.py

```
import logging

def setup_logging():
    logging.basicConfig(level=logging.DEBUG,
                        format='%(asctime)s - %(name)s - %(levelname)s -
%(message)s',
                        datefmt='%Y-%m-%d %H:%M:%S')
```

Імпорт модуля «logging» додає можливість створювати і обробляти логи в додатку.

Функція «setup_logging()» використовується для налаштування основних параметрів логування.

Виклик «logging.basicConfig()» встановлює рівень логування як DEBUG, визначає формат повідомлень у логах та формат часу.

«level=logging.DEBUG» визначає рівень серйозності логів, які будуть оброблятися. Рівень DEBUG означає, що будуть записуватися всі повідомлення, включаючи налагоджувальні, інформаційні, попередження, помилки та критичні повідомлення. Це забезпечує максимально детальну інформацію для розробників під час налагодження коду.

Це налаштування дозволяє програмі вести детальний журнал подій, що дуже корисно для відстеження роботи додатку та пошуку помилок під час розробки і тестування.

2.4.3 Файл database.py

Початком є імпорт бібліотек pyodbc та logging для для підключення до бази даних через ODBC та введення логів відповідно.

Нижче продемонстровані частини коду із загальним його значенням.

```
def save_to_database(parsed_data):
conn = pyodbc.connect('DRIVER={SQL
Server};SERVER=NOUTTUF;DATABASE=log_data;Trusted_Connection=yes;')
c = conn.cursor()
```

Ця частина коду встановлює з'єднання з базою даних та створює курсор для виконання SQL-запитів. Функція «save_to_database(parsed_data)» зберігає оброблені дані у базу даних.

```
c.execute('''SELECT 1 FROM users ... WHERE ...''', (...))
if c.fetchone():
    logger.debug("Duplicate entry found, skipping insert.")
    continue
```

Перевіряє, чи існує вже такий запис у базі даних. Якщо існує, пропускає вставку.

```
c.execute('INSERT INTO router ... VALUES ...', (...))
id_router = c.execute('SELECT @@IDENTITY').fetchone()[0]

c.execute('INSERT INTO message ... VALUES ...', (...))
id_messag = c.execute('SELECT @@IDENTITY').fetchone()[0]

c.execute('INSERT INTO users ... VALUES ...', (...))

conn.commit()
conn.close()
```

Вставляє нові записи у відповідні таблиці (router, message, users) та отримує їх ідентифікатори для зв'язку між таблицями. Останні два рядки фіксують всі зміни у базі даних та закривають з’єднання.

```
def fetch_from_database():
conn = pyodbc.connect('DRIVER={SQL
Server};SERVER=NOUTTUF;DATABASE=log_data;Trusted_Connection=yes;')
c = conn.cursor()
c.execute('SELECT ... FROM users ... JOIN ... ORDER BY u.id_List ASC')
data = c.fetchall()
conn.close()
```

Функція «fetch_from_database()» отримує інформацію з бази даних. Код, що нижче від функції, виконує запит для отримання даних з об'єднаних таблиць та закриває з'єднання.

```
return [{'id_List': row[0], 'router_ip': row[1], 'time_massag': row[2],
'in_iface': row[3], 'comment': row[4],
         'src_mac': row[5], 'src_ip': row[6].split(':')[0], 'src_port':
row[6].split(':')[1],
         'dst_ip': row[7].split(':')[0], 'dst_port': row[7].split(':')
[1]} for row in data]
```

Цей код формує результат, перетворює отримані дані у список словників для зручного доступу.

Підсумовуючи цей файл, функція «save_to_database» перевіряє та зберігає оброблені дані у базу даних, уникаючи дублікатів, а «fetch_from_database» отримує дані з бази даних та формує їх у зручний формат.

2.4.3 Файл log_viewer.py

Наступний код створює додаток для перегляду логів, який дозволяє користувачам переглядати, фільтрувати, сортувати та додавати логи до бази даних, а також відображати активність IP-адрес на графіку.

```
import tkinter as tk
from tkinter import ttk, filedialog
import matplotlib.pyplot as plt
from matplotlib.dates import DateFormatter
import pyodbc
from log_parser import LogParser
from database import save_to_database, fetch_from_database
import logging
```

Цим кодом імпортували необхідні бібліотеки для створення графічного інтерфейсу, роботи з графіками, базою даних, парсингом логів та ведення логів.

```
class LogViewerApp:
    def __init__(self, master, parsed_data):
        self.master = master
        self.parsed_data = parsed_data
        self.filtered_data = parsed_data
        self.master.title("Log Viewer")
        self.tree = ttk.Treeview(master, show="headings")
        self.scrollbar = ttk.Scrollbar(master, orient="vertical",
command=self.tree.yview)
        self.tree.configure(yscrollcommand=self.scrollbar.set)
        self.sort_column = None
        self.sort_reverse = False
        self.search_field = tk.StringVar()
        self.search_value = tk.StringVar()
        self._setup_widgets()
        self._build_tree()
```

«__init__» – конструктор класу. Він ініціалізує головні змінні, налаштовує заголовок вікна, створює основні віджети (дерево Treeview для відображення логів

та вертикальну прокрутку), а також викликає методи для налаштування віджетів та побудови дерева.

Функція «def _setup_widgets(self)» створює та розміщує всі необхідні елементи керування (пошукове поле, кнопки пошуку та очищення, кнопка для додавання в базу даних) на графічному інтерфейсі.

```
def _build_tree(self):
    self.tree['columns'] = (
        'Номер запису', 'IP маршрутизатора', 'Дата та час', 'Інтерфейс', 'Коментарій', 'MAC джерела',
        'IP та Port джерела', 'IP та Port отримувача')
    for col in self.tree['columns']:
        col_text = col.replace(':', ' & ').title()
        self.tree.heading(col, text=col_text,
                          command=lambda _col=col: self._sort_by_column(_col, not self.sort_reverse))
        self.tree.column(col, width=100, anchor='center', stretch=True)
    self._update_treeview()
```

Функція «_build_tree» налаштовує структуру відображення даних в таблиці. Він визначає стовпці, форматує заголовки, встановлює команди сортування та налаштовує параметри стовпців (ширину, вирівнювання). Після налаштування, функція викликає «_update_treeview» для заповнення дерева даними.

```
def _update_treeview(self):
    for row in self.tree.get_children():
        self.tree.delete(row)
    for item in self.filtered_data:
        self.tree.insert('', 'end', values=(
            item.get('id_List'), item.get('router_ip'), item.get('time_massag'), item.get('in_iface'),
            item.get('comment'), item.get('src_mac'), item.get('src_ip') + ':' + item.get('src_port'),
            item.get('dst_ip') + ':' + item.get('dst_port')))
```

Функція «_update_treeview» відповідає за оновлення вмісту дерева (таблиці) з логами у графічному інтерфейсі користувача. Вона очищає поточний вміст дерева і заповнює його новими даними, які містяться у властивості «filtered_data.».

```
def _sort_by_column(self, col, reverse):
    col_index = self.tree['columns'].index(col)
    if col == 'IP та Port джерела':
```

```
        self.filtered_data.sort(key=lambda x: (self._ip_to_tuple(x['src_ip']),
int(x['src_port'])), reverse=reverse)
    elif col == 'IP та Port отримувача':
        self.filtered_data.sort(key=lambda x: (self._ip_to_tuple(x['dst_ip']),
int(x['dst_port'])), reverse=reverse)
    else:
        self.filtered_data.sort(key=lambda x:
x[self._get_key_from_column(col_index)], reverse=reverse)
    self.sort_column = col
    self.sort_reverse = reverse
    self._update_treeview()
    self._update_headings()
```

Коли користувач натискає на заголовок колонки, «_sort_by_column» забезпечує сортування даних у дереві за обраною колонкою. Вона враховує особливості сортування IP адрес і портів, щоб забезпечити правильний порядок. Після сортування функція оновлює відображення дерева і заголовків колонок, щоб відобразити поточний стан сортування.

```
def _search(self):
    field = self.search_field.get()
    value = self.search_value.get()
    field_mapping = {
        'Номер запису': 'id_List',
        'IP маршрутизатора': 'router_ip',
        'Дата та час': 'time_massag',
        'Інтерфейс': 'in_iface',
        'Коментарій': 'comment',
        'MAC джерела': 'src_mac',
        'IP та Port джерела': 'src_ip',
        'IP та Port отримувача': 'dst_ip'
    }
    if field and value:
        search_key = field_mapping.get(field, '')
        if search_key in ['src_ip', 'dst_ip']:
            self.filtered_data = [item for item in self.parsed_data if
                                  value in item[search_key] or value in
item[search_key.replace('_ip', '_port')]]
        else:
            self.filtered_data = [item for item in self.parsed_data if value in
str(item.get(search_key, ''))]
        self._update_treeview()
```

Короткий опис алгоритму функції «_search»:

– отримання поля і значення для пошуку;

– взаємодія зі словником, що відображає зрозумілі для користувача назви полів у відповідні ключі у даних;

– перевірка наявності поля і значення;

– фільтрація даних;

– оновлення вмісту дерева.

В загальному, вона забезпечує механізм пошуку і фільтрації даних у дереві на основі введених користувачем параметрів. Вона отримує вибране поле і значення для пошуку, фільтрує дані за цими параметрами і оновлює відображення дерева для відображення лише відповідних записів.

```
def _get_key_from_column(self, col_index):
    column_mapping = {
        0: 'id_List',
        1: 'router_ip',
        2: 'time_massag',
        3: 'in_iface',
        4: 'comment',
        5: 'src_mac',
        6: 'src_ip',
        7: 'dst_ip'
    }
    return column_mapping.get(col_index, '')
```

Дана функція використовується при сортуванні, щоб визначити, по якому саме ключу (полю) слід сортувати дані і також застосовується у «_get_key_from_column», щоб визначити, по якому полю потрібно сортувати. Вона складається з трьох частин: вхідного параметру «col_index», словника «column_mapping» та поверненням ключа «return column_mapping.get(col_index, '')». Першою є індекс колонки, який відповідає певному полю у таблиці. Друга частина – словник, що зіставляє індекси колонок у дереві з відповідними ключами у даних. Ключі використовуються для доступу до відповідних полів у записах логів. І остання – використовує метод «get» словника, що використовується для отримання значення за заданим ключем (індексом колонки). Якщо індекс не знайдено, повертається порожній рядок ('').

```
def _update_headings(self):
    for col in self.tree['columns']:
        col_text = col.replace(':', ' & ').title()
        if col == self.sort_column:
```

```
            col_text += ' ▲' if self.sort_reverse else ' ▼'
        self.tree.heading(col, text=col_text,
                          command=lambda _col=col:
self._sort_by_column(_col, not self.sort_reverse))
```

«_update_headings» відповідає за оновлення заголовків колонок у дереві (Treeview) після сортування даних. Вона додає відповідні індикатори (▲ або ▼) до заголовків колонок, щоб вказати, за якою колонкою та в якому напрямку дані були відсортовані. Ця функція зазвичай викликається після зміни сортування в дереві, щоб вказати користувачеві поточний стан сортування.

Коротко про код:

– отримує список колонок у дереві та циклом ітерує по кожній колонці в дереві;

– формує текст заголовку;

– додає індикатори сортування;

– оновлює заголовок колонки.

```
@staticmethod
def _ip_to_tuple(ip):
    return tuple(int(part) for part in ip.split('.'))
```

Функція «_ip_to_tuple» перетворює IP-адресу з формату рядка на кортеж цілих чисел для коректного сортування. «_ip_to_tuple» використовується в «_sort_by_column» для сортування IP-адрес у дереві. Оскільки IP-адреси у форматі рядків можуть сортуватися неправильно (наприклад, "192.168.0.10" може йти перед "192.168.0.2"), перетворення їх на кортежі цілих чисел дозволяє правильно порівнювати їх значення.

```
def _clear_search(self):
   self.search_field.set('')
   self.search_value.set('')
   self.filtered_data = self.parsed_data
   self._update_treeview()
```

Функція «_clear_search» очищає вибрані користувачем поля для пошуку та пошукові значення, скидає відфільтровані дані до початкового стану та оновлює відображення дерева для показу всіх записів. Це дозволяє користувачу легко скасувати пошук і переглянути всі доступні дані. Другий і третій рядок призначені щоб очистити поля для вибору пошукового критерію та поля для введення пошукового значення. Четвертий – скидає відфільтровані дані до початкового стану. П’ятий – оновлює відображення даних у дереві.

```
def _on_enter(self, event):
   self._search()
```

«_on_enter» виконує лише одну дію: викликає функцію _search, при натисканні на клавушу Enter. Це дозволяє здійснювати пошук без натискання кнопки "Пошук" вручну, забезпечуючи зручність при використанні програми.

```
def _on_tree_select(self, event):
        selected_item = self.tree.focus()
        if selected_item:
            item_values = self.tree.item(selected_item, 'values')
            src_ip = item_values[6].split(':')[0]
            self._show_ip_activity(src_ip)
```

«_on_tree_select» обробляє подію вибору елемента в дереві користувача. Коли обирається рядок у таблиці, ця функція отримує дані цього рядка і викликає метод _show_ip_activity, щоб показати активність IP-адреси, що міститься в обраному рядку.

Коротко про значення рядків коду:

– оголошення функції;

– отримання вибраного елемента;

– перевірка наявності вибраного елемента;

– отримання значень вибраного елемента;

– витягування IP-адреси джерела;

– виклик методу _show_ip_activity.

Отже, _on_tree_select обробляє вибір елемента в таблиці, витягує IP-адресу з обраного рядка і викликає метод для відображення активності цієї IP-адреси. Це дозволяє легко переглядати детальну активність за конкретними IP-адресами, що робить інтерфейс більш інтуїтивним і функціональним.

```
    def _show_ip_activity(self, ip):
        conn = pyodbc.connect('DRIVER={SQL
Server};SERVER=NOUT;DATABASE=log_data;Trusted_Connection=yes;')
        c = conn.cursor()
        c.execute('''
            SELECT m.time_massag, COUNT(*)
            FROM users u
            JOIN message m ON u.id_messag = m.id_messag
            WHERE u.ip_and_port_sender LIKE ?
            GROUP BY m.time_massag
            ORDER BY m.time_massag
        ''', (ip + ':%',))
        data = c.fetchall()
        conn.close()

        times = [row[0] for row in data]
        counts = [row[1] for row in data]

        fig, ax = plt.subplots()
        ax.plot(times, counts, marker='o')
        ax.set(xlabel='Time', ylabel='Activity Count',
               title=f'Activity for IP {ip}')
        ax.xaxis.set_major_formatter(DateFormatter('%Y-%m-%d %H:%M:%S'))
        fig.autofmt_xdate()
        plt.show()
```

_show_ip_activity встановлює з'єднання з базою даних, виконує SQL-запит для отримання часу повідомлень і кількості записів для заданої IP-адреси, а потім закриває з'єднання. Вона обробляє отримані дані, витягуючи часи і кількість записів, і створює графік активності для заданої IP-адреси за допомогою Matplotlib. Графік показує активність IP-адреси за часом, що дозволяє візуально аналізувати інтенсивність використання мережі.

```
    def _add_to_database(self):
        file_path = filedialog.askopenfilename()
        if file_path:
            logger.info(f"Selected file for parsing: {file_path}")
            parser = LogParser(file_path)
            parser.parse_log()
            logger.info(f"Parsed data from selected file: {parser.parsed_data}")
            save_to_database(parser.parsed_data)
            self.parsed_data = fetch_from_database()
```

```
        self.filtered_data = self.parsed_data
        self._update_treeview()
```

Метод _add_to_database дозволяє вибрати файл журналу для подальшого парсингу та збереження його в базі даних. Після вибору файлу, метод логує цю подію, ініціалізує парсер для обробки обраного файлу та розбирає його вміст. Розпарсені дані потім зберігаються в базі даних, і оновлені дані відображаються в таблиці графічного інтерфейсу. Таким чином, метод робить процес додавання нових даних до бази даних простим і інтуїтивно зрозумілим для користувача, забезпечуючи актуальність відображених даних у графічному інтерфейсі.

Основна функціональність:

– вибір файлу та перевірка наявності шляху до нього;

– логування вибору файлу;

– ініціалізація парсера журналів та їх парсинг;

– логування розпарсених даних;

– збереження даних у базу даних;

– оновлення відображених даних.

Отже, код «log_viewer.py» створює додаток для перегляду логів, який дозволяє користувачам переглядати, фільтрувати, сортувати та додавати логи до бази даних, а також відображати активність IP-адрес на графіку.

2.4.4 Файл main.py

```
import logging
import tkinter as tk
from gui.log_viewer import LogViewerApp
from database import fetch_from_database
from utils.logger import setup_logging
```

Першою частиною файлу є імпорт бібліотек та модулів. Бібліотека «logging» призначена для роботи з логуванням. «tkinter as tk» – бібліотека для створення

графічного інтерфейсу користувача. «LogViewerApp» – клас для відображення логів у графічному інтерфейсі. «fetch_from_database» – функція для отримання дах з бази даних. «setup_logging» – функція для налаштування логування.

```
logger = logging.getLogger(__name__)
setup_logging()
```

Перший рядок призначений для створення логера для поточного модуля, а другий – для виклику функції для налаштування логування.

Також присутня основна функція main(), яка запускає програму.

```
def initialize():
        logger.info("Initializing application")
        root = tk.Tk()
        root.geometry("1200x800")
        app = LogViewerApp(root, fetch_from_database())
        root.resizable(True, True)
        root.protocol("WM_DELETE_WINDOW", root.quit)
        root.mainloop()
        logger.info("Application closed")
```

Функція «initialize» відповідає за початкове налаштування та запуск графічного інтерфейсу програми. Вона створює головне вікно, налаштовує його параметри (розмір, можливість зміни розміру), створює додаток для перегляду логів, отримує дані з бази даних, налаштовує обробку події закриття вікна і запускає основний цикл обробки подій.

```
if __name__ == "__main__":
main()
```

Ця частина перевіряє, чи модуль запущений як основний, і викликає функцію «main».

В загальному, код файлу «main.py» налаштовує логування, створює та налаштовує головне вікно програми з використанням «tkinter», створює екземпляр класу «LogViewerApp» для відображення логів, завантажених з бази даних, і запускає основний цикл обробки подій.

2.3 Висновок до розділу

Отже, налаштування маршрутизатора Mikrotik є критичним кроком для забезпечення надійної та безпечної роботи мережі. Мережі та системи мають бути захищеними, а дані – надійно збереженими. Наявність мережі з відмінною продуктивністю, але недостатньо налаштованими правилами фаєрволу, може мати негативні наслідки [36]. В даному розділі було описано конфігурацію IP-адрес, налаштування фаєрволу, маршрутизації, а також налаштування моніторингу та логування. Правильна конфігурація дозволяє забезпечити оптимальну продуктивність мережі, захист від несанкціонованого доступу та збір необхідних даних для аналізу мережевої активності.

Не менш важливим є проєктування та розробка бази даних, що є ключовим етапом для організації та збереження даних. Коли справа доходить до вибору, використання та обслуговування бази даних, розуміння її внутрішньої структури є надзвичайно важливим [37]. Було обрано системи керування базами даних, створено схему бази даних, визначено таблиці, поля та зв'язки між ними.

Шляхом застосування загальновизнаних принципів архітектури програмного забезпечення можна значно покращити ефективність розробки на протязі усього життєвого циклу будь-якої програми.[38] Під час проєктування та розробки програмного забезпечення було обрано мову програмування, у нашому випадку було обрано Python за його простоту, читабельність та велику кількість доступних бібліотек. Також було обрано основні програмні бібліотеки, визначено структуру та описано код програмного забезпечення.

3. ОГЛЯД ФУНКЦІОНУВАННЯ ЛОГУВАННЯ ТА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Загальний алгоритм функціональності програми

Алгоритми — це методи, які програми використовують для роботи зі структурами даних [39]. Усі професійні розробники повинні розуміти, які структури даних та алгоритми слід застосовувати під час розробки. Вибір цих елементів безпосередньо впливає на ефективність роботи програми [40].

У даній кваліфікаційній роботі було складено загальний алгоритм функціонування розробленого програмного забезпечення (рисунок 3.1).

Алгоритм починається зі старту роботи програми. Спочатку запускається Visual syslog server для збирання логів мережевої активності, які потім записуються у файл. Після цього можемо запустити розроблену програму, яка дозволяє переглядати та аналізувати ці логи.

Користувач має можливість оновити базу даних новими логами. Якщо він вирішить це зробити, то має обирати записаний лог файл та оновити базу даних. Якщо оновлення не потрібне, можна перейти до перегляду вже наявної бази даних.

Під час перегляду, вирішуємо, чи потрібно знайти конкретний запис. У випадку необхідності пошуку, потрібно ввести критерії пошуку у програмі та знайти потрібний запис.

Далі розглядаємо, чи потрібно переглянути графік активності конкретного користувача. Якщо так, то можна обрати необхідну IP-адресу зі списку та переглядає відповідний графік активності.

Нарешті, є вибір, чи завершувати роботу з програмою. Якщо так, програма завершує роботу. Якщо ні, користувач може продовжити роботу з базою даних або повернутись до початку процесу.

Таким чином, цей алгоритм забезпечує зручний і послідовний процес роботи з логами мережевої активності, дозволяючи оновлювати базу даних, здійснювати пошук конкретних записів і переглядати графіки активності.

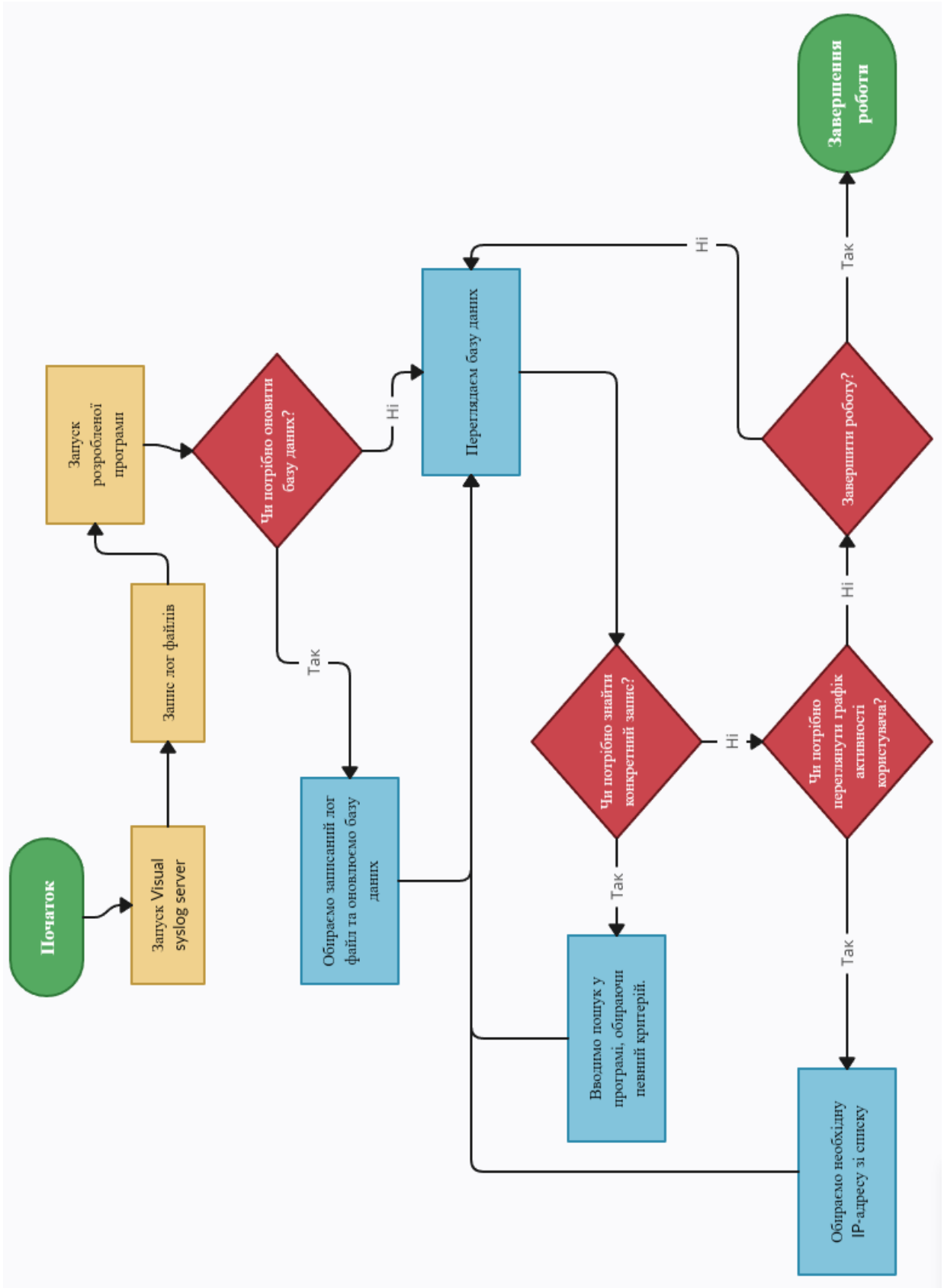


Рисунок 3.1 – Загальний алгоритм функціоналу розробленої програми

3.2 Огляд запису логів та функціонування програми

Після налаштування маршрутизатора на логування TCP пакетів, було запущено програму Visual Syslog Server, у якій було вказано директорію для запису лог-файлів. Після запуску програма починає автоматично збирати логи активності користувачів у мережі, що вказано на рисунок 2.9.

Після закриття програми Visual Syslog Server, додаток оновлює файл, на який було вказано в налаштуваннях, і зберігає інформацію у файлі syslog (рисунок 3.2).

Наступним кроком буде відкриття розробленого програмного забезпечення «Log Viewer» (рисунок 3.3). Після завантаження програми у головному вікні буде зображена таблиця із логами, які є в базі даних. Якщо запуск програми відбувається уперше, то це поле буде пусте. Таблиця має у собі такі стовпці як: «Номер запису», «Ip маршрутизатора», «Дату та Час», «Інтерфейс», «Коментар», «MAC джерела», «Ip та Port джерела» та «Ip та Port отримувача».

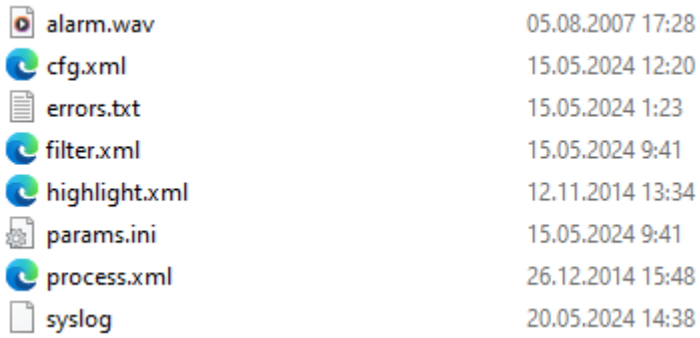
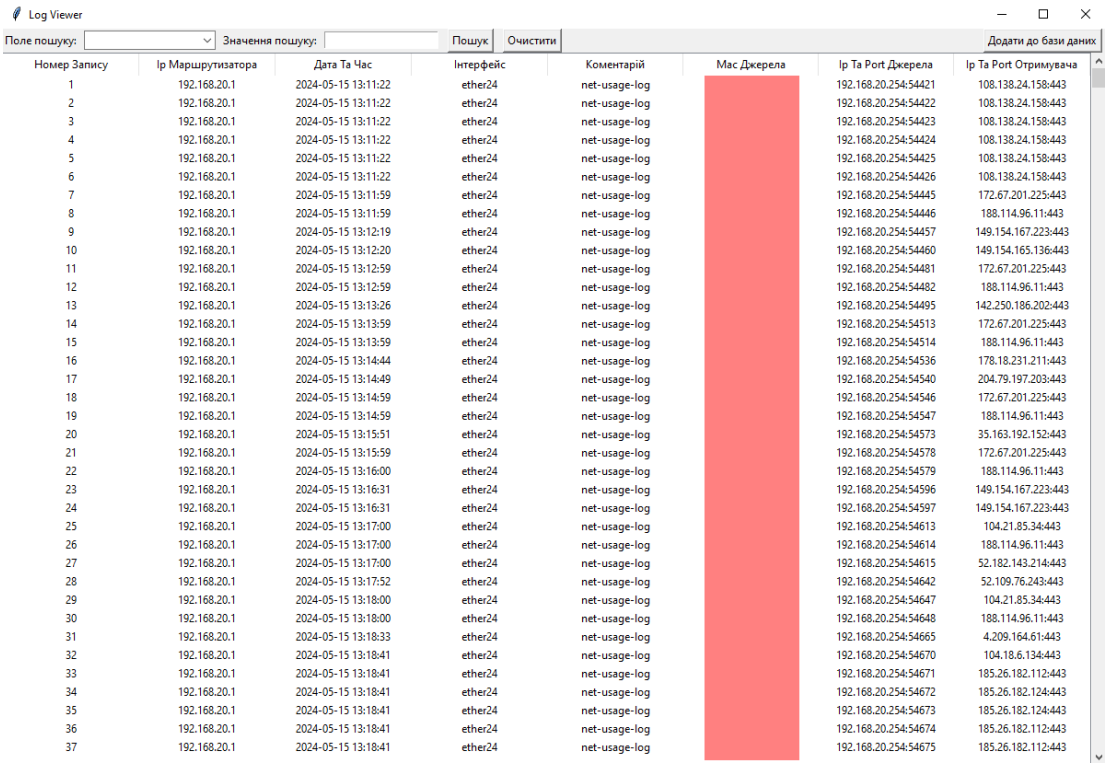


Рисунок 3.2 – Вміст папки із файлом запису syslog

Також наявний випадаючий список, що є «Полем пошуку», де ми можемо обрати критерій за яким ми будемо здійснювати пошук. Наступним є поле вводу, що підписано «Значення пошуку», де ми можемо ввести дані з конкретного запису з обраного критерію.

У програмі реалізовано сортування записів по стовпцях у таблиці. Для їх використання потрібно лише нажати на стовпці для сортування у порядку збільшення і другий раз для порядку зменшення.



Номер Запису	Ip Маршрутизатора	Дата Та Час	Інтерфейс	Коментарій	Mac Джерела	Ip Та Port Джерела	Ip Та Port Отримувача
1	192.168.20.1	2024-05-15 13:11:22	ether24	net-usage-log		192.168.20.254:54421	108.138.24.158:443
2	192.168.20.1	2024-05-15 13:11:22	ether24	net-usage-log		192.168.20.254:54422	108.138.24.158:443
3	192.168.20.1	2024-05-15 13:11:22	ether24	net-usage-log		192.168.20.254:54423	108.138.24.158:443
4	192.168.20.1	2024-05-15 13:11:22	ether24	net-usage-log		192.168.20.254:54424	108.138.24.158:443
5	192.168.20.1	2024-05-15 13:11:22	ether24	net-usage-log		192.168.20.254:54425	108.138.24.158:443
6	192.168.20.1	2024-05-15 13:11:22	ether24	net-usage-log		192.168.20.254:54426	108.138.24.158:443
7	192.168.20.1	2024-05-15 13:11:59	ether24	net-usage-log		192.168.20.254:54445	172.67.201.225:443
8	192.168.20.1	2024-05-15 13:11:59	ether24	net-usage-log		192.168.20.254:54446	188.114.96.11:443
9	192.168.20.1	2024-05-15 13:12:19	ether24	net-usage-log		192.168.20.254:54457	149.154.167.223:443
10	192.168.20.1	2024-05-15 13:12:20	ether24	net-usage-log		192.168.20.254:54460	149.154.165.136:443
11	192.168.20.1	2024-05-15 13:12:59	ether24	net-usage-log		192.168.20.254:54481	172.67.201.225:443
12	192.168.20.1	2024-05-15 13:12:59	ether24	net-usage-log		192.168.20.254:54482	188.114.96.11:443
13	192.168.20.1	2024-05-15 13:13:26	ether24	net-usage-log		192.168.20.254:54495	142.250.186.202:443
14	192.168.20.1	2024-05-15 13:13:59	ether24	net-usage-log		192.168.20.254:54513	172.67.201.225:443
15	192.168.20.1	2024-05-15 13:13:59	ether24	net-usage-log		192.168.20.254:54514	188.114.96.11:443
16	192.168.20.1	2024-05-15 13:14:44	ether24	net-usage-log		192.168.20.254:54536	178.18.231.211:443
17	192.168.20.1	2024-05-15 13:14:49	ether24	net-usage-log		192.168.20.254:54540	204.79.197.203:443
18	192.168.20.1	2024-05-15 13:14:59	ether24	net-usage-log		192.168.20.254:54546	172.67.201.225:443
19	192.168.20.1	2024-05-15 13:14:59	ether24	net-usage-log		192.168.20.254:54547	188.114.96.11:443
20	192.168.20.1	2024-05-15 13:15:51	ether24	net-usage-log		192.168.20.254:54573	35.163.192.152:443
21	192.168.20.1	2024-05-15 13:15:59	ether24	net-usage-log		192.168.20.254:54578	172.67.201.225:443
22	192.168.20.1	2024-05-15 13:16:00	ether24	net-usage-log		192.168.20.254:54579	188.114.96.11:443
23	192.168.20.1	2024-05-15 13:16:31	ether24	net-usage-log		192.168.20.254:54596	149.154.167.223:443
24	192.168.20.1	2024-05-15 13:16:31	ether24	net-usage-log		192.168.20.254:54597	149.154.167.223:443
25	192.168.20.1	2024-05-15 13:17:00	ether24	net-usage-log		192.168.20.254:54613	104.21.85.34:443
26	192.168.20.1	2024-05-15 13:17:00	ether24	net-usage-log		192.168.20.254:54614	188.114.96.11:443
27	192.168.20.1	2024-05-15 13:17:00	ether24	net-usage-log		192.168.20.254:54615	52.182.143.214:443
28	192.168.20.1	2024-05-15 13:17:52	ether24	net-usage-log		192.168.20.254:54642	52.109.76.243:443
29	192.168.20.1	2024-05-15 13:18:00	ether24	net-usage-log		192.168.20.254:54647	104.21.85.34:443
30	192.168.20.1	2024-05-15 13:18:00	ether24	net-usage-log		192.168.20.254:54648	188.114.96.11:443
31	192.168.20.1	2024-05-15 13:18:33	ether24	net-usage-log		192.168.20.254:54665	4.209.164.61:443
32	192.168.20.1	2024-05-15 13:18:41	ether24	net-usage-log		192.168.20.254:54670	104.18.6.134:443
33	192.168.20.1	2024-05-15 13:18:41	ether24	net-usage-log		192.168.20.254:54671	185.26.182.112:443
34	192.168.20.1	2024-05-15 13:18:41	ether24	net-usage-log		192.168.20.254:54672	185.26.182.124:443
35	192.168.20.1	2024-05-15 13:18:41	ether24	net-usage-log		192.168.20.254:54673	185.26.182.124:443
36	192.168.20.1	2024-05-15 13:18:41	ether24	net-usage-log		192.168.20.254:54674	185.26.182.112:443
37	192.168.20.1	2024-05-15 13:18:41	ether24	net-usage-log		192.168.20.254:54675	185.26.182.112:443

Рисунок 3.3 – Інтерфейс розробленої програми

Ще на головному екрані присутні три кнопки: «Пошук», «Очистити» та «Додати до бази даних» (рисунок 3.4). Перша кнопка здійснює пошук за певними критеріями та даними, що були раніше вказані у полях. Пошук також можна розпочати не натискаючи кнопку, а нажавши клавішу Enter на клавіатурі. Друга кнопка очищає поля, скидає параметри пошуку та виводить начальні записи без фільтрів та сортування.

Рисунок 3.4 – Панель інструментів головного вікна

Третя кнопка дозволяє додати логи із файлів у базу даних. При її натисканні відкриється окреме вікно, де потрібно обрати конкретний файл із логами або відмінити додавання у разі помилкового натискання.

При натисканні на IP адресу та порт любого запису, буде відкрито графік активності певного користувача, що має обрану IP адресу (рисунок 3.5). У цьому вікні ми можемо побачити графік відношення кількості запитів до дати, коли були ці запити. У цьому вікні також присутні 7 кнопки. Перша скидує вигляд графіку до початкового вигляду, після змін його виду. Друга та третя із зображенням стрілок, відповідно переключає між попередніми та наступними діями. Четверта – дозволяє переміщувати вид графіку. Це дозволяє сконцентруватися на певній частині графіку. П’ята – дозволяє збільшувати чи віддаляти вид на графіку для детальнішої інформації по кількості запитів та дозволяє побачити точніший час при зближенні. Шоста – дозволяє змінити розміри поля графіка. І остання, дозволяє зберегти графік у вигляді картинки у різних форматах на комп’ютері.

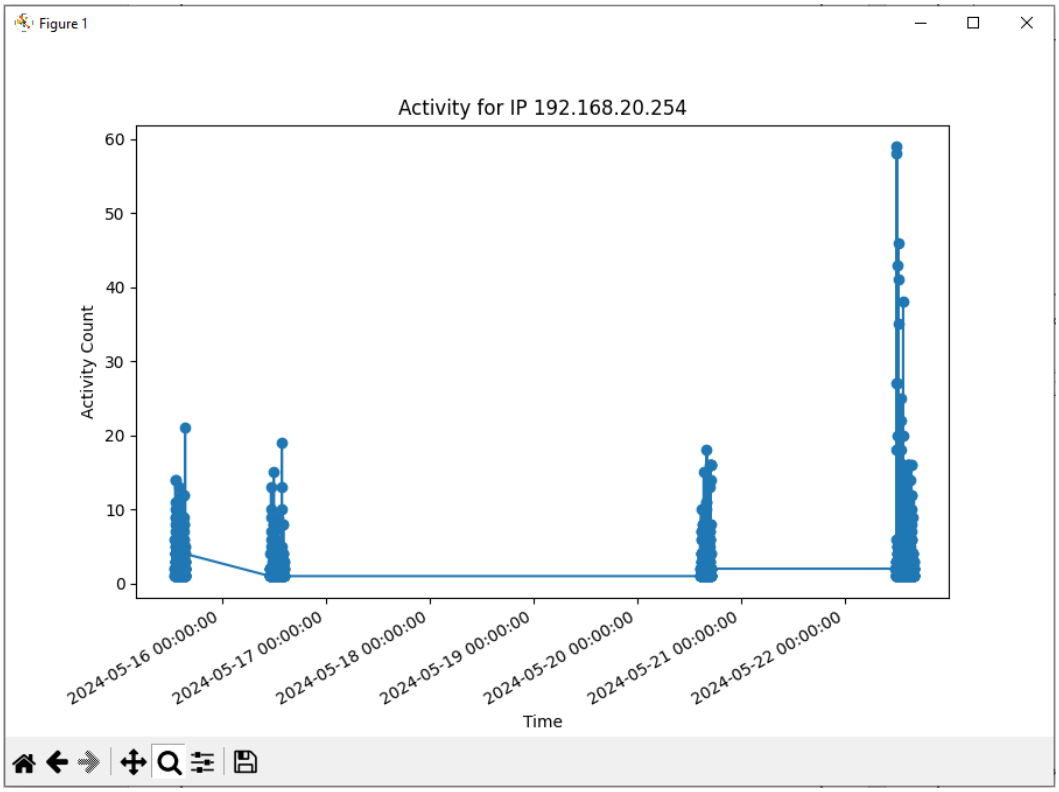


Рисунок 3.5 – Інтерфейс розробленої програми

3.3 Висновок до розділу

У цьому розділі описано детальний алгоритм функціонування розробленого програмного забезпечення для роботи з мережевими логами. Основні кроки включають запуск Visual syslog server для збору логів, відкриття програми для перегляду та аналізу логів, оновлення бази даних новими логами, пошук конкретних записів за критеріями та перегляд графіків активності користувачів на основі IP-адрес. Інтерфейс програми інтуїтивно зрозумілий, з можливістю сортування записів, пошуку та додавання нових логів. Це забезпечує ефективну роботу з логами, надаючи всю необхідну інформацію у зручній формі.

У процесі роботи з програмою користувач має можливість налаштовувати маршрутизатор для логування TCP пакетів, запускати Visual Syslog Server для збирання логів та використовувати розроблену програму для перегляду та аналізу цих логів. Програма надає функції пошуку за різними критеріями, сортування записів, а також візуалізацію активності користувачів. Це дозволяє користувачам ефективно аналізувати мережеву активність та приймати обґрунтовані рішення на основі зібраних даних.

ВИСНОВОК

У даній кваліфікаційній роботі було виконано детальний аналіз предметної області, а також проведено дослідження необхідних теоретичних аспектів для розробки системи логування мережевої активності користувачів. На основі отриманих знань було обрано відповідне апаратне та програмне забезпечення, зокрема, використання NAT та DHCP для налаштування мережевої інфраструктури.

Було розглянуто різні засоби віддаленого збору логів, такі як Zabbix, Rsyslog, Splunk та Wireshark, з метою визначення найоптимальнішого рішення для поставленого завдання. Подальші етапи роботи включали налаштування маршрутизатора Mikrotik, зокрема налаштування брандмауера, DHCP-клієнта та сервера, а також інтеграцію системи логування із Visual Syslog Server.

Особлива увага була приділена проєктуванню та розробці бази даних для зберігання зібраних логів, а також розробці програмного забезпечення з використанням відповідних мов програмування та програмних бібліотек. Було створено кілька основних файлів програмного забезпечення, таких як log_parser.py, logger.py, database.py, log_viewer.py та main.py, які забезпечують функціональність збору, обробки, зберігання та аналізу логів.

Фінальний етап включав тестування розробленого програмного забезпечення для забезпечення його коректної роботи та відповідності поставленим вимогам. Результати тестування підтвердили ефективність і надійність створеної системи.

Загалом, розробивши комплексне рішення для збору даних про активність користувачів у локальній мережі, їх зберігання в базі даних та надання можливостей для пошуку та аналізу зібраної інформації, мета проєкту виконана.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Florian S., Wurzenberger M., Landauer M. Smart Log Data Analytics: Techniques for Advanced Security Analysis. Luxemburg: Springer, 2021. 360 p.
2. Log File Formats. Xcitium. URL: https://www.xcitium.com/log-files/ (Дата звернення 06.05.2024)
3. Колба С. Аналіз існуючих систем логування та активності Центру цифрових технологій ХНУ. Звіт з преддип. практ. студ. 2024. с. 28.
4. Новини Полтави. Роутери компанії Microtik – надійні та доступні маршрутизатори. KOLO.NEWS. 2022 URL: http://surl.li/qubry (Дата звернення 02.05.2024).
5. WinBox. Help.mikrotik URL: https://help.mikrotik.com/docs/display/ROS/WinBox (Дата звернення 04.05.2024).
6. Max Belkov. Visual Syslog Server for Windows. GitHub. 2016. URL: https://github.com/MaxBelkov/visualsyslog?tab=readme-ov-file (Дата звернення 04.05.2024)
7. CIDR. Wikipedia. URL: https://uk.wikipedia.org/wiki/CIDR (Дата звернення 05.05.2024)
8. Introduction of Firewall in Computer Network. Geeks for geeks. URL: https://www.geeksforgeeks.org/introduction-of-firewall-in-computer-network/ (Дата звернення 05.05.2024)
9. Network Address Translation. Avinetworks. URL: https://avinetworks.com/glossary/network-address-translation/ (Дата звернення 06.05.2024)
10. Dynamic Host Configuration Protocol. Wikipedia. URL: https://en.wikipedia.org/wiki/Dynamic_Host_Configuration_Protocol (Дата звернення 06.05.2024)
11. IP Addressing: DHCP Configuration Guide, Cisco IOS Release 15SY. Cisco. URL: http://surl.li/ulpjo (Дата звернення 06.05.2024)

12. Keep track of your network's health and performance. Zabbix. URL: https://www.zabbix.com/network_monitoring (Дата звернення 06.05.2024)

13. Anna Poga. Zabbix. GitHub. 2024. URL: https://github.com/zabbix/zabbix (Дата звернення 06.05.2024)

14. Zabbix. TechExpert. URL: https://techexpert.ua/it-products/zabbix/ (Дата звернення 06.05.2024)

15. Welcome to Rsyslog. Rsyslog. URL: https://www.rsyslog.com/doc/index.html (Дата звернення 06.05.2024)

16. How to Collect, Process, and Ship Log Data with Rsyslog. Better Stack. URL: https://betterstack.com/community/guides/logging/rsyslog-explained/ (Дата звернення 06.05.2024)

17. Rsyslog. Geek University. URL: https://geek-university.com/rsyslog/ (Дата звернення 06.05.2024)

18. Splunk. TechExpert. URL: https://techexpert.ua/it-products/splunk-platform/ (Дата звернення 06.05.2024)

19. What is Splunk? Fortinet. URL: https://www.fortinet.com/resources/cyberglossary/what-is-splunk (Дата звернення 06.05.2024)

20. Power the SOC of the Future. Splunk. URL: https://www.splunk.com/en_us/products/cyber-security.html (Дата звернення 06.05.2024)

21. Wireshark User's Guide. Wireshark URL: https://www.wireshark.org/docs/wsug_html/ (Дата звернення 06.05.2024)

22. Bock L. Learn Wireshark: A definitive guide to expertly analyzing protocols and troubleshooting networks using Wireshark. 2nd ed. Birmingham: Packt Publishing, 2022. 606 p.

23. Edward Kost. What is Wireshark? The Free Network Sniffing Tool. UpGuard. 2023. URL: https://www.upguard.com/blog/what-is-wireshark стаття (Дата звернення 06.05.2024)

24. Kumar Nainar N., Ramdoss Y., Orzach Y. Network Analysis Using Wireshark 2 Cookbook - Second Edition: Practical recipes to analyze and secure your network using Wireshark 2. 2nd ed. Birmingham: Packt Publishing, 2018. 626 p.

25. Pahrial Ms. Configure Bridge on MikroTik Router. Medium. 2021 URL: https://pahrialms.medium.com/configure-bridge-on-mikrotik-router-d5c6a44cc38a (Дата звернення 07.05.2024)

26. Даталогічна модель бази даних. Studwood. URL: https://studwood.ru/1553750/ekonomika/datalogichna_model_bazi_danih (Дата звернення 07.05.2024)

27. Колесник В. В. Бази даних та їх види. Основні поняття. Класифікація БД. Види СУБД. MS Access. На Урок. 2018. – URL: https://naurok.com.ua/bazi-danih-ta-h-vidi-osnovni-ponyattya-klasifikaciya-bd-vidi-subd-ms-access-74441.html (Дата звернення 08.05.2024)

28. Joanna Pasterczyk. Все, що ви маєте знати про Python. Які в нього недоліки, а які переваги? Justgeek.it. 2022. URL: http://surl.li/uaxyy (Дата звернення 08.05.2024)

29. Romano F. Learn Python Programming - Second Edition: The no-nonsense, beginner's guide to programming, data science, and web development with Python 3.7. 2nd ed. Birmingham: Packt Publishing, 2018. 508 p.

30. Vadim Korolik. The 5 Best Logging Libraries for Python. Highlight.io. 2023. URL: https://www.highlight.io/blog/5-best-python-logging-libraries (Дата звернення 09.05.2024)

31. Громко Г. Знайомимось з Tkinter. Робимо перші кроки, і це нескладно. Medium. 2024 URL: http://surl.li/uayec (Дата звернення 09.05.2024)

32. re – Regular expression operations. Python. URL: https://docs.python.org/uk/3/library/re.html (Дата звернення 10.05.2024)

33. Стандартна бібліотека мови Python. Wikipedia. URL: http://surl.li/uaygd (Дата звернення 10.05.2024)

34. Connecting to ODBC databases with pyodbc in Python. Devart. URL: http://surl.li/uayie (Дата звернення 11.05.2024)

35. Matplotlib. Wikipedia. URL: https://en.wikipedia.org/wiki/Matplotlib (Дата звернення 11.05.2024)

36. Hart T. Networking with MikroTik: MTCNA Study Guide. Chicago: Independently published, 2017. 360 p.

37. Petrov A. Database Internals: A Deep Dive into How Distributed Data Systems Work. 1st ed. Sebastopol: O'Reilly Media, 2019. 598 p.

38. Martin R. Clean Architecture: A Craftsman's Guide to Software Structure and Design (Robert C. Martin Series). 1st ed. London: Pearson, 2017. 432 p.

39. Lafore R. Data Structures and Algorithms in Java. 2nd ed. Carmel: Sams Publishing, 2017. 800 p.

40. Karimov E. Data Structures and Algorithms in Swift: Implement Stacks, Queues, Dictionaries, and Lists in Your Apps. 1st ed. New York: Apress, 2020. 226 p.

ДОДАТОК А

(Обов’язковий)

Код програмного забезпечення

Main.py

```
import logging
import tkinter as tk
from gui.log_viewer import LogViewerApp
from database import fetch_from_database
from utils.logger import setup_logging

logger = logging.getLogger(__name__)
setup_logging()

def main():
    def initialize():
        logger.info("Initializing application")
        root = tk.Tk()
        root.geometry("1200x800")
        app = LogViewerApp(root, fetch_from_database())
        root.resizable(True, True)
        root.protocol("WM_DELETE_WINDOW", root.quit)
        root.mainloop()
        logger.info("Application closed")

    initialize()

if __name__ == "__main__":
    main()
```

Database.py

```
import pyodbc
import logging

logger = logging.getLogger(__name__)

def save_to_database(parsed_data):
    conn = pyodbc.connect('DRIVER={SQL
Server};SERVER=NOUTTUF;DATABASE=log_data;Trusted_Connection=yes;')
    c = conn.cursor()
    for entry in parsed_data:
        c.execute('''
        SELECT 1 FROM users u
        JOIN router r ON u.id_router = r.id_router
        JOIN message m ON u.id_messag = m.id_messag
        WHERE r.ip_router = ? AND u.in_interface = ? AND u.comment = ? AND
u.mac_source = ?
        AND u.ip_and_port_sender = ? AND u.ip_and_port_receiver = ? AND
m.time_massag = ?
```

```
        AND m.tipe_local_messag = ? AND m.tipe_messag = ? AND m.source_messag
= ?
        AND m.protocol_messag = ? AND m.length_messag = ?
        ''', (entry['router_ip'], entry['in_iface'], entry['comment'],
entry['src_mac'],
              entry['src_ip'] + ':' + entry['src_port'], entry['dst_ip'] +
':' + entry['dst_port'],
              entry['time_massag'], entry['tipe_local_messag'],
entry['tipe_messag'],
              entry['source_messag'], entry['protocol_messag'],
entry['length_messag']))

        if c.fetchone():
            logger.debug("Duplicate entry found, skipping insert.")
            continue

        logger.debug(f"Saving entry to database: {entry}")

        c.execute('SELECT id_router FROM router WHERE ip_router=?',
(entry['router_ip'],))
        router_row = c.fetchone()
        if router_row:
            id_router = router_row[0]
        else:
            c.execute('INSERT INTO router (ip_router, out_Interface) VALUES
(?, ?)',
                      (entry['router_ip'], entry['out_iface']))
            id_router = c.execute('SELECT @@IDENTITY').fetchone()[0]

        c.execute(
            'INSERT INTO message (time_massag, tipe_local_messag,
tipe_messag, source_messag, protocol_messag, length_messag) VALUES
(?, ?, ?, ?, ?, ?)',
            (entry['time_massag'], entry['tipe_local_messag'],
entry['tipe_messag'], entry['source_messag'],
             entry['protocol_messag'], entry['length_messag']))
        id_messag = c.execute('SELECT @@IDENTITY').fetchone()[0]

        c.execute(
            'INSERT INTO users (id_router, id_messag, in_interface, comment,
mac_source, ip_and_port_sender, ip_and_port_receiver) VALUES
(?, ?, ?, ?, ?, ?, ?)',
            (id_router, id_messag, entry['in_iface'], entry['comment'],
entry['src_mac'],
             entry['src_ip'] + ':' + entry['src_port'], entry['dst_ip'] + ':'
+ entry['dst_port']))

    conn.commit()
    conn.close()

def fetch_from_database():
    conn = pyodbc.connect('DRIVER={SQL
Server};SERVER=NOUTTUF;DATABASE=log_data;Trusted_Connection=yes;')
    c = conn.cursor()
    c.execute(
        'SELECT u.id_List, r.ip_router, m.time_massag, u.in_interface,
u.comment, u.mac_source, u.ip_and_port_sender, u.ip_and_port_receiver FROM
users u JOIN router r ON u.id_router = r.id_router JOIN message m ON
u.id_messag = m.id_messag ORDER BY u.id_List ASC')
    data = c.fetchall()
    conn.close()
    return [{'id_List': row[0], 'router_ip': row[1], 'time_massag': row[2],
'in_iface': row[3], 'comment': row[4],
             'src_mac': row[5], 'src_ip': row[6].split(':')[0], 'src_port':
```

```
row[6].split(':')[1],
            'dst_ip': row[7].split(':')[0], 'dst_port':
row[7].split(':')[1]} for row in data]
```

Log_viewer.py

```
import tkinter as tk
from tkinter import ttk, filedialog
import matplotlib.pyplot as plt
from matplotlib.dates import DateFormatter
import pyodbc
from log_parser import LogParser
from database import save_to_database, fetch_from_database
import logging

logger = logging.getLogger(__name__)

class LogViewerApp:
    def __init__(self, master, parsed_data):
        self.master = master
        self.parsed_data = parsed_data
        self.filtered_data = parsed_data
        self.master.title("Log Viewer")
        self.tree = ttk.Treeview(master, show="headings")
        self.scrollbar = ttk.Scrollbar(master, orient="vertical",
command=self.tree.yview)
        self.tree.configure(yscrollcommand=self.scrollbar.set)
        self.sort_column = None
        self.sort_reverse = False
        self.search_field = tk.StringVar()
        self.search_value = tk.StringVar()
        self._setup_widgets()
        self._build_tree()

    def _setup_widgets(self):
        control_frame = tk.Frame(self.master)
        control_frame.pack(fill='x')

        search_frame = tk.Frame(control_frame)
        search_frame.pack(side='left')

        tk.Label(search_frame, text="Поле пошуку:").pack(side='left')
        search_options = ['Номер запису', 'IP маршрутизатора', 'Дата та час',
'Інтерфейс', 'Коментарій', 'MAC джерела',
                          'IP та Port джерела', 'IP та Port отримувача']
        search_dropdown = ttk.Combobox(search_frame,
textvariable=self.search_field, values=search_options)
        search_dropdown.pack(side='left', padx=5)

        tk.Label(search_frame, text="Значення пошуку:").pack(side='left')
        search_entry = tk.Entry(search_frame, textvariable=self.search_value)
        search_entry.pack(side='left', padx=5)
        search_entry.bind("<Return>", self._on_enter)

        search_button = tk.Button(search_frame, text="Пошук",
command=self._search)
        search_button.pack(side='left', padx=5)
```

```
        clear_button = tk.Button(search_frame, text="Очистити",
command=self._clear_search)
        clear_button.pack(side='left', padx=5)

        add_button = tk.Button(control_frame, text="Додати до бази даних",
command=self._add_to_database)
        add_button.pack(side='right', padx=5)

        self.scrollbar.pack(side='right', fill='y')
        self.tree.pack(side='left', fill='both', expand=True)
        self.tree.bind("<ButtonRelease-1>", self._on_tree_select)

    def _build_tree(self):
        self.tree['columns'] = (
            'Номер запису', 'IP маршрутизатора', 'Дата та час', 'Інтерфейс',
'Коментарій', 'MAC джерела',
            'IP та Port джерела', 'IP та Port отримувача')
        for col in self.tree['columns']:
            col_text = col.replace(':', ' & ').title()
            self.tree.heading(col, text=col_text,
                              command=lambda _col=col:
self._sort_by_column(_col, not self.sort_reverse))
            self.tree.column(col, width=100, anchor='center', stretch=True)
        self._update_treeview()

    def _update_treeview(self):
        for row in self.tree.get_children():
            self.tree.delete(row)
        for item in self.filtered_data:
            self.tree.insert('', 'end', values=(
                item.get('id_List'), item.get('router_ip'),
item.get('time_massag'), item.get('in_iface'),
                item.get('comment'), item.get('src_mac'), item.get('src_ip')
+ ':' + item.get('src_port'),
                item.get('dst_ip') + ':' + item.get('dst_port')))

    def _sort_by_column(self, col, reverse):
        col_index = self.tree['columns'].index(col)
        if col == 'IP та Port джерела':
            self.filtered_data.sort(key=lambda x:
(self._ip_to_tuple(x['src_ip']), int(x['src_port'])), reverse=reverse)
        elif col == 'IP та Port отримувача':
            self.filtered_data.sort(key=lambda x:
(self._ip_to_tuple(x['dst_ip']), int(x['dst_port'])), reverse=reverse)
        else:
            self.filtered_data.sort(key=lambda x:
x[self._get_key_from_column(col_index)], reverse=reverse)
        self.sort_column = col
        self.sort_reverse = reverse
        self._update_treeview()
        self._update_headings()

    def _get_key_from_column(self, col_index):
        column_mapping = {
            0: 'id_List',
            1: 'router_ip',
            2: 'time_massag',
            3: 'in_iface',
            4: 'comment',
            5: 'src_mac',
            6: 'src_ip',
            7: 'dst_ip'
        }
        return column_mapping.get(col_index, '')
```

```
        def _update_headings(self):
            for col in self.tree['columns']:
                col_text = col.replace(':', ' & ').title()
                if col == self.sort_column:
                    col_text += ' ▲' if self.sort_reverse else ' ▼'
                self.tree.heading(col, text=col_text,
                                  command=lambda _col=col:
self._sort_by_column(_col, not self.sort_reverse))

    @staticmethod
    def _ip_to_tuple(ip):
        return tuple(int(part) for part in ip.split('.'))

    def _search(self):
        field = self.search_field.get()
        value = self.search_value.get()
        field_mapping = {
            'Номер запису': 'id_List',
            'IP маршрутизатора': 'router_ip',
            'Дата та час': 'time_massag',
            'Інтерфейс': 'in_iface',
            'Коментарій': 'comment',
            'MAC джерела': 'src_mac',
            'IP та Port джерела': 'src_ip',
            'IP та Port отримувача': 'dst_ip'
        }
        if field and value:
            search_key = field_mapping.get(field, '')
            if search_key in ['src_ip', 'dst_ip']:
                self.filtered_data = [item for item in self.parsed_data if
                                      value in item[search_key] or value in
item[search_key.replace('_ip', '_port')]]
            else:
                self.filtered_data = [item for item in self.parsed_data if
value in str(item.get(search_key, ''))]
            self._update_treeview()

    def _clear_search(self):
        self.search_field.set('')
        self.search_value.set('')
        self.filtered_data = self.parsed_data
        self._update_treeview()

    def _on_enter(self, event):
        self._search()

    def _on_tree_select(self, event):
        selected_item = self.tree.focus()
        if selected_item:
            item_values = self.tree.item(selected_item, 'values')
            src_ip = item_values[6].split(':')[0]
            self._show_ip_activity(src_ip)

    def _show_ip_activity(self, ip):
        conn = pyodbc.connect('DRIVER={SQL
Server};SERVER=NOUTTUF;DATABASE=log_data;Trusted_Connection=yes;')
        c = conn.cursor()
        c.execute('''
            SELECT m.time_massag, COUNT(*)
            FROM users u
            JOIN message m ON u.id_messag = m.id_messag
            WHERE u.ip_and_port_sender LIKE ?
            GROUP BY m.time_massag
```

```
            ORDER BY m.time_massag
        ''', (ip + ':%',))
        data = c.fetchall()
        conn.close()

        times = [row[0] for row in data]
        counts = [row[1] for row in data]

        fig, ax = plt.subplots()
        ax.plot(times, counts, marker='o')
        ax.set(xlabel='Time', ylabel='Activity Count',
               title=f'Activity for IP {ip}')
        ax.xaxis.set_major_formatter(DateFormatter('%Y-%m-%d %H:%M:%S'))
        fig.autofmt_xdate()
        plt.show()

    def _add_to_database(self):
        file_path = filedialog.askopenfilename()
        if file_path:
            logger.info(f"Selected file for parsing: {file_path}")
            parser = LogParser(file_path)
            parser.parse_log()
            logger.info(f"Parsed data from selected file:
{parser.parsed_data}")
            save_to_database(parser.parsed_data)
            self.parsed_data = fetch_from_database()
            self.filtered_data = self.parsed_data
            self._update_treeview()
```

Logger.py

```
import logging

def setup_logging():
    logging.basicConfig(level=logging.DEBUG,
                        format='%(asctime)s - %(name)s - %(levelname)s
- %(message)s',
                        datefmt='%Y-%m-%d %H:%M:%S')
```

Log_parser.py

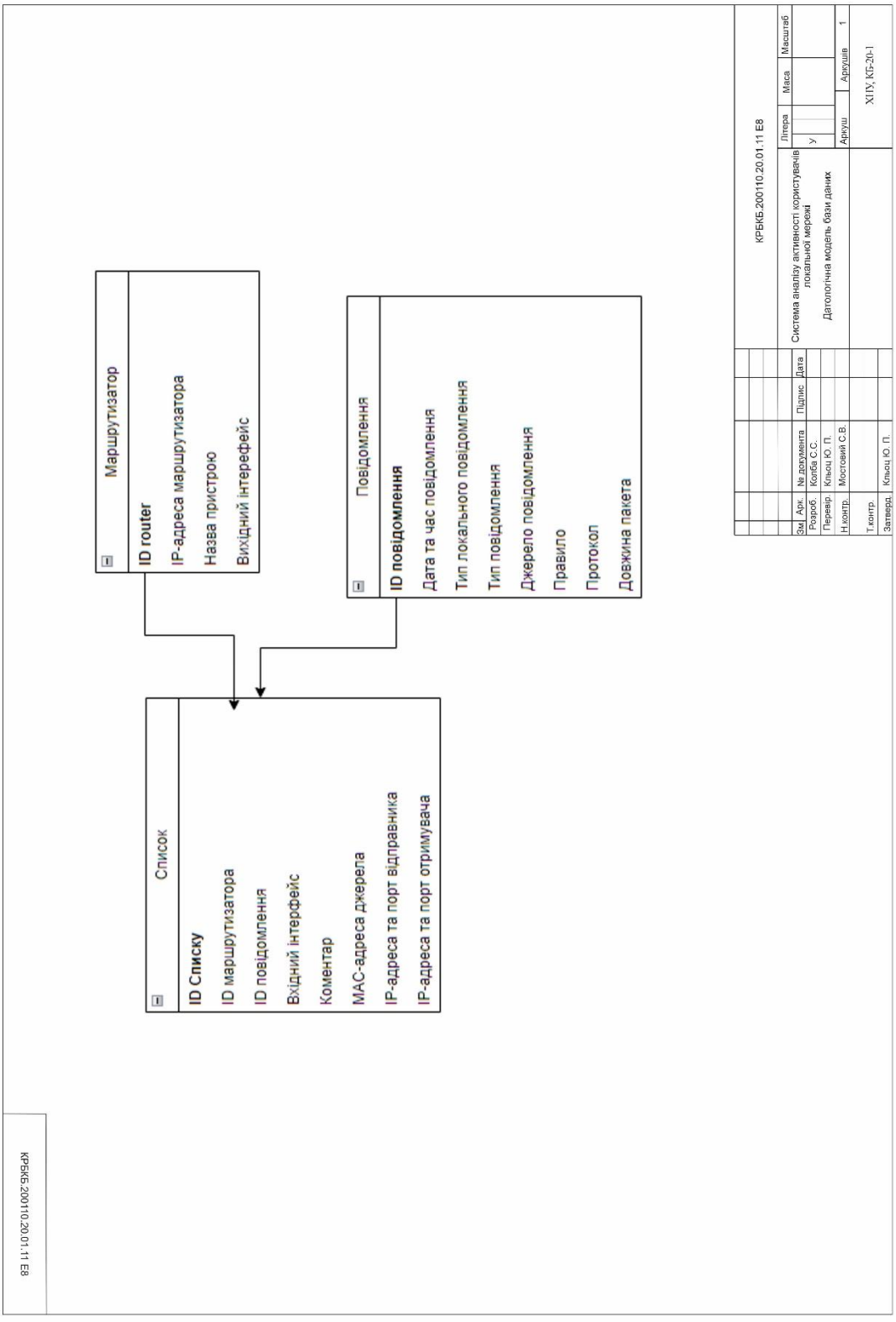
```
import re
from datetime import datetime
import logging

logger = logging.getLogger(__name__)

class LogParser:
    def __init__(self, filepath=None):
        self.filepath = filepath
        self.parsed_data = []

    def parse_log(self, filepath=None):
```

```
        self.filepath = filepath or self.filepath
        if not self.filepath:
            raise ValueError("No file path provided for parsing.")

        current_year = datetime.now().year
        with open(self.filepath, 'r') as file:
            for line in file:
                parsed_line = self._parse_line(line, current_year)
                if parsed_line:
                    self.parsed_data.append(parsed_line)
                    logger.debug(f"Parsed line: {parsed_line}")
                else:
                    logger.warning(f"Failed to parse line: {line}")

    def _parse_line(self, line, current_year):
        pattern = (r'(?P<router_ip>\d+\.\d+\.\d+\.\d+)\t'
                   r'(?P<date>\w+ \d+ \d+:\d+:\d+)\t'
                   r'MikroTik\t(?P<tipe_local_messag>\w+)\t'
                   r'(?P<tipe_messag>\w+)\t'
                   r'(?P<source_messag>\w+)\t'
                   r'(?P<comment>net-usage-log) forward:
in:(?P<in_aiface>[^ ]+) out:(?P<out_iface>[^,]+), '
                   r'src-mac (?P<src_mac>[^,]+), proto
(?P<protocol_messag>[^ ]+) \([^)]+\), '

r'(?P<src_ip>\d+\.\d+\.\d+\.\d+):(?P<src_port>\d+)->(?P<dst_ip>\d+\.\d+\.\d+\
.\d+):(?P<dst_port>\d+), len (?P<length_messag>\d+)')
        match = re.search(pattern, line)
        if match:
            data = match.groupdict()
            data['time_massag'] = datetime.strptime(f"{current_year}
{data['date']}", '%Y %b %d %H:%M:%S')
            return data
        else:
            return None
```

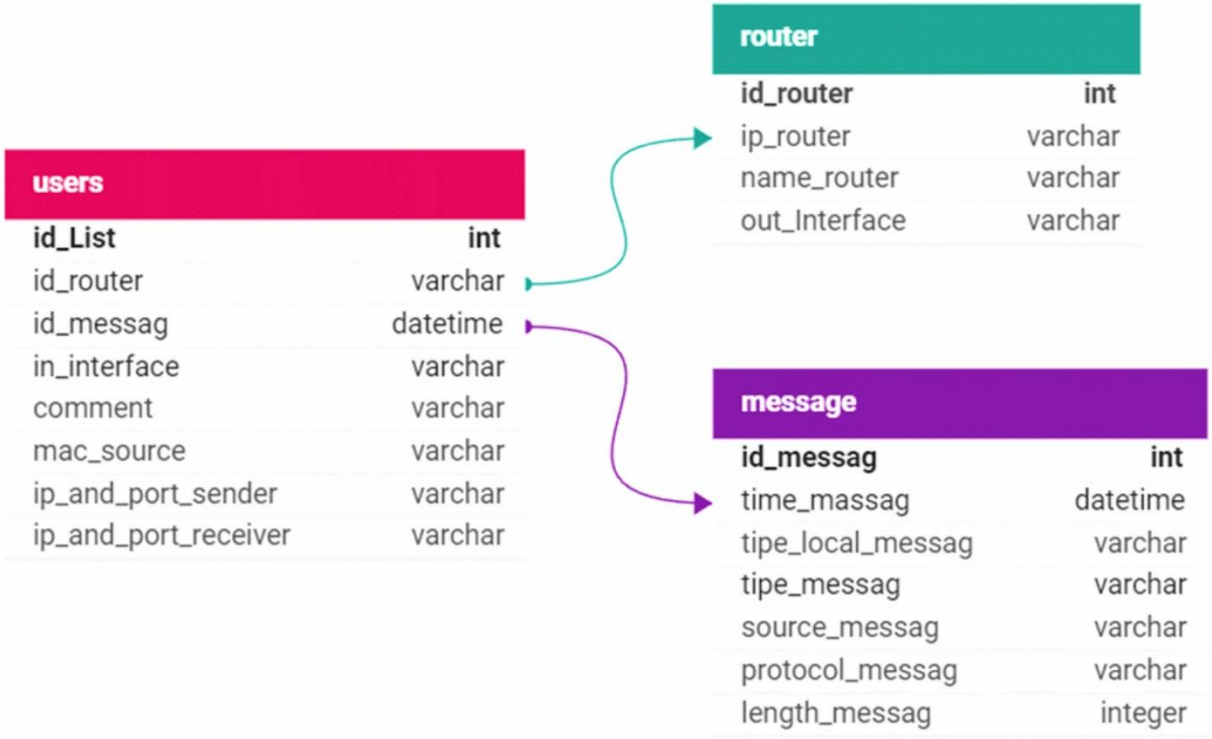
ДОДАТОК Б

(Обов’язковий)

Датологічна модель бази даних



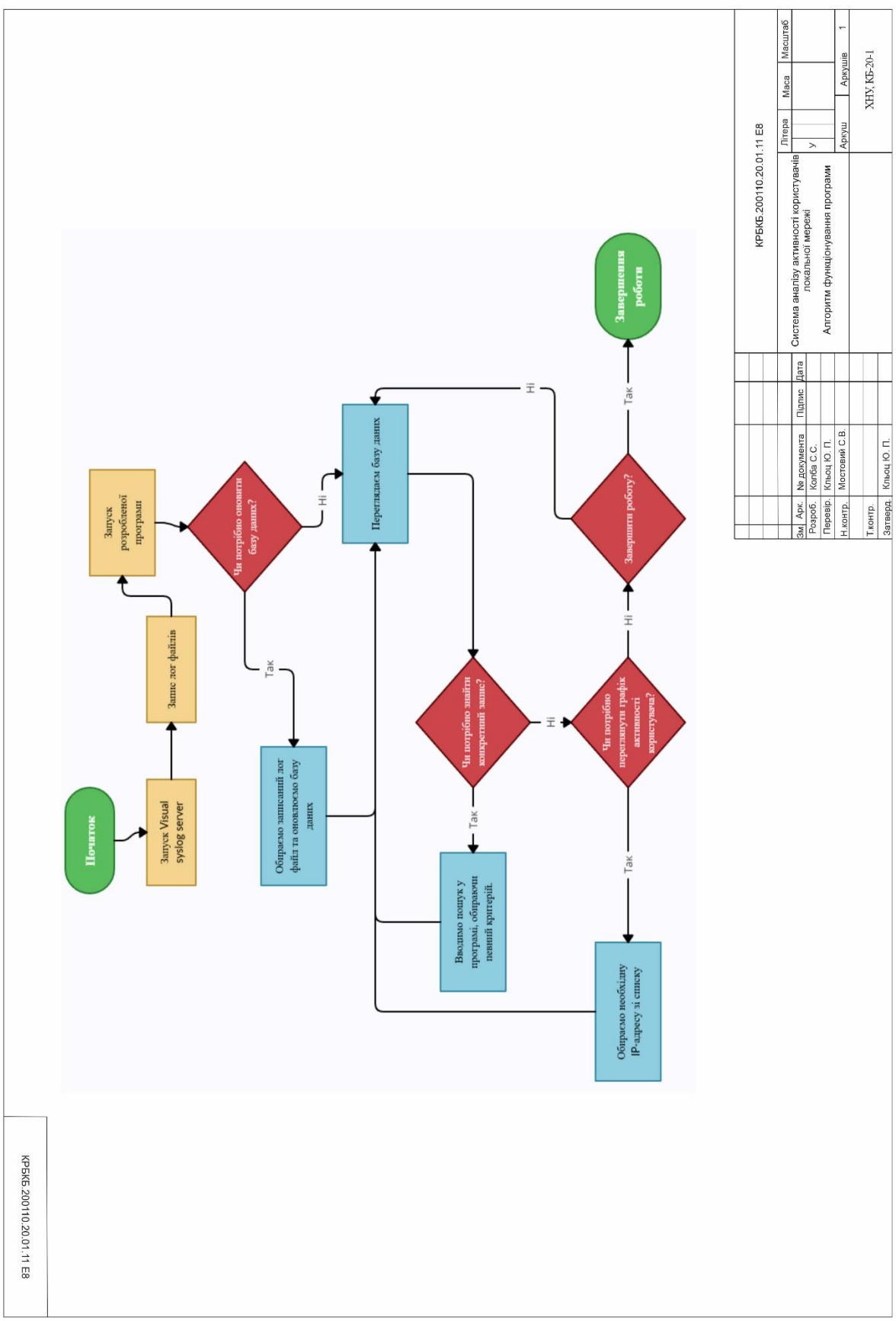
КРБКБ.200110.20.01.11 Е8



					КРБКБ.200110.20.01.11 Е8	
						Літера / Маса / Масштаб
Зм.	Арк.	№ документа	Підпис	Дата	Система аналізу активності користувачів локальної мережі	У
Розроб.		Колба С.С.				
Перевір.		Кльоц Ю. П.			Фізична модель бази даних	
Н.контр.		Мостовий С.В.				Аркуш / Аркушів 1
Т.контр.						ХНУ, КБ-20-1
Затверд.		Кльоц Ю. П.				

Фізична модель бази даних

Алгоритм функціонування програми



Завідувачу кафедри кібербезпеки
к.т.н., доц. Кльоцу Ю.П.
Колби Сергія Сергійовича
ПІБ здобувача вищої освіти

Студента ФІТ, 4 курсу, групи КБ-20-1

ЗАЯВА

З правилами чинного Положення «Про систему забезпечення академічної доброчесності у хмельницькому національному університеті» від 31.08.2023, згідно з яким виявлення плагіату є підставою для відмови в допуску кваліфікаційної роботи до захисту та застосування заходів дисциплінарної та академічної відповідальності, ознайомлений. Про використання програмно-технічних засобів для перевірки кваліфікаційних робіт здобувачів вищої освіти на плагіат оповіщений 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.

1.06.2024 р.
дата

підпис

Anti-Plagiarism v-15.257

Максимальне співпадіння з одним документом 1.0%

Словники перевірки: en_US, ru_RU, ua_UA. **Помилок в документах: 14%**

	Документ		Сумарний збіг по Базі Даних	
ID: 132410 Назва: Система аналізу активності користувачів локальної мережі Додано в БД: 2024-06-24 Автора: Колба С.С. Керівники: Кльоц Ю.П. Консультанти: Опоненти:	Символи	Лексеми	Символи	Лексеми
	53558	817	986 (2%)	17 (2%)

Джерело плагіату

ID	Опис	Наявність плагіату в документі	
		Символи	Лексеми

UNICHECK
by Turnitin

Ім'я користувача:
Кафедра кібербезпеки

ID перевірки:
1016385604

Дата перевірки:
24.06.2024 14:00:52 EEST

Тип перевірки:
Doc vs Internet + Library

Дата звіту:
24.06.2024 14:01:14 EEST

ID користувача:
100008300

Назва документа: **Колба_Зміст і основа222 (на плагіат)**

Кількість сторінок: **57** **Кількість слів:** **9510** **Кількість символів:** **75986** **Розмір файлу:** **1.24 MB** **ID файлу:** **1016196643**

Виявлено модифікації тексту (можуть впливати на відсоток схожості)

2.65%
Схожість

Найбільша схожість: **0.29%** з Інтернет-джерелом (https://uk.wikipedia.org/wiki?curid=3022884)

2.65% Джерела з Інтернету	262	Сторінка 59
0.35% Джерела з Бібліотеки	5	Сторінка 60

0% Цитат

Вилучення цитат вимкнене

Вилучення списку бібліографічних посилань вимкнене

0%
Вилучень

Немає вилучених джерел

Модифікації

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

Замінені символи	3
Підозріле форматування	24 сторінки

РІШЕННЯ ЕКСПЕРНОЇ КОМІСІЇ

КАФЕДРИ КІБЕРБЕЗПЕКИ

ПРО ДОПУСК КВАЛІФІКАЦІЙНОЇ РОБОТИ ДО ЗАХИСТУ

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

Назва: Комплексна система захисту інформації автоматизованого робочого місця керівника підприємства

Автор: Колба Сергій Сергійович

Спеціальність: 125 – Кібербезпека

Освітня програма: освітньо-професійна

Керівник: Кльоц Юрій Павлович, канд. техн. наук, доцент

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:

Оригінальність тексту роботи за результатами перевірки системою Unichek складає 97,35%, оригінальність тексту роботи за результатами перевірки системою Anti-Plagiarism v 15.257 складає 99%.

Згідно з Положенням про систему забезпечення академічної доброчесності у ХНУ (https://khmnu.edu.ua/wp-content/uploads/normatyvni-dokumenty/polozhennya/pro-systemu-zabezpechennya-akademichnoyi-dobrochesnosti.pdf, Додаток В) кваліфікаційна робота, виконана за освітньо-професійною програмою, кількісні показники рівня унікальності тексту у відсотках до загального обсягу матеріалу в якій складає 75-100 %, визнається роботою з високою унікальністю тексту: «Текст вважається унікальним і не потребує додаткових дій щодо запобігання неправомірним запозиченням».

Керівник роботи _____ Юрій КЛЬОЦ

Завідувач кафедри кібербезпеки _____ Юрій КЛЬОЦ

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

РЕЦЕНЗІЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ

освітнього ступеня «бакалавр»

Студент Колба Сергій Сергійович

Тема Система аналізу активності користувачів локальної мережі

Спеціальність 125 – Кібербезпека

Обсяг кваліфікаційної роботи освітньо-кваліфікаційного рівня «бакалавр»:

кількість листів креслень 3; кількість сторінок записки 66.

1. Короткий зміст роботи та прийнятих рішень У кваліфікаційній роботі була розроблена система збору та аналізу активності користувачів в локальній мережі. Ця система збирає трафік користувачів та формує графік їх активності. У процесі розробки були створені база даних та програмне забезпечення. У роботі також розглянуто декілька варіантів для вибору сторонніх програм з метою збору трафіку в мережі.

2. Висновок про відповідність кваліфікаційної роботи завданню У кваліфікаційній роботі було виконано поставлене завдання як у теоретичній, так і в практичній частині.

3. Характеристика виконання кожного розділу роботи, ступінь використання останніх досягнень науки і техніки і передових методів роботи: У вступі роботи наведена загальна характеристика задачі, визначені об'єкти та методи їх дослідження, а також сформульована мета. Окреслено завдання, необхідні для досягнення поставленої мети, проведено аналіз досліджуваної проблеми та розроблено обгрунтований підхід до її вирішення. У першому розділі розглядаються необхідне програмне та апаратне забезпечення, засоби віддаленого збору логів та постанова задачі. Наступний розділ описує поетапну розробку бази даних та програми, а також налаштування маршрутизатора Mikrotik з метою подальшої реалізації всієї системи логування. Останній розділ описує загальний алгоритм функціональності програми та запису трафіку із її перетворенням на графічну частину у вигляді графіка для подальшого аналізу.

4. Позитивні сторони роботи Кваліфікаційна робота має як теоретичну так і практичну цінність. В теоретичній частині вона полягає у розгляді декілька варіантів вибору програм для реалізації поставленої мети. У практичній частині – у розробці системи аналізу активності користувачів в локальній мережі, що дозволяє проаналізувати час активності учасників мережі. Завдяки цьому можливо оптимізувати робочий час, виявляти та усувати проблеми пов'язані із перевантаженням локальної інфраструктури. При проєктуванні системи логування використовувався мережевий маршрутизатор Mikrotik.

5. Негативні сторони роботи: В системі не реалізовано шифрування лог-файлів на випадок викрадення чи витоку цієї інформації, що дозволяє зловмиснику зробити аналіз мережевої активності користувачів у разі успішної атаки.

6. Оцінка графічного оформлення та пояснювальної записки роботи Графічне оформлення кваліфікаційної роботи відповідає темі роботи та виконане з дотриманням стандартів. В цілому, графічне оформлення є якісним, а пояснювальна записка відповідає нормам оформлення.

7. Відгук про роботу в цілому Кваліфікаційна робота заслуговує на позитивну оцінку завдяки структурованому, чіткому та послідовному викладенню матеріалу. Усі розділи роботи логічно впорядковані, що полегшує розуміння представленої інформації в межах теми. Графічний матеріал ефективно ілюструє доцільність прийнятих рішень та їхню результативність у досягненні поставленої мети..

8. Інші зауваження В переліку використаних джерел наявні посилання на популярні ресурси, такі, як Вікіпендія, які не рекомендовано використовувати при написанні кваліфікаціних робіт.

9. Оцінка кваліфікаційної роботи: З урахуванням усіх позитивних і негативних аспектів представленої кваліфікаційної роботи, можна зробити висновок, що вона заслуговує оцінки «добре».

РЕЦЕНЗЕНТ (прізвище, ім'я, по батькові, посада, місце роботи)
Підченко Сергій Костянтинович,
завідувач кафедри ТМІТ, доктор технічних наук, професор

« 18 » 06 2024. (підпис)