МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Центральноукраїнський національний технічний університет

Кафедра кібербезпеки та програмного забезпечення

На правах рукопису

Олійник Артур Олегович

Програмне забезпечення системи аналізу мережевого трафіку

Спеціальність: 123 «Комп'ютерна інженерія»

Освітній ступінь: бакалавр

Науковий керівник:

Доренський Олександр Павлович _____

(підпис) (дата)

кандидат технічних наук

ДОПУЩЕНО ДО ЗАХИСТУ

Завідувач кафедри

_____ О.А. Смірнов

(підпис) ПІБ

«____»_____2021 р.

Кропивницький – 2021

Міністерство освіти і науки України
Центральноукраїнський національний технічний університет

Факультет *Механіко-технологічний*
Кафедра *Кібербезпеки та програмного забезпечення*
Освітній ступінь *бакалавр*
Спеціальність *123 Комп'ютерна інженерія*

ЗАТВЕРДЖУЮ
Завідувач кафедри
д.т.н., проф.
_____О.А.Смірнов
« 11 » січня 2021 року

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ БАКАЛАВРСЬКУ РОБОТУ СТУДЕНТУ

Олійнику Артуру Олеговичу
(прізвище, ім'я, по батькові)

1. Тема роботи *Програмне забезпечення системи аналізу мережевого трафіку*

керівник роботи *Доренський Олександр Павлович, канд. техн. наук*
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу № 204-02 від 28.12.2020 року

2. Строк подання студентом роботи до захисту *22.05.2021 р.*

3. Мета та завдання кваліфікаційної бакалаврської роботи*: Метою розробки є програмне забезпечення системи аналізу мережевого трафіку*

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

1. Призначення та область використання.
2.Перегляд аналогічних існуючих систем.
3. Опис і обґрунтування проектних рішень.
4. Етапи програмування системи.
5. Впровадження системи в промислову експлуатацію.
6. Висновки

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

Структурна схема системи	*1 аркуш*
Функціональна схема системи	*1 аркуш*
Діаграма процесів	*1 аркуш*
Блок-схема алгоритму роботи додатку	*2 аркуша*

6. Дата видачі завдання « 11 » січня 2021 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної бакалаврської роботи	Строк виконання етапів кваліфікаційної бакалаврської роботи	Примітка
1.	Аналіз існуючих систем	10.03.2021 р.	
2.	Постановка задачі, оформлення ТЗ	15.03.2021 р.	
3.	Розробка моделі компонента	20.03.2021 р.	
4.	Розробка структур даних	25.03.2021 р.	
5.	Розробка алгоритмів зв’язку та відображення	30.03.2021 р.	
6.	Програмування алгоритмів	10.04.2021 р.	
7.	Оформлення ПЗ	17.04.2021 р.	
8.	Попередній захист роботи	14.05.2021 р.	

Студент _____ _____
(підпис) (прізвище та ініціали)

Керівник роботи _____ _____
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Олійник А.О. Програмне забезпечення системи аналізу мережевого трафіку. 123 Комп'ютерна інженерія. Центральноукраїнський національний технічний університет. Кропивницький. 2021.

В даній кваліфікаційній бакалаврській розроблено програмне забезпечення, яке призначено для системи аналізу мережевого трафіку.

Метою розробки є програмне забезпечення системи аналізу мережевого трафіку.

Результат роботи – програмна реалізація системи аналізу мережевого трафіку.

В процесі роботи над програмною моделлю виконано аналіз існуючих апаратних та програмних засобів. В повній мірі описані всі компоненти розробленого програмного забезпечення.

Розроблено зручний інтерфейс користувача. Наведені інструкції по роботі з програмними засобами.

Програма може використовуватися на ПЕОМ архітектури IBM PC з ОС Windows XP/Vista/7/8/10.

Програму розроблено в середовищі Embarcadero Delphi.

Ключові слова: комп'ютерна інженерія, мережевий трафік

ABSTRACT

Oliinyk A.O. Network traffic analysis system software. 123 Computer Engineering. Central Ukrainian National Technical University. Kropyvnytskyi. 2021

In this bachelor's qualification the software which is intended for system of the analysis of a network traffic is developed.

The purpose of the development is the software of the net work traffic analysis system.

The result is a software implementation of the network traffic analysis system.

In the process of working on the software model, an analysis of existing hardware and software was performed. All components of the developed software are fully described.

Developed user-friendly interface. Instructions for working with software are given.

The program can be used on an IBM PC with Windows XP / Vista / 7/8/10.

The program was developed in the Embarcadero Delphi environment.

Keywords: computer engineering, network traffic

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ

ІБ	Інформаційна безпека
ІТ	Інформаційні технології
MIB	Інформаційна база даних з керування
NMS	Система керування мережею
NMSs	Network Management Systems
PDU	Одиниця даних протоколу
RMON	Віддалений моніторинг
SCNM	Монітор мережі із власною конфігурацією
SNMP	протокол прикладного рівня, який є частиною протоколу TCP/IP. Протокол простого мережного моніторингу
WREN	Перегляд ресурсів на кінцях мережі

ВСТУП

Актуальність теми. Аналіз трафіку є процесом, важливість якого відома будь-якому IT-професіоналові, незалежно від того, чи працює він у невеликій компанії або у великій корпорації. Адже виявлення й виправлення проблем з мережею – це справжнє мистецтво, яке прямо залежить як від інстинкту самого фахівця, так і від глибини і якості оперуємих їм даних. І аналізатор трафіку є саме тем інструментом, який ці дані надає вам. Обране з розумом рішення для аналізу мережного трафіку може не тільки допомогти вам з'ясувати, як пакети відправляються, ухвалюються й наскільки збережено передаються по вашій мережі, але й дозволить зробити набагато-набагато більше!

Зараз на ринку представлена велика кількість варіацій програмного забезпечення для аналізу мережного трафіку. Причому деякі з них здатні викликати ностальгічні спогади у фахівців «старої школи»; вони використовують термінальний шрифт і інтерфейс командного рядка, і на перший погляд видадуться складними у використанні. Інші рішення, навпаки, – виділяються простотою установки й орієнтовані на аудиторію з візуальним сприйняттям (вони буквально перенасичені різними графіками). Ціновий діапазон цих рішень також досить суттєво відрізняється – від безцінних до рішень з досить дорогою корпоративною ліцензією.

Для того, щоб ви залежно в ід своїх завдань і переваг змогли вибрати краще рішення для аналізу мережного трафіку, представляємо у другому розділі список з найцікавіших з доступних зараз на ринку програмних продуктів для аналізу трафіку, а також короткий огляд вбудованої в них функц іональності для добування, обробки й візуального надання різної мережної інформації. Частина цих функцій у всіх наведених у цьому огляді рішень для аналізу мережного трафіку схожа – вони дозволяють із тим або іншим рівнем деталізації побачити відправлені й отримані мережні пакети, – але практично всі з них мають деякі

характерні риси, які роблять їхніми унікальними при використанні в певних ситуаціях або мережних середовищах. Зрештою, до аналізу мережного трафіку ми прибігаємо тоді, коли в нас з'явилася мер ежна проблема, але ми не можемо швидко звести її до певної машини, пристрою або протоколу, і нас доводиться проводити більш глибокий пошук.

Мета й завдання дослідження. Метою роботи є програмне забезпечення системи аналізу мережевого трафіку.

Для досягнення поставленої мети визначена програма дослідження, що складається з наступних завдань:

– Огляд існуючих систем аналізу мережевого трафіку.

– Дослідження системи аналізу мережевого трафіку.

– Програмна реалізація системи аналізу мережевого трафіку.

Практична цінність отриманих результатів полягає в тому, що розроблені алгоритми дозволяють успішно вирішувати задачі аналізу мережевого трафіку.

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи аналізу мережевого трафіку, є актуальною задачею, яка потребує вирішення у даній кваліфікаційній бакалаврській роботі.

1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ

1.1 Призначення системи

У термінології системних адміністраторів і фахівців з інформаційної безпеки часто зустрічається поняття - «аналізатори трафіку». Під аналізатором трафіку розуміється пристрій або програма, яка перехоплює трафік і потім його аналізує. У сфері ІБ використовується термін «сніффер». Як правило, сніффери «слухають» той трафік, який проходить через мережну карту. Одна з найвідоміших вільно розповсюджуваних програм-сніфферів – це Wireshark. Але хочеться відзначити, що насправді функціонал аналізаторів трафіку вже давно «переріс» проблеми хакингу й інформаційної безпеки. Сучасні комерційні аналізатори випускаються як у вигляді програмних рішень, так і у вигляді апаратних пристроїв і служать для комплексного аналізу продуктивності великих інформаційних мереж, а також користувацьких застосунків.

За допомогою аналізаторів трафіку системний адміністратор вирішує наступні завдання:

- знаходить проблеми в роботі мережі (затримки в передачі інформації) і швидко їх усуває;
- виявляє сторонню активність (несанкціонований доступ зловмисників), атаки на корпоративні ресурси і т.д.;
- «прослуховує мережу»;
- аналізує працездатність користувацьких застосунків;
- збирає статистику.

Існують два основні методи аналізу трафіку:

1. «У режимі реального часу».

2. «Ретроспективний аналіз», який припускає «захват трафіку» і його збереження, а потім вивчення й одержання звітності.

У плані технічних рішень, існує методика аналізу трафіку, заснована на даних маршрутизатора. Тобто використовується певний софт, який збирає дані маршрутизатора, аналізує їх і представляє системному інженерові звітність. Тут можна згадати про Netflow (Cisco), який збирає IP-трафік. Є методики, які не засновані на маршрутизаторах, у цьому випадку встановлюється окреме встаткування й програмне забезпечення, яке й збирає дані з мережі, аналізує їх, надаючи звітність і експертний рішення по певних проблемах.

1.2 Область застосування

Розглянемо область застосування системи. Одним з найважливіших вимог безпеки є підзвітність. Можливість фіксувати діяльність суб'єктів системи, а потім асоціювати їх з індивідуальними ідентифікаторами користувачів дозволяє виявляти порушення безпеки й визначати відповідальних за ці порушення.

Для забезпечення властивості підзвітності в комп'ютерних мережах використовуються різні програмно-апаратні засоби, здатні аналізувати стан і параметри елементів системи. До таких засобів ставляться підсистеми аудита й реєстрації ОС, міжмережеві екрани, системи виявлення й запобігання вторгнень, антивірусні системи, мережні монітори.

У рамках забезпечення підзвітності вирішуються наступні завдання:

– Формування правил відбору підлягаючих реєстрації подій. Це завдання виконує адміністратор мережі. Як правило, до таких подій відносять спроби успішного й неуспішного логічного виходу в систему, запуску програм, доступу до ресурсів, що захищаються, спроби зміни атрибутів об'єктів і повноважень користувачів і ін.

– Селективна реєстрація подій у журналі реєстрації подій, називана також протоколюванням або журналізацією. Журнал – це сукупність хронологічно впорядкованих записів про події, відібрані для реєстрації. Порушники можуть зробити спробу «стерти сліди» своєї злочинної діяльності, тому дані журналу

реєстрації повинні бути надійно захищені від модифікації й руйнування неавторизованими суб'єктами.

– Аналіз накопиченої інформації – аудит. Аудит – по своїй природі є реактивною (а не проактивною) дією – тобто записи стають надбанням фахівця з безпеки вже по закінченні деякого часу після того, як ці події відбулися. У тих випадках, коли аналіз інформації про події проводиться в реальному часі, ми маємо справу із системами виявлення й попередження вторгнень – вони дозволяють припинити атаку раніше, чим вона завдасть великої шкоди.

Аудит – це набір процедур обліку й аналізу всіх подій, що представляють потенційну погрозу для безпеки системи. Аудит дозволяє «шпигувати» за обраними об'єктами й видавати повідомлення тривоги, коли, наприклад, який-небудь рядовий користувач спробує прочитати або модифікувати системний файл. Якщо хтось намагається виконати дії, обрані системою безпеки для моніторингу, то система аудита пише повідомлення в журнал реєстрації, ідентифікуючи користувача.

Системний менеджер може готовити звіти безпеки, які містять інформацію з журналу реєстрації. Для «зверхбезпечних» систем передбачаються аудіо- і відеосигнали тривоги, установлювані на машинах адміністраторів, відповідальних за безпеку. Аудитові повинні підлягати не тільки події, ініційовані користувачами, але й дії адміністраторів мережі, які теж повинні бути підзвітні нарівні з усіма. Тому час від часу аудит повинен проводитися сторонніми організаціями.

При реалізації аудита в більших складних системах підсистем, що полягають із безлічі, іноді необхідно власними засобами протоколювати подій, іноді необхідно прокласти спеціальні зусилля по «синхронізації» журналів реєстрації. Зокрема, оскільки в різних частинах великої системи ті самі об'єкти можуть мати різні імена й, навпаки, різні об'єкти – однакові імена, адміністраторам, можливо, прийде прийняти спеціальну угоду про однозначне іменування об'єктів.

Записи журналу реєстрації подій можуть використовуватися й з метою розслідування якого-небудь уже що відбувся інциденту за допомогою реконструкції послідовності подій. Однак аналіз може виявитися важким завданням уже при тривалості періоду спостережень у кілька днів. Навіть найпростіші журнали подій містять така величезна кількість відомостей, що їх практично неможливо аналізувати «вручну», без спеціальних засобів.

Для обробки й аналізу даних журналу реєстрації можуть бути використані наступні засоби:

– Засобу попередньої обробки даних аудита, призначені для стиску інформації журналу реєстрації за рахунок видалення з нього малоінформативних записів, які тільки створюють непотрібний «шум».

– Засобу виявлення аномальних ситуацій, які використовують один з найефективніших приймань розпізнавання порушень. Він полягає в тому, що постійно відслідковуються середньостатистичні значення параметрів системи, які рівняються з їхніми поточними значеннями. Наприклад, факт входу в систему в годинник, нетипові для режиму роботи якого-небудь співробітника, уже є приводом для видачі попередження адміністраторові.

– Засобу розпізнавання атак по їхніх сигнатурах, тобто по характерних ознаках атаки, що виражаються у вигляді специфічного фрагмента коду, нетипового поведінці, аномально частого звертання до якого-небудь мережного порту комп'ютера. Атака детектується порівнянням послідовності подій у реальній системі з тієї, яка втримується в сигнатурі.

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи аналізу мережевого трафіку, є актуальною задачею, яка потребує вирішення у даній кваліфікаційній бакалаврській роботі.

2 ПЕРЕГЛЯД АНАЛОГІЧНИХ ІСНУЮЧИХ СИСТЕМ

2.1 Огляд існуючих систем, технологій, архітектур, програмних рішень за профілем теми кваліфікаційної бакалаврської роботи

Solarwinds Network Bandwidth Analyzer

Дане рішення позиціонується виробником як програмний пакет із двох продуктів – Network Performance Monitor (базове рішення) і Netflow Traffic Analyzer (модульне розширення). Як заявляється, вони мають схожі, але все-таки одмінні функціональні можливості для аналізу мережного трафіку, що доповнюють один одного при спільному використанні відразу двох продуктів.

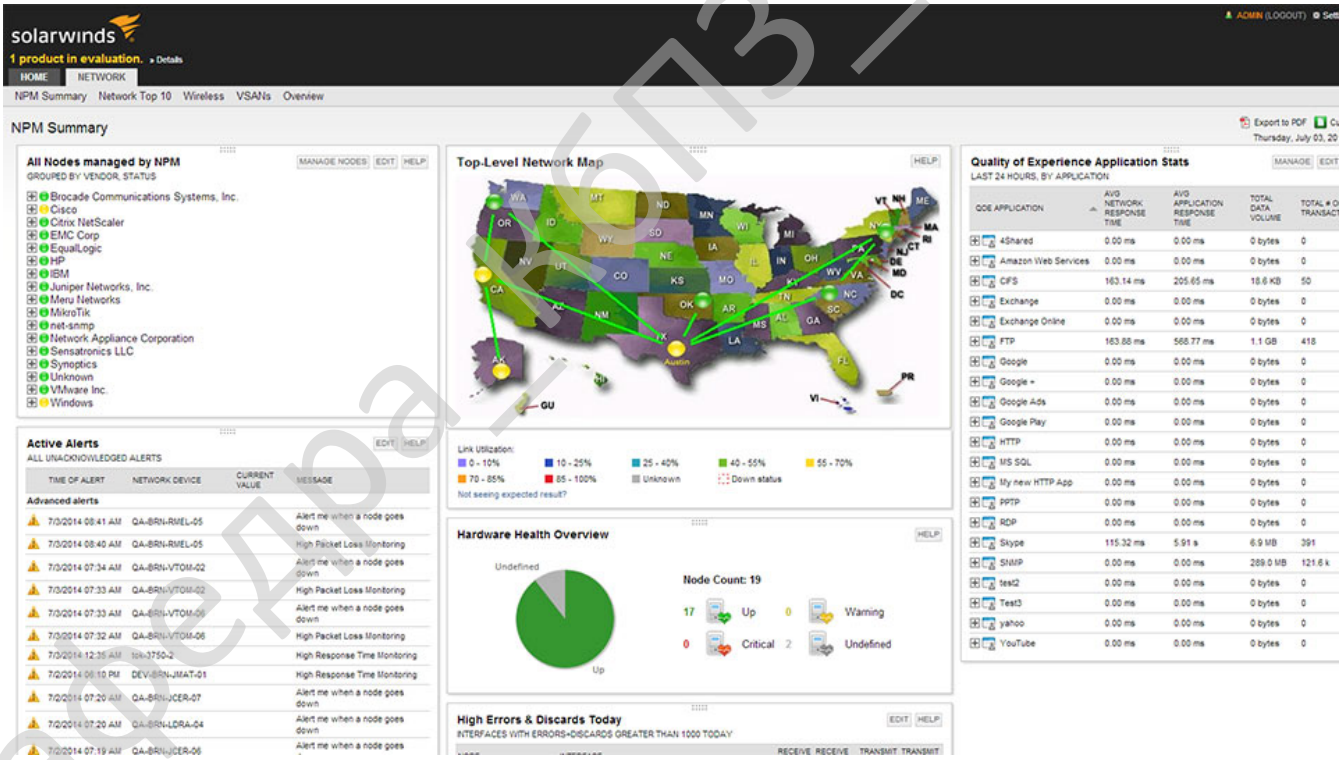


Рисунок 2.1 – Інтерфейс користувача Solarwinds Network Bandwidth Analyzer

Network Performance Monitor, як випливає из назви, здійснює моніторинг продуктивності мережі й стане для вас привабливим вибором, якщо ви прагнете одержати загальну виставу про те, що відбувається у вашій мережі. Купуючи це рішення, ви платите за можливість контролювати загальну працездатність вашої мережі: опираючись на величезну кількість статистичних даних, таких як швидкість і надійність передачі даних і пакетів, у більшості випадків ви зможете швидко ідентифікувати несправності в роботі вашої мережі. А просунуті інтелектуальні можливості програми по виявленню потенційних проблем і широкі можливості по візуальній виставі результатів у вигляді таблиць і графіків із чіткими попередженнями про можливі проблеми, ще більше полегшать цю роботу. Модульне розширення Netflow Traffic Analyzer більше сконцентроване на аналізі самого трафіку. У той час, як функціональність базового програмного розв'язку Network Performance Monitor більше призначена для одержання загальної вистави про продуктивність мережі, в Netflow Traffic Analyzer фокус уваги спрямований на більш детальний аналіз процесів, що відбуваються в мережі. Зокрема, ця частина програмного пакета дозволить проаналізувати перевантаження або аномальні перегони смуги пропущення й надасть статистику, відсортовану по користувачах, протоколах або застосунках. Зверніть увагу, що дана програма доступна тільки для середовища Windows.

Wireshark

Wireshark є відносно новим інструментом у великій родині рішень для мережної діагностики, але за цей час він уже встигнув завоювати собі визнання й повага з боку IT -професіоналів. З аналізом трафіку Wireshark справляється гарно, прекрасно виконуючи для вас свою роботу. Розроблювачі змогли знайти золоту середину між вихідними даними й візуальною представленням цих даних, тому в Wireshark ви не знайдете перекосів у ту або іншу сторону, яким грішать більшість інших рішень для аналізу мережного трафіку. Wireshark простий, сполучимо й портативний. Використовуючи Wireshark, ви одержуєте саме те, що очікуєте, і одержуєте це швидко.

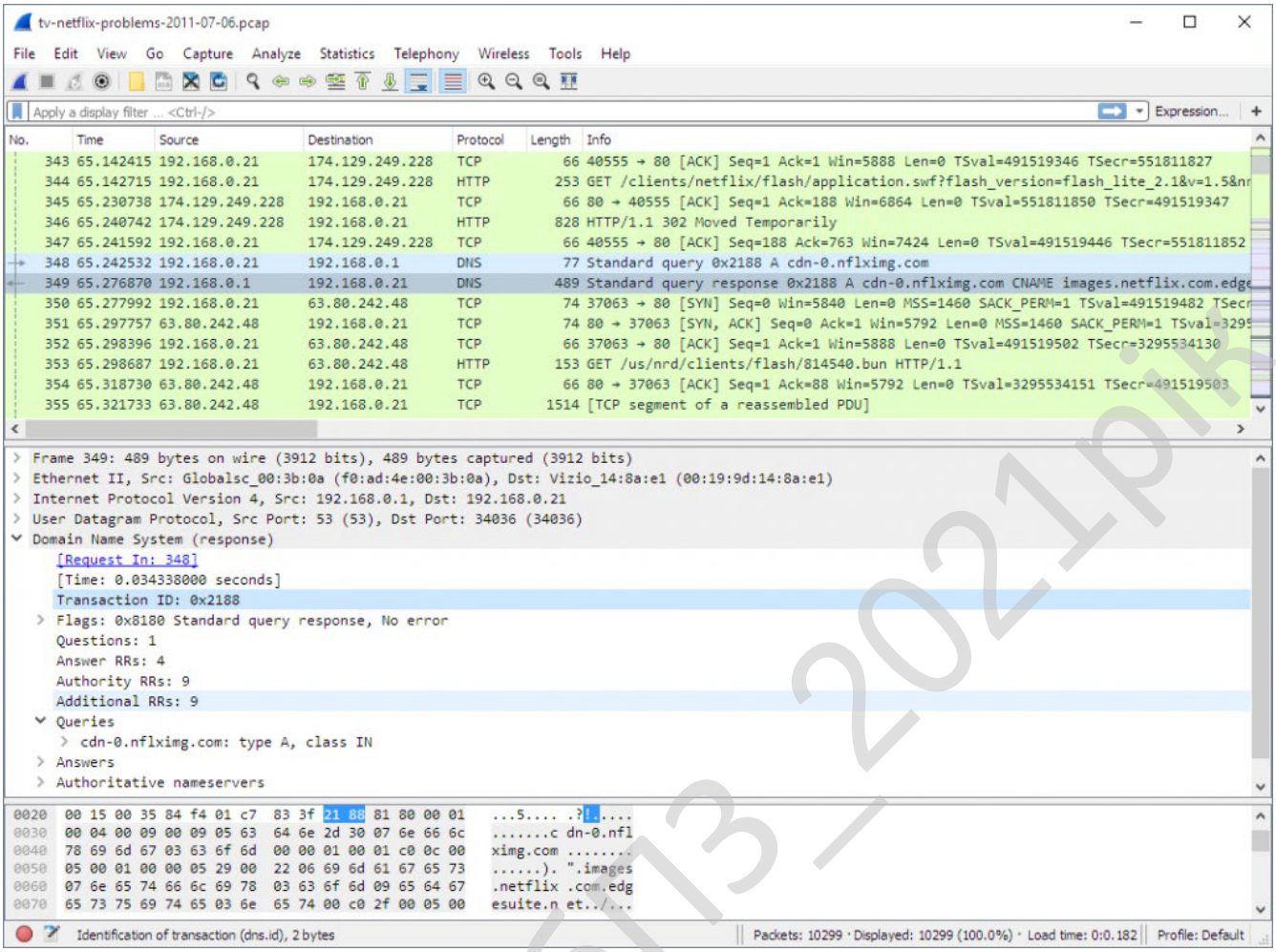


Рисунок 2.2 – Інтерфейс користувача Wireshark

Wireshark має прекрасний користувацький інтерфейс, безліч опцій для фільтрації й сортування, і, що багато з нас зможуть оцінити належним чином, аналіз трафіку Wireshark прекрасно працює з кожним із трьох самих популярних сімейств операційних систем – *NIX, Windows і macOS. Додайте до всього перерахованого вище той факт, що Wireshark – програмний продукт із відкритим вихідним кодом і поширюється безкоштовно, і ви одержите прекрасний інструмент для проведення швидкої діагностики вашої мережі.

Tcpdump

Аналізатор трафіку tcpdump виглядає як якийсь прадавній інструмент, і, якщо вже бути до кінця відвертими, з погляду функціональності працює він також. Незважаючи на те, що зі своєю роботою він справляється й справляється

добре, причому використовуючи для цього мінімум системних ресурсів, наскільки це взагалі можливо, багатьом сучасним фахівцям буде складно розібратися у величезній кількості «сухих» таблиць із даними.

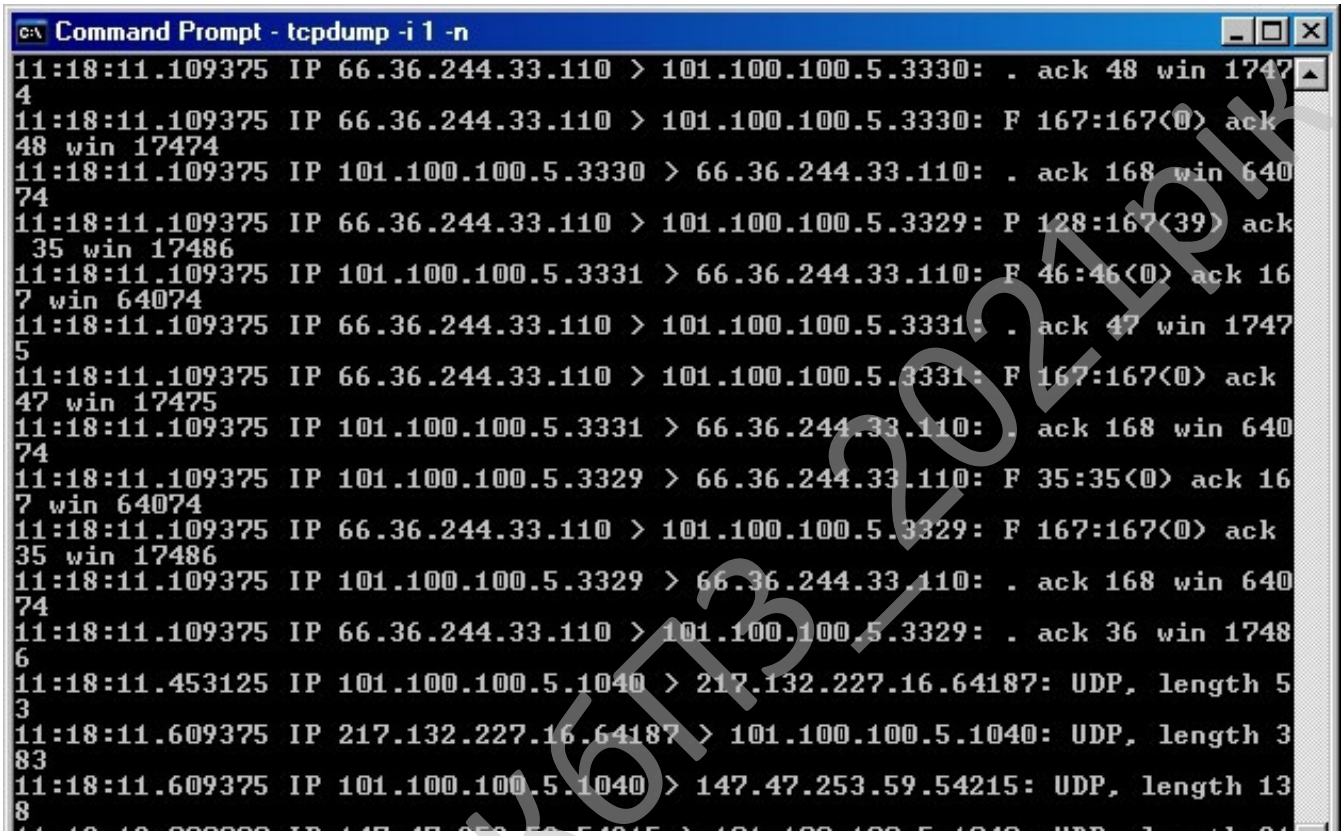


Рисунок 2.3 – Інтерфейс користувача tcpdump

Але бувають у житті ситуації, коли використання настільки обрізаних і невибагливих до ресурсів рішень може бути корисно. У деяких середовищах або на ледве працюючих ПК мінімалізм може виявитися єдиним прийнятним варіантом.

Споконвічно програмне рішення tcpdump розроблене для середовища *NIX, але на даний момент він також працює з декількома портами Windows. Він має всю базову функціональність, яку ви очікуєте побачити в будь-якому аналізаторі трафіку – захват, запис і т.д., – але вимагати чогось більшого від нього не варто.

Kismet

Аналізатор трафіку Kismet – ще один приклад програмного забезпечення з відкритим вихідним кодом, заточеного для розв'язку конкретних завдань. Kismet не просто аналізує мережний трафік, він надає вам набагато більш розширені функціональні можливості. Приміром, він здатний проводити аналіз трафіку схованих мереж і навіть бездротових мереж, які не транслюють свій ідентифікатор SSID! Подібний інструмент для аналізу трафіку може бути надзвичайно корисний, коли у вашій бездротовій мережі є щось, що викликає проблеми, але швидко знайти їхнє джерело ви не можете. Kismet допоможе вам виявити неавторизовану мережу або точку доступу, яка працює, але має не зовсім правильні налаштування.

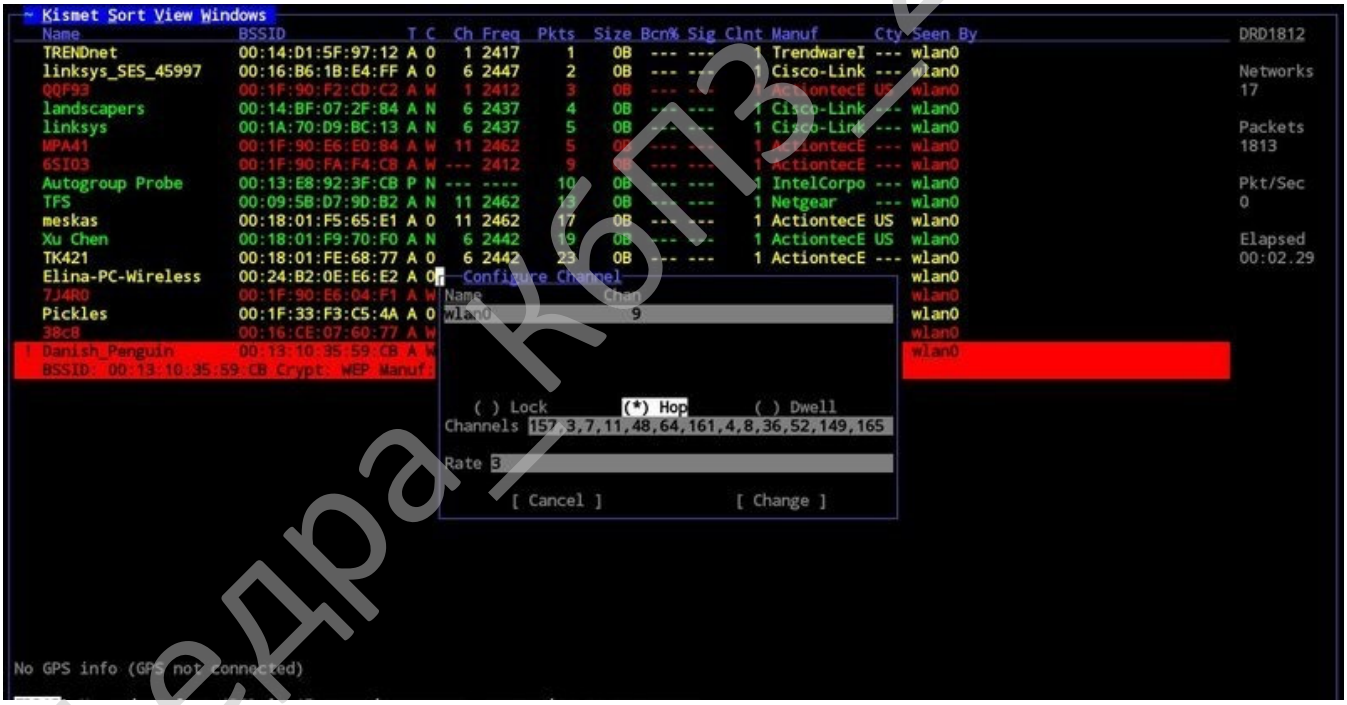


Рисунок 2.4 – Інтерфейс користувача Kismet

Багато з нас знають не з чуток, що завдання стає більш складним, коли справа доходить до аналізу трафіку бездротових мереж, тому наявність під рукою такого спеціалізованого інструмента, як Kismet, не тільки бажане, але й,

найчастіше, необхідно. Аналізатор трафіку Kismet стане прекрасним вибором для вас, якщо ви постійно маєте справу з більшою кількістю бездротового трафіку й бездротових пристроїв, і ви потребуєте гарного інструмента для аналізу трафіку бездротової мережі. Kismet доступний для середовищ * NIX, Windows під Cygwin і macOS.

Etherape

По своїх функціональних можливостях Etherape багато в чому наближається до Wireshark, і він також є програмним забезпеченням з відкритим вихідним кодом і поширюється безкоштовно. Однак те, чому він дійсно виділяється на тлі інших рішень – це орієнтація на графіку.

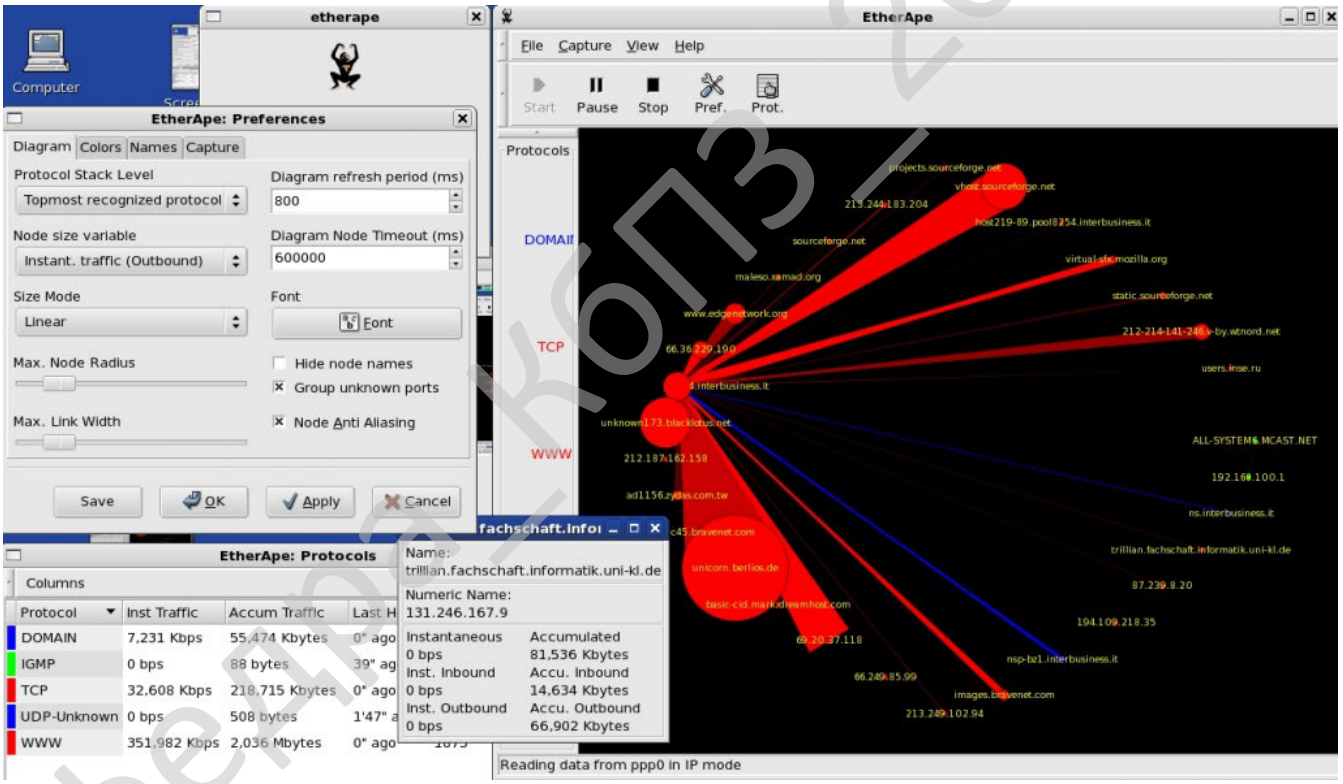


Рисунок 2.5 – Інтерфейс користувача Etherape

І якщо ви, приміром, результати аналізу трафіку Wireshark переглядаєте в класичному цифровому виді, те мережний трафік Etherape відображається за допомогою просунутого графічного інтерфейсу, де кожна вершина графа являє

собою окремий хост, розміри вершин і ребер указують на розмір мережного трафіку, а кольором відзначаються різні протоколи. Для тих людей, хто віддає перевагу візуальному сприйняттю статистичної інформації, аналізатор Etherape може стати кращим вибором. Доступний для середовищ *NIX і macOS.

Cain and Abel

У даного програмного забезпечення з досить цікавою назвою можливість аналізу трафіку є скоріше допоміжною функцією, чому основною. Якщо ваші завдання виходять далеко за межі простого аналізу трафіку, то вам варто звернути увагу на цей інструмент.

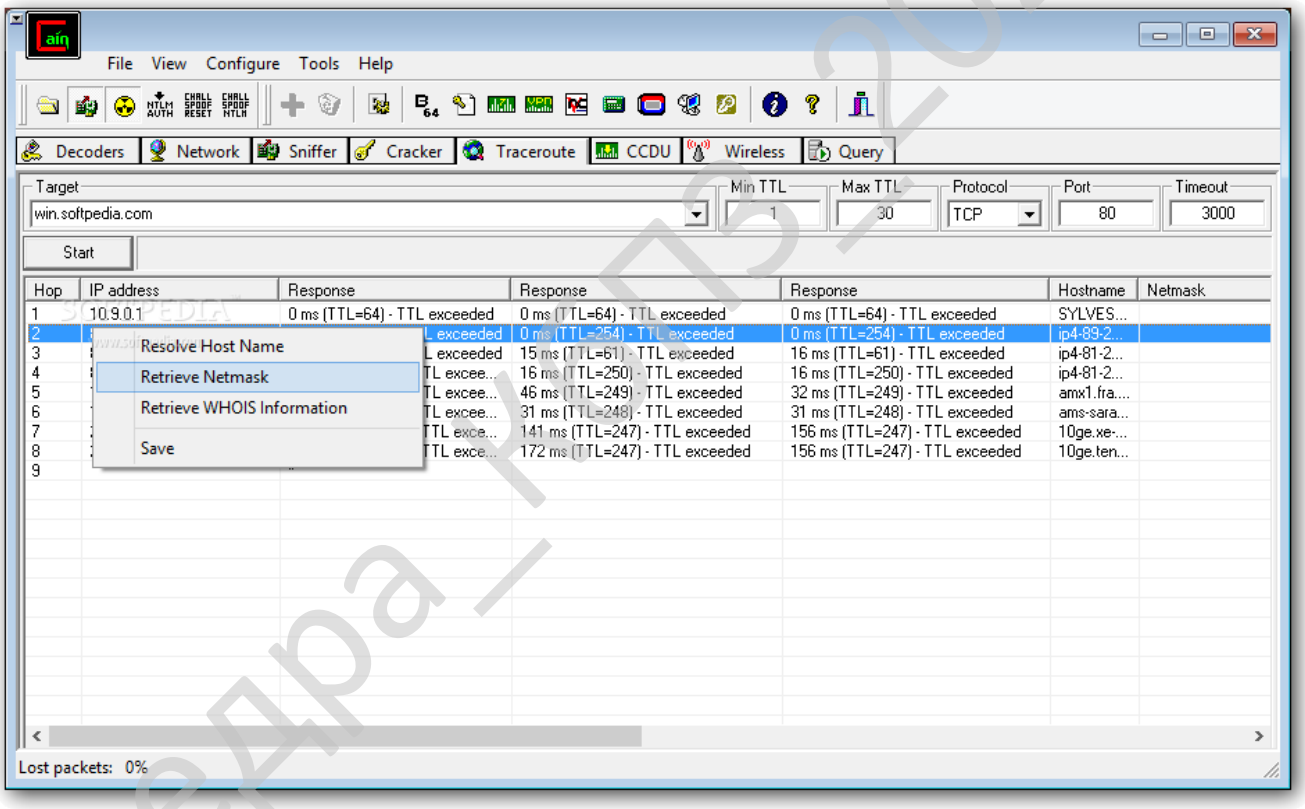


Рисунок 2.6 – Інтерфейс користувача Cain and Abel

З його допомогою ви можете відновлювати паролі для ОС Windows, робити атаки для одержання загублених облікових даних, вивчати дані VoIP у мережі, аналізувати маршрутизацію пакетів і багато чого іншого. Це дійсно

потужний інструментарій для системного адміністратора із широкими повноваженнями. Працює тільки в середовищі Windows.

Networkminer

Рішення Networkminer – ще одне програмне рішення, чия функціональність виходить за рамки звичайного аналізу трафіку. У той час як інші аналізатори трафіку зосереджують своя увага на відправленні й одержанні пакетів, Networkminer стежить за тими, хто безпосередньо здійснює це відправлення й одержання. Цей інструмент більше підходить для виявлення проблемних комп'ютерів або користувачів, чому для проведення загальної діагностики або моніторингу мережі як такий. Networkminer розроблений для ОС Windows.

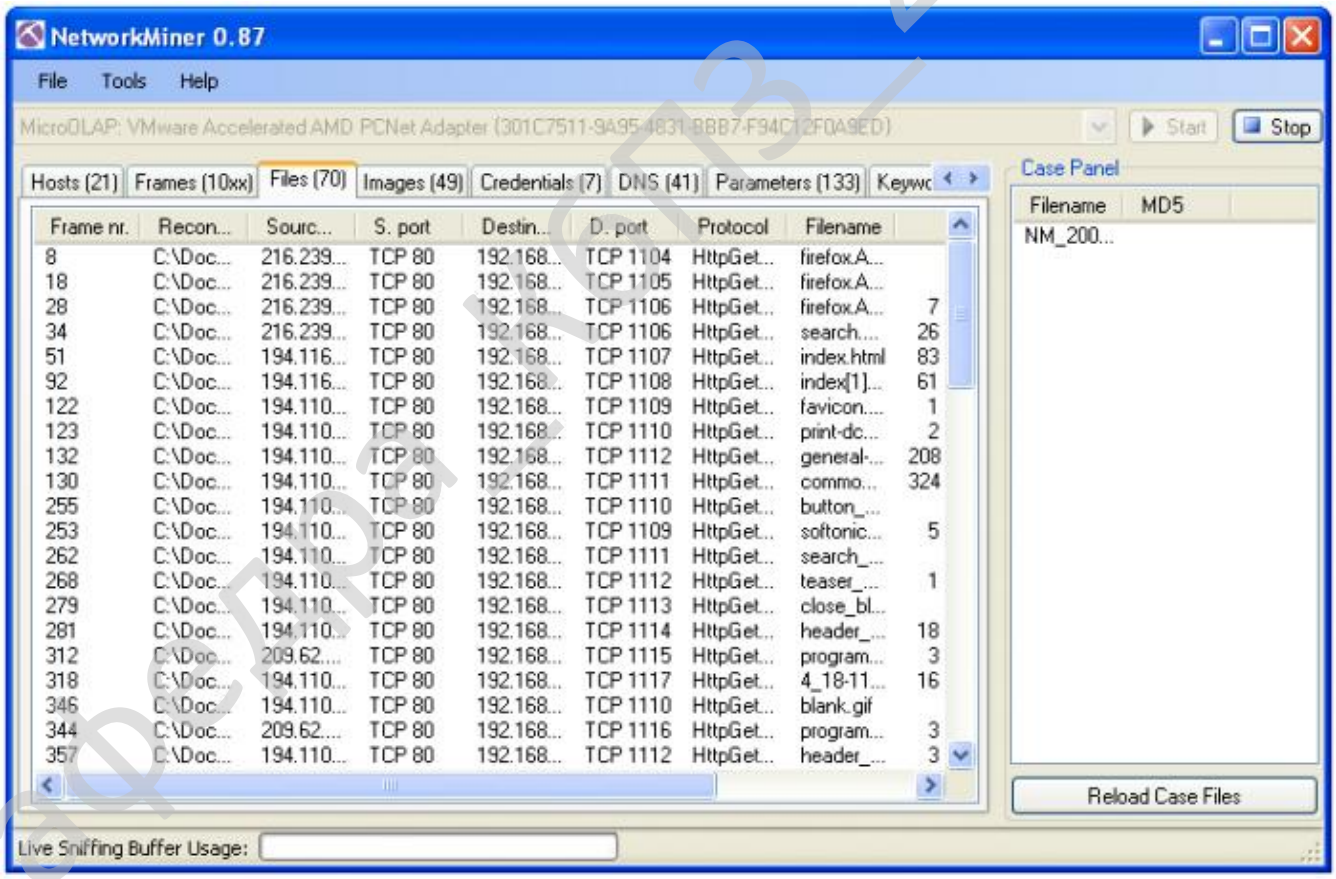
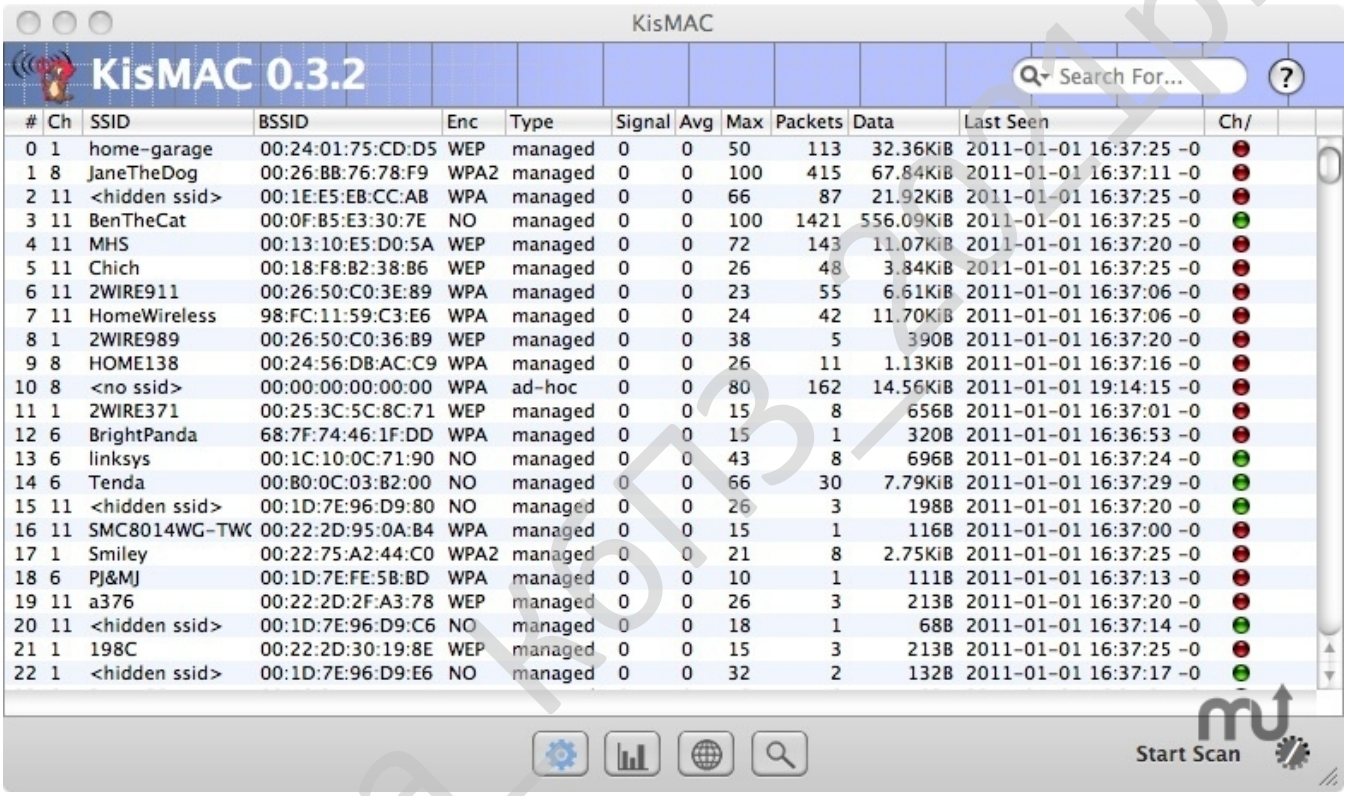


Рисунок 2.7 – Інтерфейс користувача Networkminer

Kismac

Kismac – назва даного програмного продукту говорить саме за себе – це Kismet для macOS. У наші дні Kismet уже має порт для операційного середовища macOS, тому існування Kismac може здатися зайвим, але отут варто звернути увагу на той факт, що рішення Kismac фактично має свою власну кодову базу й не є безпосередньо похідним від аналізатора трафіку Kismet.



#	Ch	SSID	BSSID	Enc	Type	Signal	Avg	Max	Packets	Data	Last Seen	Ch/
0	1	home-garage	00:24:01:75:CD:D5	WEP	managed	0	0	50	113	32.36KiB	2011-01-01 16:37:25 -0	
1	8	JaneTheDog	00:26:BB:76:78:F9	WPA2	managed	0	0	100	415	67.84KiB	2011-01-01 16:37:11 -0	
2	11	<hidden ssid>	00:1E:E5:EB:CC:AB	WPA	managed	0	0	66	87	21.92KiB	2011-01-01 16:37:25 -0	
3	11	BenTheCat	00:0F:B5:E3:30:7E	NO	managed	0	0	100	1421	556.09KiB	2011-01-01 16:37:25 -0	
4	11	MHS	00:13:10:E5:D0:5A	WEP	managed	0	0	72	143	11.07KiB	2011-01-01 16:37:20 -0	
5	11	Chich	00:18:F8:B2:38:B6	WEP	managed	0	0	26	48	3.84KiB	2011-01-01 16:37:25 -0	
6	11	2WIRE911	00:26:50:C0:3E:89	WPA	managed	0	0	23	55	6.61KiB	2011-01-01 16:37:06 -0	
7	11	HomeWireless	98:FC:11:59:C3:E6	WPA	managed	0	0	24	42	11.70KiB	2011-01-01 16:37:06 -0	
8	1	2WIRE989	00:26:50:C0:36:B9	WEP	managed	0	0	38	5	390B	2011-01-01 16:37:20 -0	
9	8	HOME138	00:24:56:DB:AC:C9	WPA	managed	0	0	26	11	1.13KiB	2011-01-01 16:37:16 -0	
10	8	<no ssid>	00:00:00:00:00:00	WPA	ad-hoc	0	0	80	162	14.56KiB	2011-01-01 19:14:15 -0	
11	1	2WIRE371	00:25:3C:5C:8C:71	WEP	managed	0	0	15	8	656B	2011-01-01 16:37:01 -0	
12	6	BrightPanda	68:7F:74:46:1F:DD	WPA	managed	0	0	15	1	320B	2011-01-01 16:36:53 -0	
13	6	linksys	00:1C:10:0C:71:90	NO	managed	0	0	43	8	696B	2011-01-01 16:37:24 -0	
14	6	Tenda	00:B0:0C:03:B2:00	NO	managed	0	0	66	30	7.79KiB	2011-01-01 16:37:29 -0	
15	11	<hidden ssid>	00:1D:7E:96:D9:80	NO	managed	0	0	26	3	198B	2011-01-01 16:37:20 -0	
16	11	SMC8014WG-TWC	00:22:2D:95:0A:B4	WPA	managed	0	0	15	1	116B	2011-01-01 16:37:00 -0	
17	1	Smiley	00:22:75:A2:44:C0	WPA2	managed	0	0	21	8	2.75KiB	2011-01-01 16:37:25 -0	
18	6	PJ&MJ	00:1D:7E:FE:5B:BD	WPA	managed	0	0	10	1	111B	2011-01-01 16:37:13 -0	
19	11	a376	00:22:2D:2F:A3:78	WEP	managed	0	0	26	3	213B	2011-01-01 16:37:20 -0	
20	11	<hidden ssid>	00:1D:7E:96:D9:C6	NO	managed	0	0	18	1	68B	2011-01-01 16:37:14 -0	
21	1	198C	00:22:2D:30:19:8E	WEP	managed	0	0	15	3	213B	2011-01-01 16:37:25 -0	
22	1	<hidden ssid>	00:1D:7E:96:D9:E6	NO	managed	0	0	32	2	132B	2011-01-01 16:37:17 -0	

Рисунок 2.8 – Інтерфейс користувача Kismac

Особливо слід зазначити, що Kismac пропонує деякі можливості, такі як нанесення на карту місця розташування й атака деавтентифікації на macOS, які Kismet сам по собі не надає. Ці унікальні особливості в певних ситуаціях можуть переважити чашу ваг на користь саме цього програмного розв'язку.

Програми для аналізу мережного трафіку можуть стати життєво важливим для вас інструментарієм, коли ви періодично зустрічаєтеся з мережними проблемами різних видів – будь те продуктивність, скинуті з'єднання

або проблеми з мережними резервними копіями. Практично все, що пов'язане з передачею й одержанням даних у мережі, може бути швидко ідентифіковане й виправлене завдяки відомостям, отриманим за допомогою програмного забезпечення з вищенаведеного списку.

Результати, які дасть вам проведений якісний аналіз трафіку мережі за допомогою перевіреного спеціалізованого програмного інструментарію, допоможе вам поглибиться значно нижче верхнього шару проблеми, і зрозуміти, що насправді відбувається у вашій мережі, або не відбувається, але повинне відбуватися.

2.2 Обґрунтування вибору засобів для побудови системи та мови програмування

Embarcadero Delphi, раніше Borland Delphi і Codegear Delphi, – інтегроване середовище розробки ПЗ для Microsoft Windows, Mac OS, iOS і Android мовою Delphi (що раніше носила назву Object Pascal), створена спочатку фірмою Borland і на даний момент приналежна й розроблювальна Embarcadero Technologies. Embarcadero Delphi є частиною пакета Embarcadero RAD Studio і поставляється в чотирьох редакціях: Community (поширюється безкоштовно й має обмежену ліцензію на використання в комерційних цілях), Professional, Enterprise і Architect.

Delphi 10.4 Sydney

Випущено 26 травня 2020 року. RAD Studio Delphi 10.4 забезпечує значно поліпшену високопродуктивну нативну підтримку Windows, кращу продуктивність розробки, миттєві підказки code completion, прискорення виконання коду із синтаксисом керованих записів, поліпшення виконання паралельних завдань на сучасних багатоядерних CPU, а також містить більш 1000 виправлень багів, поліпшення продуктивності середовища й бібліотек і багато чого крім того.

Основні можливості Delphi 10.4.1:

− Істотні розширення для Windows: поліпшення для застосунків на моніторах 4K High DPI, інтеграція з новим WebView2 на базі Chromium, використання розширених title bars, таких же, як в Office, Explorer, Google Chrome.

− Керування пам'яттю в Delphi тепер стандартизоване на всіх підтримуваних платформах – мобільних, настільних і серверних – використовучи класичну реалізацію керування пам'яттю об'єктів.

− Істотне поліпшення Delphi Code Insight (без можливого блокування IDE – в окремому процесі), що допоможе при роботі з великими проектами.

− Тип даних Delphi «record» тепер підтримуть довільні ініціалізацію, фіналізацію й операції копіювання.

− Розширена підтримка бібліотек C++: ZeroMQ, SDL2, SOCI, libSIMDpp і Nematode.

− Відладник Win 64 (на LLDB) і збирач для C++.

− Поліпшення для C++: Включена велика кількість поліпшень STL з Dinkumware.

− Підтримка Metal Driver GPU для macOS і iOS.

− Вбудований Fmxlinux.

− Компонент Twebbrowser для iOS тепер реалізований на Wkwebview API. Реалізація компонента Media Player для macOS тепер використовує Avfoundation. Реалізований заново стилізуємий FMX компонент TMemo на платформі Windows значно поліпшений і тепер має відмінну підтримку IME.

− Численні поліпшення швидкості й стабільності роботи нашої бібліотеки The Parallel Programming Library (PPL).

− Додані оновлені драйвери для FireBird, PostgreSQL і SQLite.

− Клієнтські бібліотеки HTTP і REST Client розширені застосунковими можливостями роботи з HTTPS. Також були розширені можливості підтримки Amazon AWS services

– У технологію Visual LiveBindings внесена безліч поліпшень, у тому числі швидкодії, що стосуться, застосунків на VCL і FireMonkey

RAD Studio 10.4 Короткий огляд:

– Істотні розширення для Windows. Створення застосунків, що чудово виглядають, із чіткими елементами інтерфейсу на 4k моніторах High DPI за допомогою нової гнучкої підтримки стилів елементів керування на екрані. Інтеграція із сучасними, безпечними web-технологіями від Microsoft – новим WebView2 на базі Chromium. Використання сучасних розширених title bars, таких же, як в Office, Explorer, Google Chrome, у своїх проектах. Істотні поліпшення надійності налагодження в новому відладнику для C++ Windows 64-bit.

– Зросла продуктивність розробки. Ріст продуктивності за рахунок миттєвої реакції підказок code completion у середовищі IDE. Краща сумісність із уже наявною кодовою базою, і спрощення програмування за рахунок уніфікованої архітектури керування пам'яттю. Швидке зв'язування даних і візуальних елементів за допомогою розширеної технології Visual LiveBindings з підвищеною швидкодією. Просте використання розповсюджених бібліотек C++, наприклад, ZeroMQ, SDL2, SOCI, libSIMDpp і Nematode. Оновлена підтримка Amazon AWS cloud.

– Поліпшення швидкодії і якості. Більш 1000 поліпшень швидкодії і якості. Краща ефективність коду за допомогою нового синтаксису custom managed records. Більш швидке виконання паралельних завдань на сучасних багатоядерних CPU. Переконаєтеся в прискоренні відображення на екрані з підтримкою Metal API на macOS і iOS. Краща сумісність із уже наявною кодовою базою й спрощення програмування за рахунок уніфікованої архітектури керування пам'яттю.

Істотне поліпшення Delphi Code Insight

Як найбільше й головне поліпшення інструментів програмування Delphi за багато років, в 10.4 Delphi Code Insight реалізований через Language Server Protocol (LSP). LSP – це технологія генерації результатів для code completion,

навігації й інших сервісів в окремому процесі. Це значить, що code completion і Code Insight одержать більш точні результати без блокування IDE. 10.4 забезпечує набагато більш високу продуктивність розроблювачів, які працюють із більшими проектами, що містять мільйони рядків коду.

Delphi Custom Managed Records

Ключове розширення мови Delphi: тип даних Delphi «record» тепер підтримуть довільні ініціалізацію, фіналізацію й операції копіювання. Управляйте тем, як ці структури створюються, копіюються й звільняються з допомогу вашого коду, який буде виконуватися у відповідний момент.

Це розширює потужність конструкцій records в Delphi, які використовуться щоб одержати більшу ефективність у порівнянні із класами.

Єдине керування пам'яттю

Керування пам'яттю в Delphi тепер стандартизоване на всіх підтримуваних платформах – мобільних, настільних і серверних – використовучи класичну реалізацію керування пам'яттю об'єктів.

У порівнянні з Automatic Reference Counting (ARC), це дає кращу сумісність із існучим кодом і спрощує написання компонентів, бібліотек і застосунків.

ARC модель керування пам'яттю model залишилася для керування рядками й посиланнями на тип інтерфейсу на всіх платформах. Для C++ це означає, що при створенні й звільненні Delphi-style класів в C++ використовується звичайне керування пам'яттю, як у будь-якого heap-allocated класу C++, що значно знижує складність коду.

Розширена підтримка бібліотек C++

В 10.4 ми портували багато популярних бібліотек C++ у C++Builder.

Забезпечивши оптимізовану підтримку бібліотек ZeroMQ, SDL2, SOCI, libSIMDpp і Nematode, поряд із уже підтримуваними Boost і Eigen, які можуть бути додані за допомогою менеджера пакетів Getit.

Win 64-відладник і збирач для C++

В 10.4 з'явився новий відладник C++ для Windows 64 -bit. Відладник заснований на LLDB і показує значне збільшення стабільності при налагодженні 64-bit застосунків поряд з новими відладочними можливостями, такими як перегляд і інспекція типів начебто рядків C++ і Delphi, а також колекцій STL, включаючи std::vector, std::map і інших. Крім того, згенерована для застосунку відладочна інформація має інший внутрішній формат, сприяючи більш стабільному й багатому на можливості процесу налагодження, більш докладним перегляду й інспекції в debug-time.

Підвищення якості й швидкодії інструментів

- Велика кількість поліпшень STL від Dinkumware.
- Поліпшені деякі найважливіші методи й області RTL, на базі поліпшень сумісності з популярними бібліотеками C++.
- Поліпшена підтримка Cmake.
- Велика кількість виправлень для підвищення стабільності і якості.
- Відновлення Windows API – Обновлено й додали безліч декларацій API щоб добитися ще більшої інтеграції із платформою Windows.
- Загальні вдосконалення в бібліотеці доступу до БД FireDAC, включаючи оновлені драйвера для FireBird, PostgreSQL і SQLite. Вибір статичного або динамічного підключення SQLite до застосунку.

Змінені стилі VCL для High DPI

В 10.4, архітектура стилізації VCL була суттєво розширена для підтримки High DPI і 4K моніторів. Тепер усі елементи UI на формі VCL автоматично масштабуться під відповідне до монітора дозвіл для показу форми. Був оновлений API стилізації для підтримки стилів high DPI.

Кожний графічний елемент UI може бути обраний з наборів різних масштабів і масштабований до потрібного DPI, що дає чітке зображення елементів UI на всіх моніторах.

Нові High DPI стилі й стилізація окремих VCL компонент

Обновлено велике число вбудованих і преміальних VCL стилів для підтримки нового режиму стилізації High-dpi. Це дозволяє вам створювати застосунку з відмінним дизайном для всіх моніторів.

Розроблювачі VCL застосунків тепер можуть використовувати трохи VCL стилів на різних формах в одному застосунку або в різних компонентів на одній формі. Це також включає стилізацію компонентів загальною темою для платформи. Крім застосункової гнучкості використання стилів, це дозволяє використовувати нестилізуємі компоненти із зовнішніх бібліотек в VCL застосунках, що використовуть стиль.

Поліпшена кроссплатформеність

– Додана підтримка Metal Driver GPU для macOS і iOS.

– Крім підтримки останнього iOS SDK, в RAD Studio 10.4 розроблювачі можуть задовольнити нові вимоги Apple до набору стартових екранів.

– Реалізований заново стилізуємий FMX компонент TMemo на платформі Windows значно поліпшений і тепер має відмінну підтримку IME.

– Користувачам редакцій Enterprise або Architect доступна повна інтеграція Fmxlinux з IDE для створення клієнтських застосунків Linux з GUI.

– Компонент Twebbrowser для iOS тепер реалізований на Wkwebview API.

– Реалізація компонента Media Player для macOS тепер використовує Avfoundation.

Оновлений менеджер пакетів Getit

Менеджер пакетів Getit в IDE був значно вдосконалений.

Дати випуску релізів пакетів тепер видні, і можливе сортування списку по цих датах; відбір тільки встановлених пакетів, контенту, доступного тільки при наявності підписки, багато чого іншого.

Універсальний інсталятор для установки Online і Offline

В 10.4 включений новий універсальний інсталятор, який використовує технологію на базі Getit. Цей інсталятор підтримує як online, так і offline (з ISO) варіанти установки.

Тепер обоє варіанта установки дозволяють вам указати початковий набір можливостей RAD Studio для установки, наприклад, свою комбінацію мов програмування й цільових платформ, мов інтерфейсу, і додавати до нього або видаляти непотрібне в будь-який момент.

2.3 Розгорнута постановка завдання

Згідно з технічним завданням на кваліфікаційну бакалаврську роботу, реалізації підлягає програмне забезпечення, яке призначено для системи аналізу мережевого трафіку.

В процесі розробки кваліфікаційної бакалаврської роботи необхідно виконати наступний обсяг роботи:

а) провести аналіз існуючих систем-аналогів для виявлення їх позитивних і негативних якостей. Результати аналізу врахувати в подальших розробках;

б) вибрати та обґрунтувати методику побудови системи контролю роботи технологічного обладнання на виробництві в автоматизованому режимі. Розробити функціональну та структурну схеми системи;

в) розробити програмне забезпечення системи, що дозволить реалізувати поставлену технічним завданням задачу. Побудувати блок-схеми алгоритмів програми та підпрограми;

г) організувати інтерфейс користувача з метою формування та виводу на екран ЕОМ повідомлень про некоректні дії користувача та нестандартні ситуації в роботі технологічного обладнання;

д) розробити рекомендації по організаційних та методичних заходах, які забезпечать впровадження системи в промислову експлуатацію та її подальшу

успішну експлуатацію;

е) провести розрахунки по визначенню економічної ефективності розробленої системи;

ж) розробити заходи по охороні праці при впровадженні та експлуатації системи, а також розробити заходи з цивільного захисту;

з) сформувати висновки про виконаний обсяг робіт та одержані результати.

3 ОПИС І ОБҐРУНТУВАННЯ ПРОЕКТНИХ РІШЕНЬ

3.1 Опис функціонування системи

Системи аналізу мережного трафіку в багатьох, навіть дуже великих компаніях на практиці ніяк не допомагають запобігти витокам цінних корпоративних даних і виявляються зовсім пошуками при атаках хакерах. Чому так відбувається і як насправді повинен проводитися аналіз трафіку в локальній мережі, щоб від нього була користь? Давайте розбиратися. Але спочатку свіжий кейс із життя дуже великої компанії.

Витік даних в одному з найбільших бюро кредитних історій Equifax привела до того, що персональні дані 143 мільйонів американців виявилися в руках хакерів. Витік даних проходив із середини травня по липень 2017 року. Зловмисники одержали дані до прізвища, імені, номера соціального страхування, даті народження, домашнім адресам і в деяких випадках номеру водійського посвідчення. Також було украдено близько 209 000 номерів кредитних карт і документи по спорах з персональними даними 182 000 людей.

Витік даних відбувся по стандартних каналах зв'язку компанії й не вийшла за її межі в портфелі. Увесь мережний трафік звичайно блокується файєрволами на вході в мережу й системами виявлення витоків, але більшість даних систем працюють із відомими погрозами. Якщо з'являється щось нове, то аналіз трафіку можна зробити на основі раніше збереженого живого трафіку разом з корисним навантаженням. Тому важливо опиратися не тільки на дані систем безпеки, але й мати можливість постійно контролювати мережний трафік на вході й виході мережі й аналізувати звіти по них.

Можна провести аналогію із системами фізичної безпеки, коли камери відеоспостереження постійно записують обстановку в приміщенні або на

території компанії, і все це робиться на додаток до електронних замків, систем доступу, шлагбаумам і т.д.

І у випадку виникнення позаштатної ситуації завжди буде можливість відмотати плівку назад і подивитися – хто, що й коли зробив. Це дозволить зрозуміти, чому в системі виник пролом і що треба зробити, щоб вона не повторилася.

Недоліки існуючих систем аналізу мережного трафіку

У реальному IT світі про це забувають і не приділяють винну увагу системам аналізу живого трафіку, тому ситуація з витоком в Equifax підтве рджує той факт, що повинні бути історичні записи в режимі 24/7 про те, що відбувалося в мережі і як з'явилася можливість такого великого витоку даних.

Дуже часто на початку проекту, коли спілкуєшся із замовниками й говориш про моніторинг, то спливуть асоціації із системами на основі SNMP, IPLSA або Netflow. І як наслідок одержуєш відповідь – у нас усе є й усе контролюється.

Але по своїй суті дані системи не є повноцінними системами аналізу трафіку, які допоможуть. Подібні системи носять тільки статистичний характер і можуть розповісти про стан елемента інфраструктури усередненого за період (SNMP, IPSLA) або використанні каналу зв'язки (Netflow с точністю до хвилини – уже не погано) і можуть служити доповненням до систем пакетного аналізу трафіку.

На основі пакетного аналізу трафіку ви зможете бачити:

- Реальні застосунки, які використовуються в мережі.
- Бачити URL посилання й доменне імена, по яких ходять користувачі.
- Заголовки HTTP і команди, які виконуються.
- Інформація з DHCP і DNS сервісам: IP, MAC адреси й імена хостів.
- Інформація з SMTP протоколу: поштові адреси, тема повідомлення і т.д.
- Інформація з SMB і NFS.
- Вхідний і вихідний трафік не агрегований.

– Дані по GeoIP.

– Підрахунок реального обсягу трафіку.

– Можливість виявлення атак на рівні застосунку.

Основні вимоги до системи аналізу трафіку

Більшість експертів схиляються до того, що файєрволи, системи безпеки й аналіз вхідного й вихідного трафіку повинні бути фізично рознесені. Не можна покладатися на логи з файєрвола, який зберігає всі записи про трафік, так як під час атаки (хакерів або вірусу) він буде недоступний і не надасть ніякої інформації.

Виділена система запису й аналізу трафіку завжди надасть додаткову й коштовну інформацію й найголовніше, що ви зможете аналізувати трафік як у реальному часі, так і за будь-який період (обмежене обсягом сховища й шириною каналу(ів) зв'язку). Це дозволить бачити, як відбувається атака й що відбувається з вихідним трафіком з корпоративної мережі, запобігаючи витік даних.

Основні вимоги, яким повинна відповідати система аналізу трафіку:

– Гарантія запису повного пакета (не метадані).

– Гарантія запису на повній швидкості каналу (наприклад, системи мають 8-10 гігабітний інтерфейс, а реально пишуть дані на швидкості не вище 5-7 Гбіт/сек).

– Можливість зберігання пакетів у часі, не модифікуючи дані (вирізуючи, поєднуючи і т.д.).

– Можливість побудови звітів, трендів, алармів і повідомлень по SNMP, Syslog, e-mail і т.д.

– Можливість розширення сховища для збільшення часу запису трафіку.

– Вибір і пілотування системи повинне відбуватися заздалегідь, а не в момент витоку.

3.2 Розробка структурної схеми

Так як продовжують рости приватні внутрішні мережі компаній, надзвичайно важливо, щоб мережні адміністратори знали й уміли управляти вручну різними типами трафіку, який подорожує по їхній мережі. Моніторинг і аналіз трафіку необхідні для того, щоб більш ефективно діагностувати й вирішувати проблеми, коли вони відбуваються, у такий спосіб не доводячи мережні сервіси до простою в плині тривалого часу. Доступно багато різних інструментів, які дозволяють допомогти адміністраторам з моніторингом і аналізом мережного трафіку. Дана робота обговорює методи моніторингу орієнтовані на маршрутизатори й методи моніторингу не орієнтовані на маршрутизатори (активні й пасивні методи). Робота дає огляд трьох доступні й найбільше широко використовуваних методів моніторингу мережі, вбудованих у маршрутизатори (SNMP, RMON і Cisco Netflow) і надає інформацію про дві нових методах моніторингу, які використовують комбінацію пасивного й активного методів моніторингу (WREN і SCNM).

Мережний моніторинг (моніторинг мережі) – це складне завдання, що вимагає більших витрат сил, яка є життєво важливою частиною роботи мережних адміністраторів. Адміністратори постійно прагнуть підтримати безперебійну роботу своєї мережі. Якщо мережа «упаде» хоча б на невеликий період часу, продуктивність у компанії скоротиться й (у випадку організацій, що надають державні послуги) сама можливість надання основних послуг буде поставлена під погрозу. У зв'язку із цим адміністраторам необхідно стежити за рухом мережного трафіку й продуктивністю на всій мережі й перевіряти, з'явилися чи в ній проломи в безпеці.

Способи моніторингу й аналізу

Аналіз мережі – це процес захвата мережного трафіку і його швидкого перегляду для визначення того, що відбулося з мережею. У наступних підрозділах обговорюються два способи моніторингу мережі: перший –

маршрутизаторо-орієнтований, другий – не орієнтований на маршрутизатори. Функціональність моніторингу, який вбудований у самі маршрутизатори й не вимагає додаткової установки програмного або апаратного забезпечення, називають методами, заснованими на маршрутизаторі. Не засновані на маршрутизаторах методи вимагають установки апаратного й програмного забезпечення й надають більшу гнучкість. Обидві техніки обговорюються нижче у відповідних розділах.

Методи моніторингу засновані на маршрутизаторі

Методи моніторингу засновані на маршрутизаторі – жорстко задані (вшиті) у маршрутизаторах і, отже, мають низьку гнучкість. Короткий опис найбільш частий використовуваних методів такого моніторингу наведені нижче. Кожний метод розбудовувався багато років перш ніж стати стандартизованим способом моніторингу.

Протокол простого мережного моніторингу (SNMP), RFC 1157

SNMP – протокол прикладного рівня, який є частиною протоколу TCP/IP. Він дозволяє адміністраторам керувати продуктивністю мережі, знаходити й усувати мережні проблеми, планувати ріст мережі. Він збирає статистику по трафіку до кінцевого хоста через пасивні датчики, які реалізуються разом з маршрутизатором. У той час, як існують дві версії (SNMPv1 і SNMPv2), даний розділ описує тільки SNMPv1. SNMPv2 побудований на SNMPv1 і пропонує ряд удосконалень, таких як додавання операцій із протоколами. Стандартизується ще один варіант версії SNMP. Версія 3 (SNMPv3) перебуває в стадії розгляду.

Для протоколу SNMP властиві три ключові компоненти: керовані пристрої (Managed Devices), агенти (Agents) і системи керування мережею (Network Management Systems – NMSs).

Керовані пристрої містять у собі SNMP-агента й можуть складатися з маршрутизаторів, перемикачів, комутаторів, концентраторів, персональних комп'ютерів, принтерів і інших елементів, подібних цим. Вони відповідають за збір інформації й роблять її доступної для системи керування мережею (NMS).

Агенти містять у собі програмне забезпечення, яке володіє інформацією з керування, і переводять цю інформацію у форму, сумісну з SNMP. Вони закриті для пристрою керування.

Системи керування мережею (NMS) виконують застосунки, які займаються моніторингом і контролем пристроїв керування. Ресурси процесора й пам'яті, які необхідні для керування мережею, надаються NMS. Для будь-якої керованої мережі повинна бути створена хоча б одна система керування. SNMP може діяти винятково як NMS, або агент, або може виконувати свої обов'язки або ін.

Існує 4 основних команди, що використовуються SNMP NMS для моніторингу й контролю керованих пристроїв: читання, запис, переривання й операції перетинання. Операція читання розглядає змінні, які зберігаються керованими пристроями. Команда запису міняє значення змінних, які зберігаються керованими пристроями. Операції перетинання володіють інформацією про те, які змінні керованих пристроїв підтримують, і збирають інформацію з підтримуваних таблиць змінних. Операція переривання використовується керованими пристроями для того, щоб повідомити NMS про настання певних подій.

SNMP використовує 4 протокольні операції в порядку дії: Get, GetNext, Set і Trap. Команда Get використовується, коли NMS видає запит на інформацію для керованих пристроїв. SNMPv1-запит складається із заголовка повідомлення й одиниці даних протоколу (PDU). Pdu-повідомлення містить інформацію, яка необхідна для вдалого виконання запиту, який буде або одержувати інформацію від агента, або задавати значення в агентові. Керований пристрій використовує SNMP агентів, розташованих у ньому, для одержання необхідної інформації й потім посилає повідомлення NMS'у, з відповіддю на запит. Якщо агент не володіє який або інформацією з відношення до запиту, він нічого не повертає. Команда GetNext буде одержувати значення наступного екземпляра об'єкта. Для NMS також можливо надсилати запит (операція Set), коли встановлюється значення

елементів без агентів. Коли агент повинен повідомити NMS-події, він буде використовувати операцію Trap.

Як говорилося раніше, SNMP – протокол рівня застосунків, який використовує пасивні сенсори, щоб допомогти адміністраторові простежити за мережним трафіком і продуктивністю мережі. Хоча, SNMP може бути корисним інструментом для мережного адміністратора, він створює можливість для погрози безпеки, так як він не має змоги автентифікації. Він відрізняється від віддаленого моніторингу (RMON), який обговорюється в наступному розділі, тим, що RMON працює на мережному рівні й нижче, а не на прикладному.

Віддалений моніторинг (RMON), RFS 1757

RMON містить у собі різні мережні монітори й консольні системи для зміни даних, отриманих у ході моніторингу мережі. Це розширення для SNMP інформаційної бази даних з керування (MIB). У відмінності від SNMP, який повинен посилати запити про надання інформації, RMON може налаштовувати сигнали, які будуть «мониторить» мережу, засновану на певному критерії. RMON надає адміністраторам можливості управляти локальними мережами також добре, як віддаленими від однієї певної локації/точки. Його монітори для мережного рівня наведені нижче. RMON має дві версії RMON і RMON2. Однак у даній роботі говориться тільки про RMON. RMON2 дозволяє проводити моніторинг на всіх мережних рівнях. Він фокусується на IP-трафіку й трафіку прикладного рівня.

Хоча існує 3 ключових компонента мон іторингового середовища RMON, тут приводяться тільки два з них.

Два компоненти RMON це датчик, також відомий як агент або монітор, і клієнт, також відомий як керуюча станція (станція керування). У відмінності від SNMP датчик або агент RMON збирає й зберігає мережну інформацію. Датчик – це вбудоване в мережний пристрій (наприклад маршрутизатор або перемикач) програмне забезпечення. Датчик може запускатися також і на персональному комп'ютері. Датчик повинен міститися для кожного різного сегмента локальної

або глобальної мережі, так як вони здатні бачити трафік, який проходить тільки через їхні канали, але вони не знають про трафік за їхній боковий вівтарями.
Клієнт – це звичайно керуюча станція, яка пов'язана з датчиком, що використовують SNMP для одержання й корекції Rmon-даних.

RMON використовує 9 різних груп моніторингу для одержання інформації про мережу.

– Statistics – статистика обмірювана датчиком для кожного інтерфейсу моніторингу для даного пристрою.

– History – облік періодичних статистичних вибірок з мережі й зберігання їх для пошуку.

– Alarm – періодично бере статистичні зразки й порівнює їх з набором граничних значень для генерації події.

– Host – містить статистичні дані, пов'язані з кожним хостом, виявленим у мережі.

– Hosttopn – готовить таблиці, які описують вершину хостів (головний хост).

– Filters – включає фільтрацію пакетів, ґрунтуючись на фільтровому рівнянні для захвата подій.

– Packet capture – захват пакетів після їхнього проходження через канал.

– Events – контроль генерації й реєстрація подій від пристрою.

– Token ring – підтримка кільцевих лексем.

Як установлено вище, RMON, будується на протоколі SNMP. Хоча моніторинг трафіку може бути виконаний за допомогою цього методу, аналітичні дані про інформацію, отримані SNMP і RMON мають низьку продуктивність.
Утиліта Netflow, яка обговорюється в наступному розділі, працює успішно з багатьма пакетами аналітичного програмного забезпечення, щоб зробити роботу адміністратора набагато простіше.

Netflow, RFS 3954

Netflow – це розширення, яке було презентовано в маршрутизаторах Cisco, які надають можливість збирати IP мережний трафік, якщо це задане в інтерфейсі. Аналізуючи дані, які надаються Netflow, мережний адміністратор може визначити такі речі як: джерело й приймач трафіку, клас сервісу, причини переповненості. Netflow містить у собі 3 компонента: Flowcaching (кешуючий потік), Flowcollector (збирач інформації про потоки) і Data Analyzer (аналізатор даних).

Flowcaching аналізує й збирає дані про IP потоках, які входять в інтерфейс, і перетворить дані для експорту.

З Netflow-пакетів може бути отримана наступна інформація:

– Адреса джерела й одержувача.
– Номер вхідного й вихідного пристрою.
– Номер порту джерела й приймача.
– Протокол 4 рівня.
– Кількість пакетів у потоці.
– Кількість байтів у потоці.
– Часовий штамп у потоці.
– Номер автономної системи (AS) джерела й приймача.
– Тип сервісу (ToS) і прапор TCP.

Перший пакет потоку, що проходить через стандартний шлях перемикання, обробляється для створення кешу. Пакети з подібними характеристиками потоку використовуються для створення запису про потік, яка міститься в кеш для всіх активних потоків. Цей запис відзначає кількість пакетів і кількість байт у кожному потоці. К ешуєма інформація потім періодично експортується в Flow Collector (збирач потоків).

Flow Collector – відповідальний за збір, фільтрування й зберігання даних. Він містить у собі історію про інформацію про потоки, які були підключені за

допомогою інтерфейсу. Зниження обсягу даних також відбувається за допомогою Flow Collector'a за допомогою обраних фільтрів і агрегації.

Data Analyzer (аналізатор даних) необхідний, коли потрібно представити дані. Зібрані дані можуть використовуватися для різних цілей, навіть відмінних від моніторингу мережі, таких як планування, облік і побудова мережі.

Перевага Netflow над іншими способами моніторингу, такими як SNMP і RMON, у тому, що в ній існує програмні пакети, призначені для різного аналізу трафіку, які існують для одержання даних від Netflow-пакетів і вистави їх у більш дружелюбному для користувача виді.

При використанні інструментів, таких як Netflow Analyzer (це тільки один інструмент, який доступний для аналізування Netflow-пакетів), інформація, наведена вище, може бути отримана від Netflow-пакетів для створення діаграм і звичайних графіків, які адміністратор може вивчити для біль шого розуміння про його мережі. Найбільша перевага використання Netflow у відмінності від доступних аналітичних пакетів у тому, що в цьому випадку можуть бути побудовані численні графіки, що описують активність мережі в будь-який момент часу.

Технології не засновані на маршрутизаторах

Хоча технології, не вбудовані в маршрутизатор все-таки обмежені у своїх можливостях, вони пропонують більшу гнучкість, чому технології вбудовані в маршрутизатори. Ці методи класифікуються як активні й пасивні.

Активний моніторинг

Активний моніторинг повідомляє проблеми в мережі, збираючи виміру між двома кінцевими точками. Система активного виміру має справа з такими метриками, як: корисність, маршрутизатори/маршрути, затримка пакетів, повтор пакетів, втрати пакетів, нестійка синхронізація між прибуттям, вимір пропускної здатності.

Головним чином використання інструментів, такі як команда ping, яка вимірює затримку й втрати пакетів, і traceroute, яка допомагає визначити

топологію мережі, являє приклад основних активних інструментів виміру. Обоє ці інструмента посилають пробні ICMP-пакети до точки призначення й чекають, коли ця точка відповість відправникові. Команда ping використовує активний спосіб виміру, посилаючи Echo-запит від джерела через мережу у встановлену точку. Потім одержувач посилає Echo-запит назад джерелу від якого прийшов запит.

Даний метод може не тільки збирати одиничні метрики про активний вимір, але й може визначати топологію мережі. Ще один важливий приклад активного виміру – утиліта iperf. Iperf – це утиліта, яка вимірює якість пропускної здатності TCP і UDP протоколів. Вона повідомляє пропускну здатність каналу, що існує затримку й втрати пакетів.

Проблема, яка існує з активним моніторингом, – це те, що представлені проби в мережі можуть втручатися в нормальний трафік. Частий час активних проб обробляється інакше, чому нормальний трафік, що ставить під питання значимість наданої інформації від цих проб.

Згідно із загальною інформацією, описаною вище, активний моніторинг – це надзвичайно рідкий метод моніторингу, узятий окремо. Пасивний моніторинг напроти не вимагає більших мережних витрат.

Пасивний моніторинг

Пасивний моніторинг у відмінності від активного не додає трафік у мережу й не змінює трафік, який уже існує в мережі. Також у відмінності від активного моніторингу, пасивний збирає інформацію тільки про одну точку в мережі. Виміру відбуваються набагато краще, чим між двома точками, при активному моніторингу. Рис. 5 показує установку системи пасивного моніторингу, де монітор розміщений на одиничному каналі між двома кінцевими точками й спостерігає трафік коли той проходить по каналу.

Пасивні виміри мають справу з такою інформацією, як: трафік і суміш протоколів, кількість бітів (бітрейт), синхронізація пакетів і час між прибуттям.

Пасивний моніторинг може бути здійснений, за допомогою будь-якої програми, що витягає пакети.

Хоча пасивний моніторинг не має витрат, які має активний моніторинг, він має свої недоліки. З пасивним моніторингом, виміри можуть бути проаналізовані тільки оф-лайн і вони не представляють колекцію. Це створює проблему, пов'язану з обробкою більших наборів даних, які зібрані під час виміру.

Пасивний моніторинг може бути краще активного в тому, що дані службових сигналів не додаються в мережу, але пост-обробка може викликати велика кількість тимчасових витрат. От чому існує комбінація цих двох методів моніторингу.

Комбінований моніторинг

Після прочитання розділів вище, можна благополучно переходити до висновку про те, що комбінування активного й пасивного моніторингу – кращий спосіб, чому використання першого або другого окремо. Комбіновані технології використовують кращі сторони й пасивного, і активного моніторингу середовищ. Дві нові технології, що представляють комбіновані технології моніторингу, описуються нижче. Це «Перегляд ресурсів на кінцях мережі» (WREN) і «Монітор мережі із власною конфігурацією» (SCNM).

Перегляд ресурсів на кінцях мережі (WREN)

WREN використовує комбінацію технік активного й пасивного моніторингу, активно обробляючи дані, коли трафік малий, і пасивно обробляючи дані протягом часу великого трафіку. Він дивиться трафік і від джерела, і від одержувача, що уможливлює більш акуратні виміри. WREN використовує трасування пакетів від створеного додатком трафіку для виміру корисної пропускної здатності. WREN розбитий на два рівні: основний рівень швидкої обробки пакетів і аналізатор трасувань користувацького рівня.

Основний рівень швидкої обробки пакетів відповідає за одержання інформації, пов'язаної із вхідними й вихідними пакетами. Показує список

інформації, яка збирається для кожного пакета. До Web100 додається буфер для збору цих характеристик. Доступ до буфера здійснюється за допомогою двох системних викликів. Один виклик починає трасування й надає необхідну інформацію для її збору, поки другий виклик повертає трасування з ядра.

Об'єкт трасування пакетів – здатний координувати обчислення між різними машинами. Одна машина буде активувати роботу іншої машини, задаючи прапор у заголовку пакета, що йде, для початку обробки деякого діапазону пакетів, які вона трасує. Інша машина буде у свою чергу трасувати всі пакети, для яких вона бачить, що в заголовку встановлений схожий прапор. Така координація забезпечує те, що інформація про схожі пакети зберігається в кожній кінцевій точці незалежно від зв'язку й того, що відбувається між ними.

Аналізатор трасувань користувацького рівня – інший рівень у середовищі WREN. Це компонент, який починає трасування будь-якого пакета, збирає й обробляє повернуті дані на рівні ядра оператора. Згідно із проектуванням, компоненти користувацького рівня не мають потреби в читанні інформації від об'єкта трасування пакетів увесь час. Вони можуть бути проаналізовані негайно після того, як трасування буде завершено, щоб зробити висновок у реальному часі, або дані можуть бути збережені для подальшого аналізу.

Коли трафік малий, WREN буде активно вводити трафік у мережу зберігаючи порядок проходження потоків виміру. Після численних досліджень, знайдене, що WREN представляє схожі виміри в перенасичені й у не-перенасичених середовищах.

У поточній реалізації WREN, користувачі не принуждаються тільки до захвата трасувань, які були ініційовані ними. Хоча будь-який користувач може стежити за трафіком застосунків інших користувачів, вони обмежені в інформації, яка може бути отримана від трасувань інших користувачів. Вони можуть тільки одержати послідовність і підтвердження чисел, але не можуть одержати актуальні сегменти даних з пакетів.

Загалом, WREN – це дуже корисна установка, яка використовує переваги й активного, і пасивного моніторингу. Хоча ця технологія перебуває на ранньому етапі розвитку, WREN може надати адміністраторам корисні ресурси в моніторингу й аналізі їх мереж. Монітор Власного конфігурування мережі (SCNM) – інший інструментарій, який використовує технології й активного, і пасивного моніторингу.

Мережний монітор із власною конфігурацією (SCNM)

SCNM – це інструмент моніторингу, який використовує зв'язок пасивних і активних вимірів для збору інформації на 3 рівні проникнення маршрутизаторів, що виходять, і інших важливих точок моніторингу мережі. Середовище SCNM включає й апаратний, і програмний компонент.

Апаратний засіб установлюється в критичних точках мережі. Воно відповідає за пасивний збір заголовків пакетів. Програмне забезпечення запускається на кінцевій точці мережі.

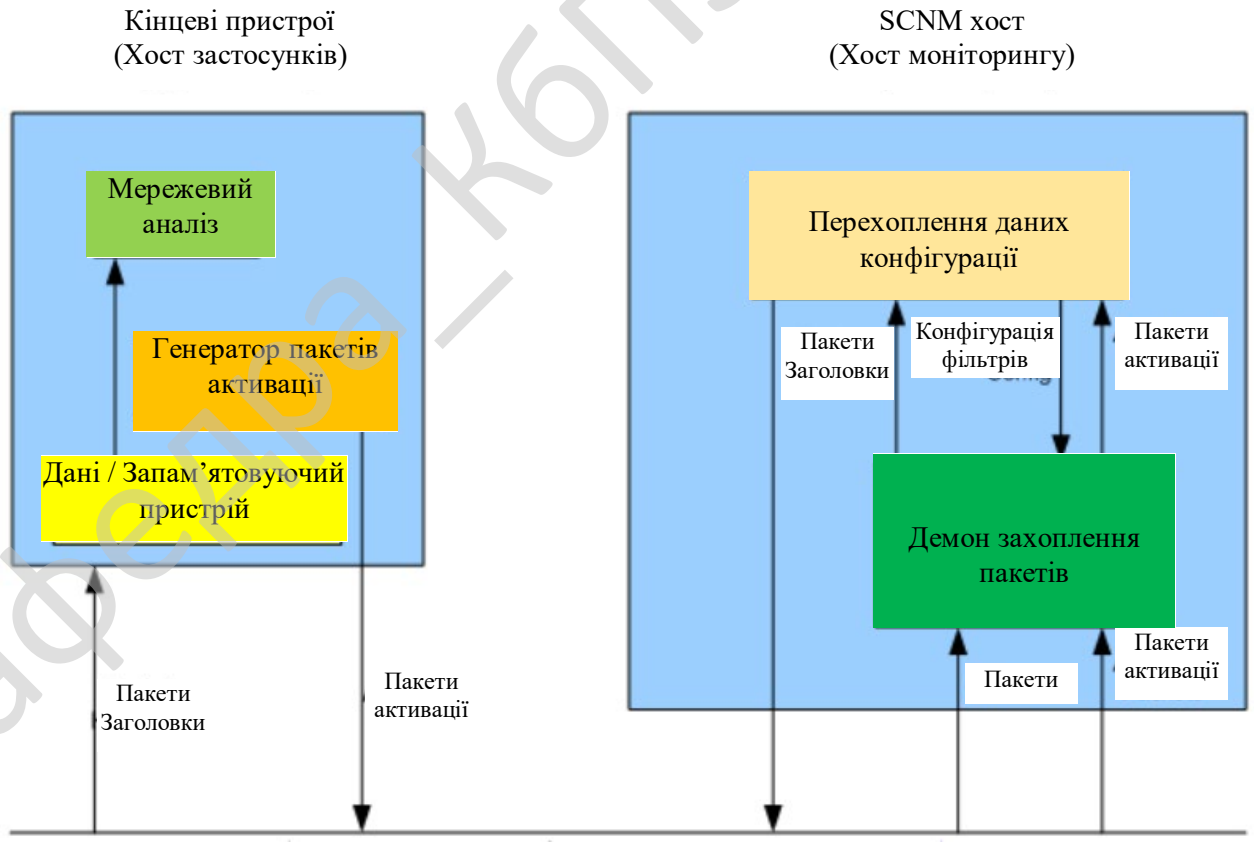


Рисунок 3.1 – Структурна схема системи

Програмне забезпечення відповідає за створення й посилку активованих пакетів, які використовуються для старту моніторингу мережі. Користувачі будуть посилати в мережу пакети активації, що містять деталі про пакети, які вони прагнуть одержати для моніторингу й збору. Користувачі не мають потреби в знанні місця розташування SCNM-хосту, ухвалюючи за істину те, що всі хости відкриті для «прослушки» пакетів. На основі інформації, яка існує в рамках активаційного пакета, фільтр міститься в потік збору даних, який також працює в кінцевій точці. Збираються заголовки пакетів мережного й транспортного рівня, які відповідають фільтру. Фільтр буде автоматично введений у тайм аут, після точно заданого часу, якщо він одержує інші пакети застосунку. Служба вибірки пакетів, яка запускається на SCNM-хості, використовує команду tcpdump (подібно програмі вибірки пакетів) у порядку отриманих запитів і записи трафіку, який відповідає запиту.

Коли інструментами пасивного моніторингу визначається проблема, трафік може бути згенеровані за допомогою інструментів активного моніторингу, дозволяючи збирати дані, що додаються, для більш детального вивчення проблеми. При розгортанні цього монітора в мережі на кожному маршрутизаторі протягом шляху, ми може вивчати тільки секції мережі, які мають проблеми.

SCNM призначений для установки й використання, головним чином, адміністраторами. Проте звичайні користувачі можуть використовувати деяку частину цієї функціональності. Хоча звичайні користувачі здатні використовувати частини середовища SCNM моніторингу, їм дозволено дивитися тільки свої власні дані.

На закінчення скажемо, що SCNM – це ще один спосіб комбінованого моніторингу, який використовує й активний, і пасивний методи, щоб допомогти адміністраторам мониторити і аналізувати їхні мережі.

Підбираючи частки інструменти для використання їх у моніторингу мережі, адміністратор повинен спочатку розв'язати, чи прагне він використовувати системи, що добре зарекомендували себе, які вже

використовувалися багато років, або нові. Якщо існуючі системи більш підходящий рішення, тоді Netflow – найбільш корисний інструмент для використання, так як у зв'язування із цією утилітою можуть використовуватися пакети, що аналізуються, даних для вистави даних у більш дружелюбному користувачеві виді. Проте, якщо адміністратор готовий спробувати нову систему, рішення комбінованого моніторингу, такі як WREN або SCNM , – кращий напрямок для подальшої роботи.

Спостереження й аналіз мережі – життєво необхідні в роботі системного адміністратора. Адміністратори повинні намагатися містити свою мережу в порядку, як для не розрізненої продуктивності усередині компанії, так і для зв'язку з будь-якими існуючими публічними сервісами. Згідно з вищеописаною інформацією, деяке число маршрутизаторо-орієнтованих технологій і не засновані на маршрутизаторах, придатні для допомоги мережним адміністраторам у щоденному моніторингу й аналізі їх мереж. Тут коротко описуються SNMP, RMON, і Cisco's Netflow – приклад декількох технологій, заснованих на маршрутизаторах. Приклади не заснованих на маршрутизаторах технологій, які обговорювалися в роботі, – це активний, пасивний моніторинг і їх комбінація.

3.3 Розробка функціональної схеми

Мережний аналізатор, може використовуватися для аналізу трафіку, що проходить через мережний інтерфейс вашого комп'ютера. Він може знадобитися для виявлення й вирішення проблем з мережею, налагодження ваших веб-застосунків, мережних програм або сайтів. Мережний аналізатор дозволяє повністю переглядати вміст пакета на всіх рівнях: так ви зможете краще зрозуміти як працює мережа на низькому рівні.

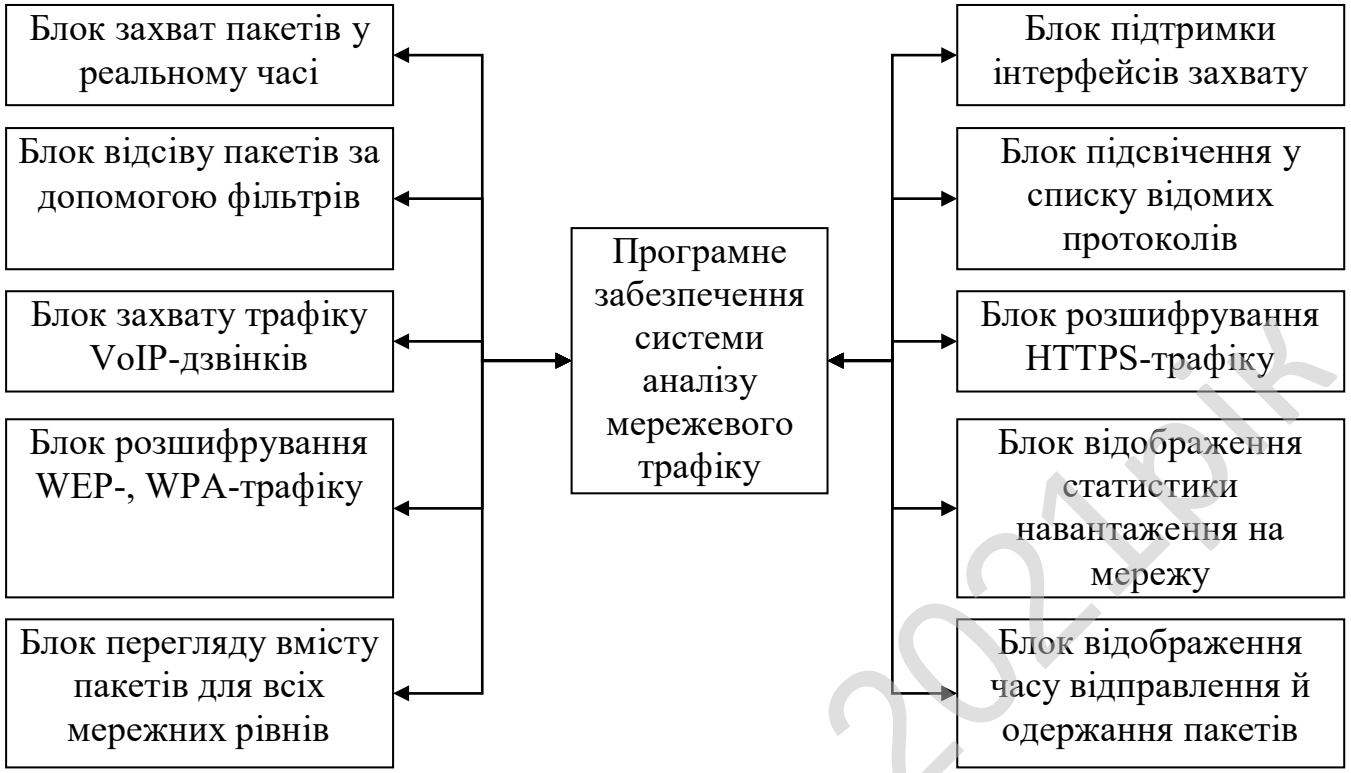


Рисунок 3.2 – Функціональна схема системи

Усі пакети перехоплюються в реальному часі й надаються в зручному для читання форматі. Програма підтримує дуже потужну систему фільтрації, підсвічування кольором, і інші особливості, які допоможуть знайти потрібні пакети.

Перед тем, як переходити до розгляду способів аналізу трафіку, потрібно розглянути, які можливості підтримує програма більш докладно, з якими протоколами вона може працювати й що робити. От основні можливості програми:

– Захват пакетів у реальному часі із дротового або будь-якого іншого типу мережних інтерфейсів, а також читання з файлу.

– Підтримуються такі інтерфейси захвата: Ethernet, IEEE 802.11, PPP і локальні віртуальні інтерфейси.

– Пакети можна відсівати за множиною параметрів за допомогою фільтрів.

– Усі відомі протоколи підсвічуються у списку різними кольорами, наприклад TCP, HTTP, FTP, DNS, ICMP і так далі.

– Підтримка захвата трафіку VoIP-дзвінків.

– Підтримується розшифрування HTTPS-трафіку при наявності сертифіката.

– Розшифрування WEP-, WPA-трафіку бездротових мереж при наявності ключа й handshake.

– Відображення статистики навантаження на мережу.

– Перегляд вмісту пакетів для всіх мережних рівнів.

– Відображення часу відправлення й одержання пакетів.

Програма має безліч інших функцій, але це були ті основні, які можуть зацікавити.

3.4 Розробка діаграми процесів

Відповідно до методичних рекомендацій розроблення графічної частини кваліфікаційної бакалаврської роботи розглянемо розроблену діаграму процесів яка зображена на рисунку 3.3.

Розроблена діаграма взаємодії процесів використовується для представлення та візуалізації процесів обробки даних тобто структурного проектування бакалаврської роботи.

Основні складові елементи діаграми взаємодії процесів це потоки даних:

– Репозиторії, потік сховища даних.

– Потоки зовнішні по відношенню до системи сутності.

– Процеси які являють собою трансформацію даних в рамках описуваної системи.

– Потоки даних гібридні між елементами трьох попередніх типів.

Відповідно до документації основна будова діаграми процесів полягає у графічному представленні складу сукупностей даних, що характеризуються як

співвідношення різних частин кожної з сукупностей. Склад статистичної сукупності графічно може бути представлений як за допомогою абсолютних, так і відносних показників. Графічне зображення складу сукупності по абсолютними і відносними показниками сприяє проведенню більш глибокого аналізу і дозволяє проводити аналіз системи.

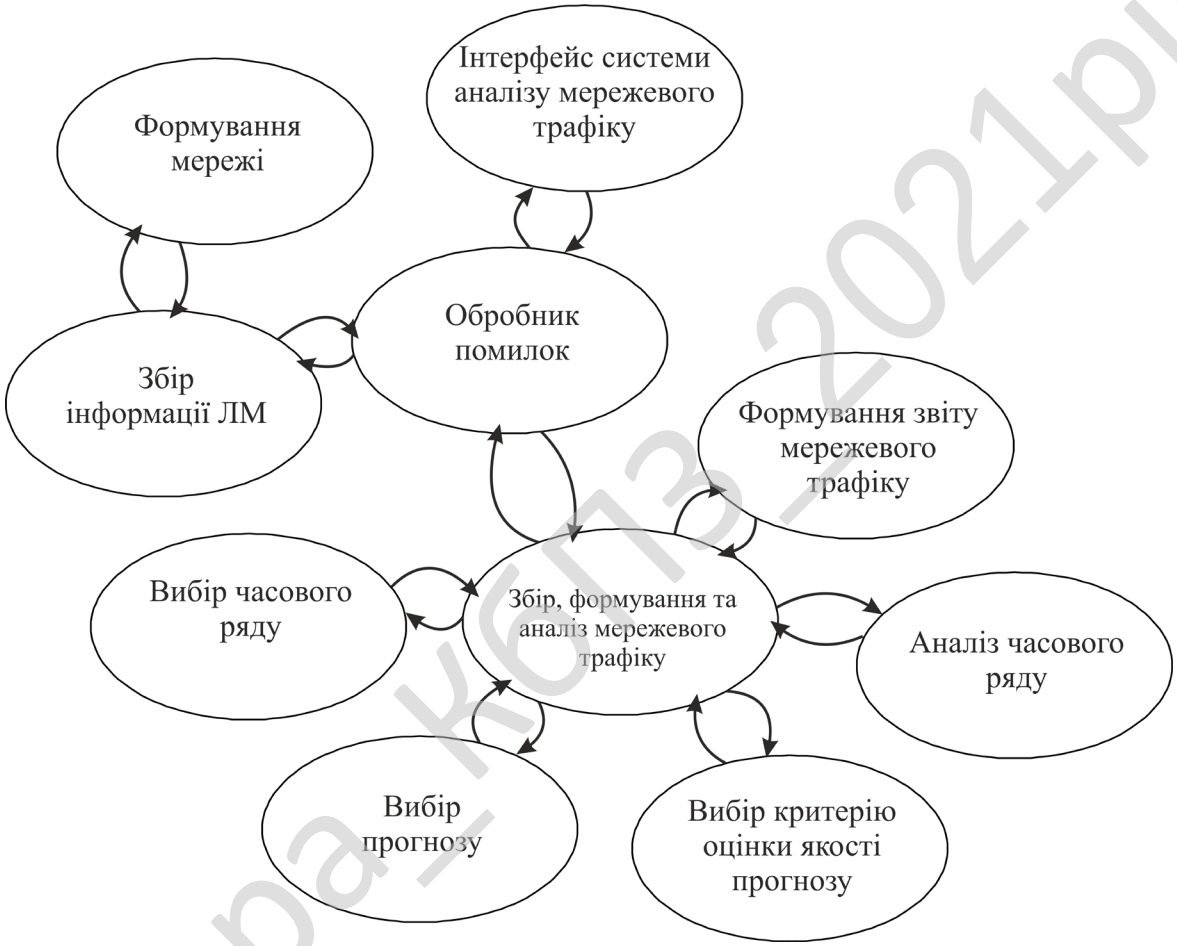


Рисунок 3.3 – Діаграма взаємодії процесів

Для схематичного представлення системи що розробляється необхідно спочатку представити діаграму взаємодії процесів даних рівня контексту, завдяки чому буде показано взаємодію системи в цілому у подальшому. Використовується модель проектування, графічне представлення «потоків» даних в інформаційній системі.

Розроблена діаграма взаємодії процесів системи в подальшому уточнюється шляхом деталізації процесів та потоків даних з метою показати систему що розробляється.

Таким чином у результаті після розгляду, вищеописаної системи, схеми структурної, функціональної, діаграми взаємодії процесів перейдемо до опису та розгляду блок-схем основної програми, та підпрограм, які використовуються, для реалізації системи.

4 РЕАЛІЗАЦІЯ РОБОТИ. РОЗРАХУНКИ І ЕКСПЕРИМЕНТАЛЬНІ ДАНІ, ЩО ПІДТВЕРДЖУЮТЬ ВІРНІСТЬ ПРОЕКТНИХ ТА ПРОГРАМНИХ РІШЕНЬ

4.1 Блок-схеми та опис алгоритмів функціонування системи

Блок-схема алгоритму роботи основної програми зображено зображена на рисунку 4.1. Зі схеми можна побачити що після запуску програми спочатку відбувається виведення вікна системи, після проходе збір інформації з послідуючим відображенням отриманих даних та побудовою графіків трафіку.

Далі проходе запит статистичного прогнозування з реалізацією та викликом підпрограми прогнозування що детально розглянута на рисунку 4.2. Далі проходе виклик підпрограми моніторингу систем мережевого аналізу з послідуючим запитом сформувати повний звіт та завершенням роботи ПЗ.

Наведемо поетапний алгоритм роботи програмного продукта, розробленого у результаті виконання роботи. Його праця складається з наступних етапів:

– Вибір часового ряду завантаженості трафіку для прогнозування. На початку роботи експертові необхідно визначитися, на підставі якого часового ряду завантаженості трафіку необхідно провести прогнозування. Після вибору часового ряду він передається в систему прогнозування.

– Візуалізація часового ряду. Переданий у систему прогнозування часовий ряд завантаженості трафіку, відображається у вигляді графіка, на якому по осі абсцис відкладаються часові такти, а по осі ординат – значення часового ряду завантаженості трафіку.

– Візуальний аналіз часового ряду завантаженості трафіку. У ході візуального аналізу часового ряду завантаженості трафіку експерт ухвалює рішення щодо того, чи варто враховувати всі дані ряду при побудові прогнозу й,

при необхідності, виключає необхідну кількість послідовних тактів ліворуч. Таким чином, ігноруються «старі» дані. У випадку, якщо експертові хочеться зіставити прогноз із реальними даними, він може виключити з розгляду кілька послідовних тактів праворуч (ігноруються «нові» дані).

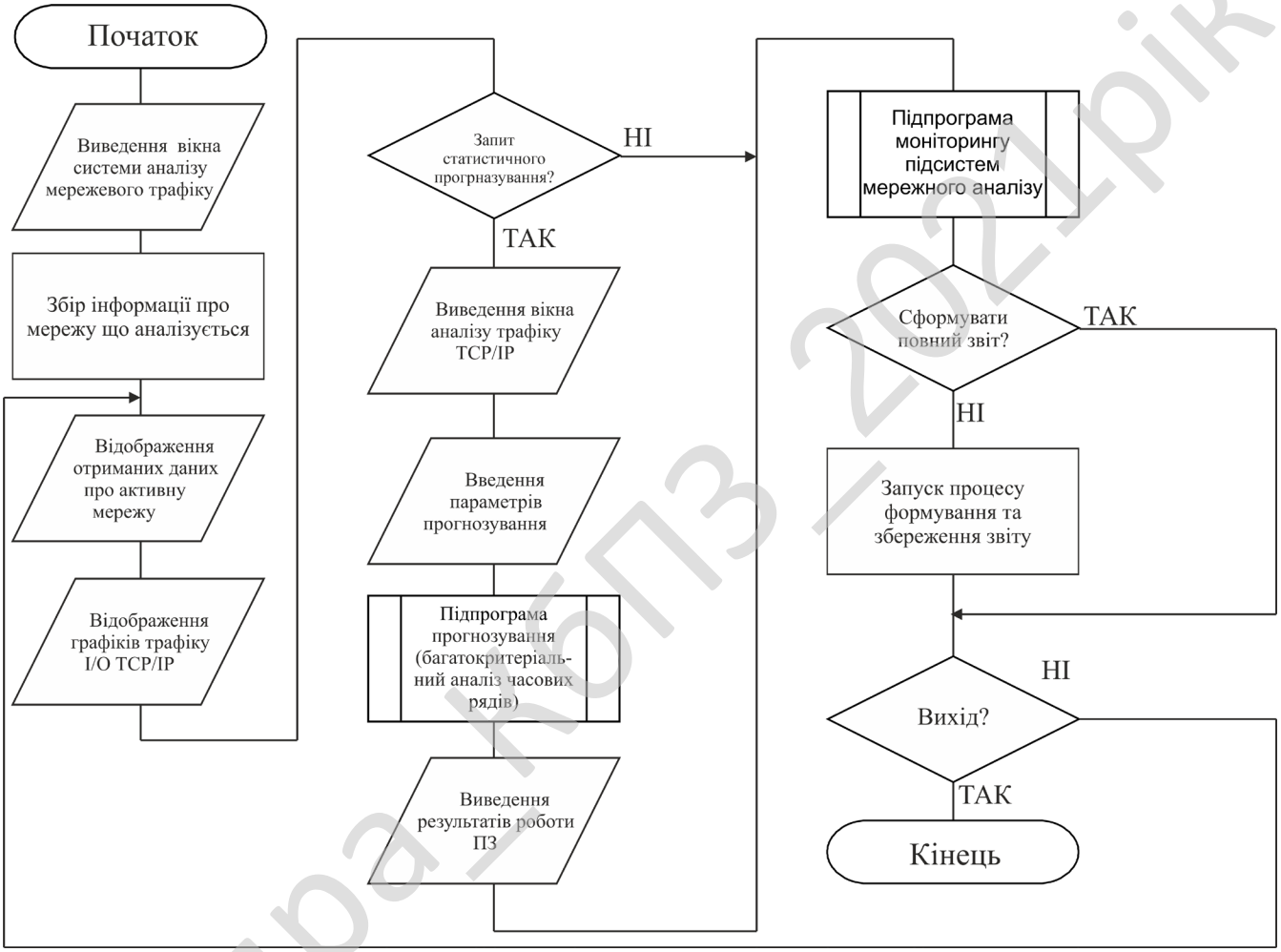


Рисунок 4.1 – Блок-схема основної програми

– «Усікання» часового ряду завантаженості трафіку по краях. Експерт указує, яку кількість тактів праворуч і ліворуч варто виключити з розгляду, після чого ці такти ігноруються системою при побудові прогнозу завантаженості трафіку.

– Вибір обрію прогнозування. На цьому етапі експерт приймає рішення, на яку кількість тактів уперед необхідно побудувати прогноз, з огляду при цьому на кількість спостережених значень у вихідному часовому ряді. Чим більше це число, тим, як правило, більш достовірним виходить прогноз.

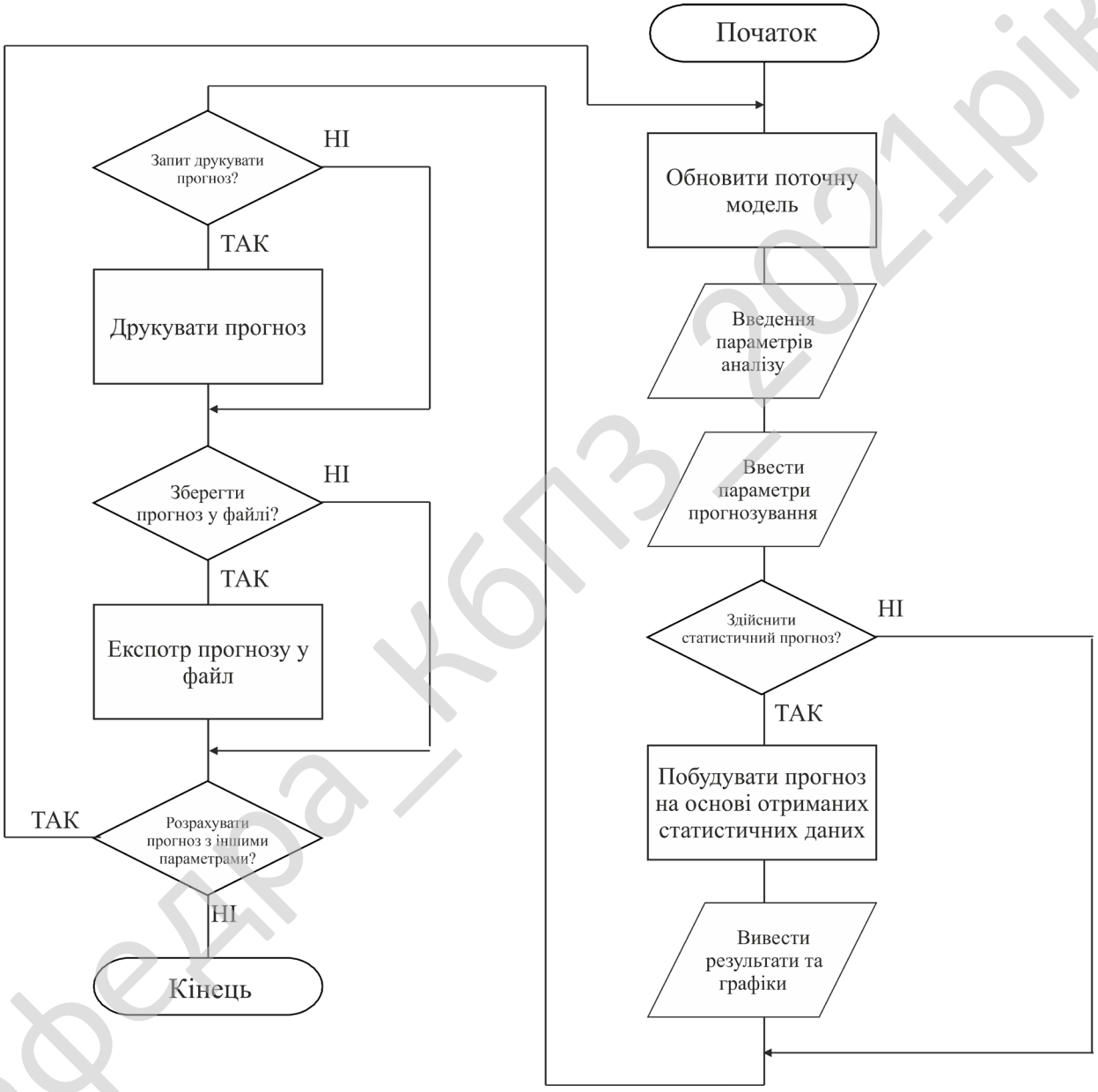


Рисунок 4.2 – Блок-схема роботи підпрограми

– Налаштування набору прогнозних моделей. Експерт визначає, які прогнозні моделі й з яким діапазоном параметрів варто використовувати при побудові прогнозу завантаженості трафіку.

– Вибір критерію оцінки якості прогнозу. Експерт вибирає формальну постановку задачі прогнозування із пропонованих системою прогнозування завантаженості трафіку. На підставі обраного критерію система прогнозування буде визначати, які прогнози «краще», а які «гірше».

– Завдання екзаменаційної вибірки. Залежно від критерію оцінки якості прогнозу й обраних прогнозних моделей, експерт вибирає крапки вихідного часового ряду, які необхідно включити в екзаменаційну вибірку, по крапках якої будуть перевірятися побудовані прогнози. За замовчуванням екзаменаційними вважаються τ останніх тактів часового ряду.

– Побудова прогнозів. Система прогнозування виконує побудову набору конкуруючих прогнозів за допомогою прогнозних моделей, обраних і настроєних користувачем.

– Ранжирування прогнозів. Після побудови набору конкуруючих прогнозів і розрахунку значень критерію якості для кожного з них, система прогнозування ранжирує ці прогнози від «кращого» до «гіршого», після чого представляє список прогнозів експертові із вказівкою значень цього критерію.

– Вибір у діалоговому режимі кращого прогнозу (кращих прогнозів). Експерт, переглядаючи послідовно (починаючи із кращого) прогнози у вигляді сполучених графіків вихідного часового ряду, допоміжного й остаточного прогнозів, вибирає найбільш раціональний.

– Експорт або друкування прогнозу. Після того як прогноз побудований, експерт може відправити результати прогнозування на друкування або здійснити експорт у файл для наступного використання.

Після розгляду загального алгоритму перейдемо до детального розгляду блок-схем основної програми та підпрограм, реалізованих у ході виконання роботи.

Для того, щоб оцінити трафік мережі та його зміну з плином часу, тобто побудувати залежність завантаженості трафіку від часу, й отримати часовий ряд, треба взяти дані про рух у мережі з таблиць маршрутизації відповідних маршрутизаторів. Для цього використовується наступна процедура.

```
procedure Get_IPForwardTable( List: TStrings );
var
  IPForwRow     : TMibIPForwardRow;
  TableSize     : DWORD;
  ErrorCode     : DWORD;
  i             : integer;
  pBuf          : PChar;
  NumEntries    : DWORD;
begin
  if not Assigned( List ) then EXIT;
  List.Clear;
  TableSize := 0;
  // перший виклик: необхідно отримати розмір таблиці у БД
ErrorCode := GetIpForwardTable( PTMibIPForwardTable( pBuf ), @TableSize,
true );
  if Errorcode <> ERROR_INSUFFICIENT_BUFFER then
    EXIT;
  // заносимо у таблицю
  GetMem( pBuf, TableSize );
  ErrorCode := GetIpForwardTable( PTMibIPForwardTable( pBuf ), @TableSize,
true );
  if ErrorCode = NO_ERROR then
  begin
    NumEntries := PTMibIPForwardTable( pBuf )^.dwNumEntries;
    if NumEntries > 0 then
    begin
     inc( pBuf, SizeOf( DWORD ) );
      for i := 1 to NumEntries do
      begin
        IPForwRow := PTMibIPForwardRow( pBuf )^;
        with IPForwRow do
List.Add( Format( ' %15s|%15s|%15s|%8.8x|%7s|   %5.5d|    %7s|        %2.2d' ,
[IPAddr2Str( dwForwardDest ),   IPAddr2Str( dwForwardMask),
IPAddr2Str(dwForwardNextHop), dwForwardIFIndex, IPForwTypes[dwForwardType],
dwForwardNextHopAS, IPForwProtos[dwForwardProto],  dwForwardMetric1] ) );
  inc( pBuf, SizeOf( TMibIPForwardRow ) );
```

```
        end;
      end
      else
        List.Add( ' Даних немає.'  );
    end
    else
      List.Add( SysErrorMessage( ErrorCode ) );
    dec( pBuf, SizeOf( DWORD ) + NumEntries * SizeOf( TMibIPForwardRow ) );
    FreeMem( pBuf );
  end;
```

Система прогнозування інтегрована в клієнтський додаток як підсистема «Прогноз», що дозволяє користувачеві в ході роботи врахувати всі кроки діалогового алгоритму побудови прогнозу.

Після вибору критерію оцінки якості відбувається завдання екзаменаційної вибірки.

Для цього виділимо з вихідного часового ряду завантаженості трафіку навчальну вибірку $X_{об} = \{x(1),\ x(2),...,\ x(N-\tau)\}$, на підставі спостережень якої побудуємо оцінки значень часового ряду завантаженості трафіку на тактах з 1 по $N-\tau$, і прогнозні значення на тактах з $N-\tau+1$ по N. $X_{доп} = \{x'(1),\ x'(2),...,\ x'(N-\tau),\ x'(N-\tau+1),\ x'(N-\tau+2),...,\ x'(N)\}$. Потім, вибравши $к$ довільних крапок вихідного часового ряду, складемо з них екзаменаційну вибірку $X_{icn} = \{x(t_1),\ x(t_2),...,\ x(t_j),...,x(t_k)\}$, де $1 \le t_j \le N$.

З отриманого допоміжного прогнозу $X'_{доп}$ виберемо оцінки значень часового ряду на часових тактах, що ввійшли в екзаменаційну вибірку $X'_{icn} = \{x'(t_1),\ x'(t_2),...,\ x'(t_j),...,\ x'(t_k)\}$, $1 < t_j < N$. Зіставляючи отримані оцінки значень часового ряду X'_{icn}, зі значеннями екзаменаційної вибірки X_{icn}, оцінимо його якість, використовуючи різні критерії оцінки якості прогнозування. Таким чином, допоміжний прогноз буде перевірений на наявних даних.

Побудуємо прогноз X', на підставі всіх наявних спостережень часового ряду X, використовуючи при цьому тої же метод прогнозування, що й при побудові допоміжного прогнозу. У зв'язку з тим, що оцінити якість прогнозу X на

реальних даних не представляється можливим, будемо припускати, що його якість таке ж, як і якість допоміжного прогнозу.

Таким чином, залежно від критерію оцінки якості прогнозу й обраних прогнозних моделей, експерт вибирає крапки вихідного часового ряду, які необхідно включити в екзаменаційну вибірку, по крапках якої будуть перевірятися побудовані прогнози завантаженості трафіку. За замовчуванням екзаменаційними вважаються τ останніх тактів часового ряду.

За завданням екзаменаційної вибірки відбувається суто побудова прогнозів.

Для побудови набору конкуруючих прогнозів необхідно нажати кнопку «Побудувати прогноз» на панелі інструментів. Після розрахунку в нижній частині вікна системи прогнозування буде виведений ранжируваний список побудованих прогнозів із вказівкою використаної прогнозної моделі і її параметрів, значення критерію якості оцінки прогнозу, а також величина критерію «середня помилка». Цей критерій дозволяє наочно представити й зрівняти якість прогнозів, побудованих на підставі різних по масштабі даних, оскільки в ньому здійснюється перехід до відносних величин, що дозволяє експертові легко інтерпретувати величини помилок.

Після відображення результатів прогнозування відбувається візуальний аналіз часового ряду адміністратором мережі, тобто користувачем розробленої програми.

У результаті візуального аналізу часового ряду завантаженості трафіку адміністратор мережі приймає рішення про корекцію навантаження на ті або інші вузли мережі, яку він обслуговує. Для цього він змінює динамічні таблиці маршрутизації у відповідних маршрутизаторах, або проводить перекрмутування каналів у відповідних комутаторах.

Після проведення перерахованих вище дій, необхідно ще раз запустити програму, для того, щоб пересвідчитися в тому, що оптимізація мережі, яка була

проведена на основі попереднього аналізу часових рядів завантаженості трафіку, дала свої позитивні результати. Тобто мережа не є перезавантаженою.

Незважаючи на те що я працював над ПЗ один в реалізації програми я використовував підходи пришвидшення розробки на основі методологій Agile – Extreme Programming.

Екстремальне програмування (Extreme Programming, далі XP) це методологія розробки програмного забезпечення, найпопулярніша серед так званих гнучких методологій. Має на меті поліпшення якості програмного забезпечення та чутливість до змін у вимогах замовників. Як вид гнучких методологій, XP радить часті "випуски" програми у коротких циклах розробки, що має на меті поліпшити продуктивність праці та покращити можливості виконання вимог замовника що змінюються. Авторами даної методології є Кент Бек, Ворд Каннінгем, Мартін Фаулер та інші.

Інші елементи екстремального програмування включають в собі: парне програмування, проведення обширної перевірки сирцевого коду, модульне тестування всього коду, уникання створення функціональності до того як вона дійсно необхідна, простота та ясність коду, очікування на зміну вимог замовників з плином часу та коли вимоги до продукту стають ясніші, досить часте спілкування із замовником та між самими програмістами.

Назва методології походить від ідеї застосувати корисні методи і практики розробки програмного забезпечення, піднявши їх до "екстремальних" рівнів.

Критики XP зауважують на потенційні недоліки цієї методології – нестабільні вимоги, незадокументовані компроміси конфліктів користувачів, відсутність загального документу дизайну програми.

Технологія екстремального програмування була розроблена Кентом Беком, Уардом Каннінгхемом та Роном Джеффріесом під час роботи над Chrysler Comprehensive Compensation System (C3). У 1996 Кент Бек став лідером проекту і почав вдосконалювати методи розробки, що застосовувалися в роботі над проектом. Свій метод він виклав у книзі «Extreme Programming Explained», котру

було видано у жовтні 1999. Після купівлі Крайслера компанією Даймлер–Бенц проект C3 було скасовано у лютому 2000.

Хоча саме екстремальне програмування є відносно новим, багато її практик вже існували і використовувались протягом певного часу; однак, методологія підносить "найкращі практики" до екстремального рівня. Для прикладу, практика по плануванню і написанню тестів перед написанням кожної маленької частини коду було використано раніше в проекті НАСА "Меркурій". Для зменшення часу на розробку ПЗ деякі формальні документи тестування (такі як приймальне тестування) писались паралельно (або й раніше) з написанням самого ПЗ. Незалежна група тестування НАСА може писати процедури тестування базуючись на формальних вимогах до продукту до того як програмне забезпечення розроблене та інтегроване в систему. В XP ця концепція піднесена до "екстремального рівня" завдяки написанню автоматичних тестів які перевіряють поведінку навіть малих частинок коду, а не тільки значних функціональних частин ПЗ.

Посібник Extreme Programming Explained: Embrace Change описує Екстремальне Програмування, як:

– Спроба примирити гуманність і продуктивність.

– Механізм для соціальної зміни.

– Шлях до удосконалення.

– Стиль розвитку.

Дисципліна розробки програмного забезпечення.

Головною метою Екстремального Програмування є скорочення вартості неочікуваних змін. У традиційних методах розробки (на кшталт SSADM) вимоги до розвитку системи визначаються на початку роботи над проектом, і часто виправляються пізніше. Це означає, що вартість проекту через зміни буде більшою за заплановану (традиційна особливість для програмного забезпечення, що проектується).

ХР використовується для скорочення вартості змін, завдяки представленню простих значень, принципів і методів. При використанні екстремального програмування, проект повинен стати гнучкішим щодо змін.

Extreme Programming Explained описує екстремальне програмування як дисципліну розробки програмного забезпечення яка змушує людей створювати високоякісне ПЗ якомога швидше.

ХР намагається зменшити ціну зміни вимог до ПЗ завдяки малим циклам розробки, а не одним довгим циклом. Екстремальне програмування сприймає зміни до вимог як звичайні, неминучі та бажані аспекти розробки ПЗ, і ці зміни мають бути очікуваним. Основна ідея полягає в тому що неможливо розробити самодостатній пакет вимог до ПЗ, зміни в вимогах – неминучі.

Екстремальне програмування також вводить набір практик та принципів на основі методології гнучкої розробки програмного забезпечення.

Екстремальне програмування описує чотири базові активності що виконуються при розробці програмного забезпечення: написання коду, тестування, слухання та дизайн. Написання коду. Прихильники ХР заявляють що єдиним дійсно важливим результатом розробки ПЗ є код: без готового коду нема продукту.

Тестування. Методологія екстремального програмування заявляє, що якщо дрібне тестування може перевірити незначну частину функціональності, то багато дрібних тестів можуть перевірити набагато більше частинок і продукт в цілому.

Основні прийоми ХР. Дванадцять основних прийомів екстремального програмування (за першим виданням книги Extreme programming explained) можуть бути об'єднані в чотири групи:

1. Короткий цикл зворотного зв'язку (Fine scale feedback).

1.1. Розробка через тестування (Test driven development).

1.2 Гра в планування (Planning game).

1.3. Замовник завжди поруч (Whole team, Onsite customer).

1.4 Парне програмування (Pair programming).

2. Безперервний, а не пакетний процес.

2.1 Безперервна інтеграція (Continuous Integration).

2.2 Рефакторинг (Design Improvement, Refactor).

2.3 Часті невеликі релізи (Small Releases).

3. Розуміння, що поділяється всіма учасниками.

3.1 Простота (Simple design).

3.2 Метафора системи (System metaphor).

3.3 Колективне володіння кодом (Collective code ownership) або обраними шаблонами проектування (Collective patterns ownership).

3.4 Стандарт кодування (Coding standard or Coding conventions).

4. Соціальна захищеність програміста (Programmer welfare), а саме 40 годинний робочий тиждень (Sustainable pace, Forty hour week).

4.2 Захист розробленого програмного забезпечення

Дані у системі захищаються за допомогою алгоритму обміну ключа Діффі-хеллмана.

Ціль алгоритму полягає в тому, щоб два учасники могли безпечно обмінятися ключем, що надалі може використовуватися в якому-небудь алгоритмі симетричного шифрування. Сам алгоритм Діффі-хеллмана може застосовуватися тільки для обміну ключами.

Алгоритм заснований на труднощі обчислень дискретних логарифмів. Дискретний логарифм визначається в такий спосіб. Уводиться поняття примітивного кореня простого числа Q як числа, чиї ступені створюють всі цілі від 1 до Q - 1. Це означає, що якщо A є примітивним коренем простого числа Q, тоді числа:

$$A \bmod Q,\ A^2 \bmod Q,\ \ldots,\ A^{Q-1} \bmod Q,$$

є різними й складаються із цілих від 1 до Q - 1 з деякими перестановками. У цьому випадку для будь-якого цілого Y < Q і примітивного кореня A простого числа Q можна знайти єдину експоненту X, таку, що:

$$Y = A^X \bmod Q,$$

де $0 \le X \le (Q - 1)$.

Експонента X називається дискретним логарифмом, або індексом Y, по підставі A mod Q. Це позначається як $\text{ind}_{,Q}(Y)$.

Тепер опишемо алгоритм обміну ключів Діффі-хеллмана.

Загальновідомі елементи

Q – просте число.

A – A < Q і A є примітивним коренем Q.

Створення пари ключів користувачем I

Вибір випадкового числа X_i (закритий ключ):

$$X_i < Q.$$

Обчислення числа Y_i (відкритий ключ):

$$Y_i = A^{Xi} \bmod Q.$$

Створення відкритого ключа користувачем J

Вибір випадкового числа X_j (закритий ключ):

$$X_j < Q.$$

Обчислення випадкового числа Y_j (відкритий ключ):

$$Y_j = A^{Xj} \bmod Q.$$

Створення загального секретного ключа користувачем I

$$K = (Y_j)^{Xi} \bmod Q.$$

Створення загального секретного ключа користувачем J

$$K = (Y_i)^{Xj} \bmod Q.$$

Передбачається, що існують два відомих усім числа: просте число Q і ціле A, що є примітивним коренем Q.

Тепер припустимо, що користувачі I і J хочуть обмінятися ключем для алгоритму симетричного шифрування.

Користувач I вибирає випадкове число $X_i < Q$ і обчислює:

$$Y_i = A^{Xi} \bmod Q.$$

Аналогічно користувач J незалежно вибирає випадкове ціле число $X_j < Q$ і обчислює:

$$Y_j = A^{Xj} \bmod Q.$$

Кожна сторона тримає значення X у секреті й робить значення Y доступним для іншої сторони.

Тепер користувач I обчислює ключ як $K = (Y_j)^{Xi} \bmod Q$, і користувач J обчислює ключ як $K = (Y_i)^{Xj} \bmod Q$. У результаті обоє одержать те саме значення:

$$K = (Y_j)^{Xi} \bmod Q = (A^{Xj} \bmod Q)^{Xi} \bmod Q = (A^{Xj})^{Xi} \bmod Q =$$

за правилами модульної арифметики

$$= A^{Xj\,Xi} \bmod Q = (A^{Xj})^{Xj} \bmod Q = (A^{Xi} \bmod Q)^{Xj} \bmod Q = (Y_i)^{Xj} \bmod Q$$

Таким чином, дві сторони обмінялися секретним ключем. Так як X_i і X_j є закритими, супротивник може одержати тільки наступні значення: Q, A, Y_i і Y_j. Для обчислення ключа атакуючий повинен зламати дискретний логарифм, тобто обчислити:

$$X_j = \mathrm{ind}_{a,\,q}(Y_j).$$

Безпека обміну ключа в алгоритмі Діффі-хеллмана випливає з того факту, що, хоча відносно легко обчислити експоненти по модулю простого числа, дуже важко обчислити дискретні логарифми. Для великих простих чисел задача вважається нерозв'язною.

Варто помітити, що даний алгоритм уразливий для атак типу "in-the-middle". Якщо супротивник може здійснити активну атаку, тобто має можливість не тільки перехоплювати повідомлення, але й заміняти їх іншими, він може перехопити відкриті ключі учасників Y_i і Y_j , створити свою пару відкритого й закритого ключа ($X_{оп}$, $Y_{оп}$) і послати кожному з учасників свій відкритий ключ. Після цього кожний учасник обчислить ключ, що буде загальним із супротивником, а не з іншим учасником. Якщо немає контролю цілісності, то учасники не зможуть виявити подібну підміну.

5 МЕТОДИКА ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

Програма має простий та інтуїтивно зрозумілий інтерфейс, який зображений на рисунку 5.1. З рисунку головного вікна можна побачити що інтерфейс головного вікна розподілено на наступні функціональні розділи:

– Розділу виведення результату роботи системи у вигляді пів кільцевого графіку інформаційних значень.

– Функціональних кнопок: Прогнозування; Оптимізації; Авторського права.

– Навігаційного меню яке викликається натисканням правої клавіші маніпулятора миші.

– Розділу обрання мережних даних.

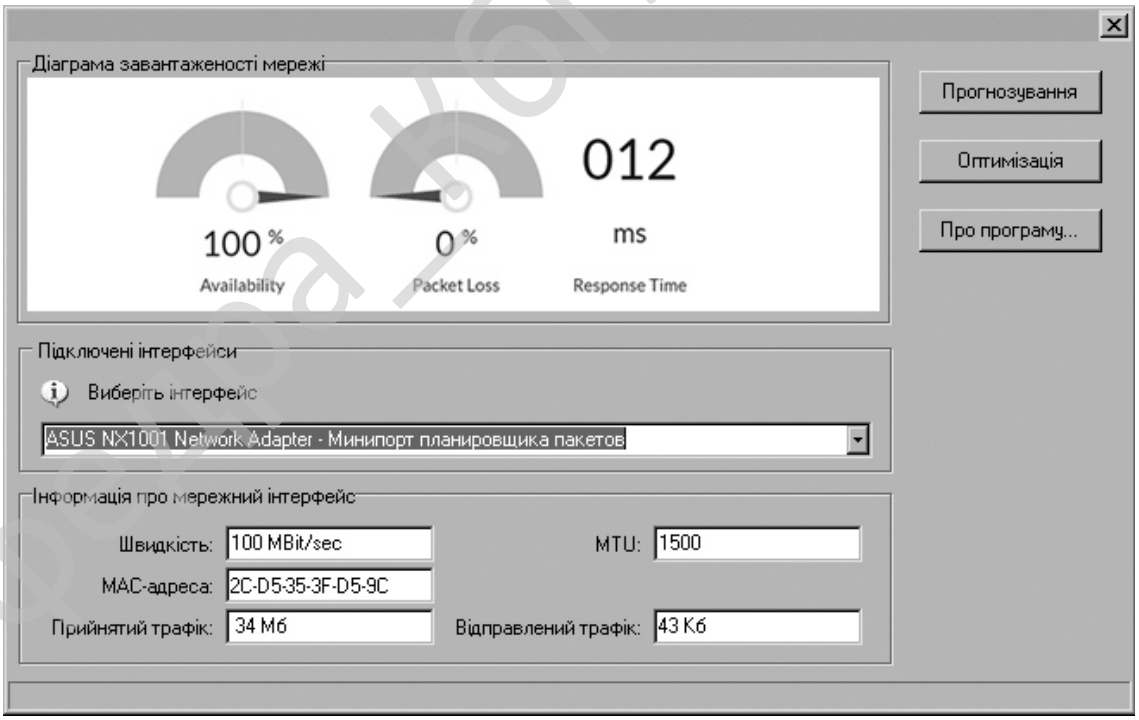


Рисунок 5.1 – Головне вікно ПЗ

На рисунку 5.2 зображено авторські дані розробленого програмного забезпечення. Розроблена програма має дуже простий і зрозумілий інтерфейс з користувачем. Кожен, хто в достатньому обсязі володіє операційним середовищем Windows без особливих складностей освоїть і цю програму, оскільки її інтерфейс інтуїтивно зрозумілий. Якщо програма не видала ніяких помилок, і працює, то можна використовувати, інакше слід слідувати інструкціям, які пропонує програма.

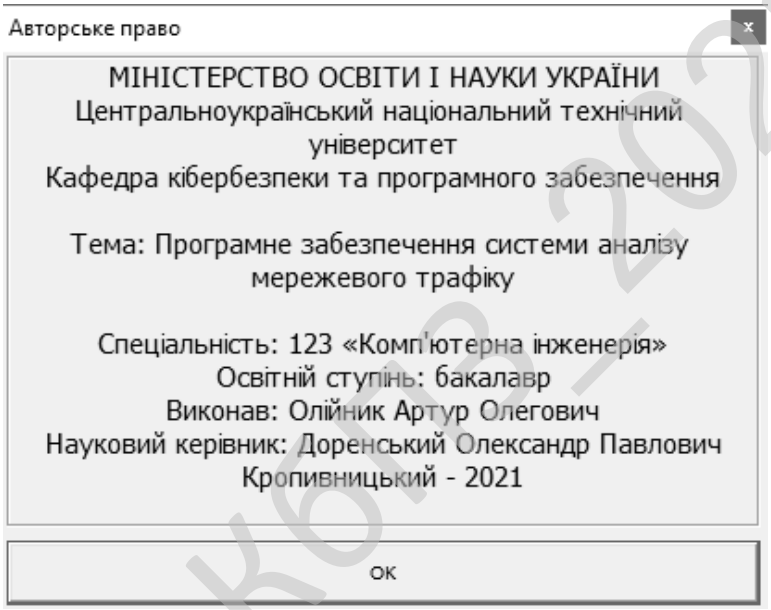


Рисунок 5.2 – Авторське право

Розглянемо процес впровадження програмного забезпечення, це процес налаштування програмного забезпечення під певні умови використання, а також навчання користувачів роботі з програмним продуктом. Впровадження програмного забезпечення це усі дії, що роблять розроблену програмну систему готовою до використання. Даний процес є частинною життєвого циклу програмного забезпечення.

6 ОСНОВНІ ВИСНОВКИ

Програмне забезпечення, створене в результаті виконання кваліфікаційної бакалаврської роботи, призначено для системи аналізу мережевого трафіку.

В межах України в недостатній мірі представлені вітчизняні розробки в цій області.

Рішення завдання полягало у вирішенні наступних задач:

– Був проведений огляд існуючих систем аналізу мережевого трафіку.

– Досліджена система аналізу мережевого трафіку.

– На основі отриманих результатів досліджень створена програмна реалізація системи аналізу мережевого трафіку.

Розроблені під час виконання кваліфікаційної бакалаврської роботи алгоритми дозволяють успішно вирішувати завдання аналізу мережевого трафіку.

Розроблене програмне забезпечення має простий, дружній та зручний інтерфейс користувача, що забезпечує легкість у освоєнні роботи програмного продукту, зручність у використанні, і не потребує особливих спеціальних знань.

При створені програмного забезпечення було використано об’єктно-орієнтований підхід, що відповідає сучасним тенденціям у галузі розробки комерційних програмних систем.

Програма реалізована на мові високого рівня Embarcadero Delphi. Дана мова програмування дозволяє найбільш ефективно обробляти дані призначені для системи аналізу мережевого трафіку. Це дозволило мінімізувати строк розробки програмного забезпечення, і, як слід, зменшити витрати на його розробку. Запропоноване програмне забезпечення ділиться на загальне програмне забезпечення, що поставляється із засобами обчислювальної техніки й спеціальне програмне забезпечення, що спеціально розроблене для даної конкретної системи й включає програми, що реалізують її функції.

Програма призначена для виконання під управлінням багатозадачної операційної системи Windows 10.

Даються необхідні рекомендації з установки розробленого програмного забезпечення.

Для підвищення рівня безпеки запропоновано застосовувати алгоритм Діффі-Хеллмана.

В цілому створене програмне забезпечення підтверджує правильність використаних проектних рішень та повністю відповідає вимогам технічного завдання. Створене програмне забезпечення має потенційну можливість для подальшого вдосконалення і застосування у різних галузях.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Конахович Г.Ф. Сети передачи пакетных данных / Г.Ф. Конахович, В.М.Чуприн. – К.: МК-Пресс, 2006. – 272 с.

2. Кучук Г.А. Управление ресурсами инфотелекоммуникаций / Г.А. Кучук, Р.П. Гахов, А.А. Пашнев. – М.: Физматлит, 2006. – 220 с.

3. Лавренков Ю.Н. Исследование и разработка комбинированных нейросетевых технологий для повышения эффективности безопасной маршрутизации информации в сетях связи: диссертация ... кандидата технических наук: 05.13.17 / Лавренков Юрий Николаевич, 2014.– 208 с.

4. Лазарев И. А. Композиционное проектирование сложных агрегативных систем / И.А. Лазарев. – М.: Радио и связь, 1986. – 312 с.

5. Лукацкий А.В. Обнаружение атак / А.В. Лукацкий. – С.Пб.: BHV, 2001. – 624 с.

6. Майника Э. Алгоритмы оптимизации на сетях и графах: пер. с англ. / Э. Майника; под ред. Е.К. Масловского. – М.: Мир, 1981. – 321 с.

7. МСЭ-Т Рекомендация G.101. Международные телефонные соединения и цепи – Общие определения //11/2003. [Електронний ресурс]. – Режим доступу до ресурсу: http://www.telecom61.ru/SharedFiles/ Download.aspx? …pageid=106

8. Назаров А.Н. Модели и методы расчета структурно-сетевых параметров АТМ сетей / Алексей Николаевич Назаров. – М.: Горячая линия - Телеком, 2002. – 256 с.

9. Одом Ш. Коммутаторы CISCO / Ш. Одом, Х. Ноттингем – М.: "Кудиц-Образ", 2003. – 528 с.

10. Олифер В.Г. Компьютерные сети. Принципы, технологии, протоколы: учебник для вузов / В.Г. Олифер, Н.А. Олифер. – 2-е изд. – СПб.: Питер, 2007. – 958 с.

11. Партыка С.А. Метод ускоренной коррекции SPT с использованием динамических алгоритмов [Електронний ресурс]. – Режим доступу до ресурсу: http://openarchive.nure.ua/bitstream/123456789/936/1/ASU_158_2012%20%2842-47%29.pdf

12. Руководство по технологиям объединенных сетей. 4-е изд. / пер.с англ. и ред. А.Н. Крикуна – М.: Изд. дом «Вильямс», 2005. – 1040 с.

13. Семенов С.Г. Математическая модель мультисервисного канала связи на основе экспоненциальной GERT-сети / С.Г. Семенов, Є.В. Мелешко, Я.В. Ілюшко // Системи озброєння і військова техніка. – Х.:ХУ ПС. – 2011. –Вип. 3(27). – С. 64-67.

14. Семенов С.Г. Математична модель системи криптографічного захисту електронних повідомлень на основі GERT-мережі / С.Г. Семенов, О.О. Сур // Системи управління, навігації та зв'язку. – К.: ЦНДІ н авігації і управління. – 2012. – Том 1. Вип. 1(21). – С. 131-137

15. Семенов С.Г. Исследования вероятностно-временных характеристик мультисервисного канала связи с использованием математического аппарата GERT-сети / С.Г. Семенов, В.В. Босько, І.А. Березюк // Системи обробки інформації. – Х.: ХУ ПС, 2012. – Т. 1., Вип. 3(101). – С. 139-142.

16. Семенов С.Г. Моделирование защищенного канала связи с использованием экспоненциальной GERT-сети / С.Г. Семенов, А.А. Можаев // Информатика, математическое моделирование, экономика. – Смоленськ.: Смоленский филиал АНО ВПО ЦС РФ "Российский университет кооперации". – 2012. – Том.1. – С. 152-160.

17. Семенов С.Г. Методика математического моделирования защищенной ИТС на основе многослойной GERT-сети / С.Г. Семенов // Вісник Національного технічного університету «Харківський політехнічний інститут». – Х.:НТУ «ХПІ». – 2012. –№62 (968). – С 173-181.

18. Семенов С.Г. Защита данных в компьютеризированных управляющих системах / С.Г. Семенов, В.В. Давыдов, С.Ю. Гавриленко. – LAP Lambert Academic Publishing GmbH & Co. KG (Саарбрюккен, Германия), 2014. – 236 с.

19. Смирнов А.А. Анализ и сравнительное исследование перспективных направлений развития цифровых телекоммуникационных систем и сетей / А.А.Смирнов, В.В.Босько, Е.В.Мелешко // Системи обробки інформації. – Х.: ХУ ПС, 2008. – Вип.7(74). – С.120-123.

20. Смирнов А.А. Усовершенствование метода управления очередями в многопротокольных узлах телекоммуникационной сети / А.А.Смирнов, Е.В.Мелешко // Збірник тез та доповідей другої всеукраїнської науково-практичної конференції «Системний аналіз. Інформатика. Управління». Запоріжжя. Тези доповідей. Запоріжжя: КПУ, 2011.

21. Смирнов С.А. Метод безопасной маршрутизации метаданных в облачные антивирусные системы / А.К. Дидык, С.А. Смирнов // Информационные технологии в управлении, образовании, науке и промышленности: монография / Под редакцией профессора В.С. Пономаренко. – Х.: Видавець Рожко С.Г., 2016. – 566 с.

22. Смирнов С. А. Сравнительные исследования математических моделей технологии распространения компьютерных вирусов в информационно-телекоммуникационных сетях / Мохамад Абу Таам Гани, А. А. Смирнов, А. В. Коваленко, С. А. Смирнов // Системи обробки інформації: зб. наук. праць. – Х.: ХУПС, 2014. – Вип. 9(125). – 105-110.

23. Смирнов С. А. Математическая модель интеллектуального узла коммутации с обслуживанием информационных пакетов различного приоритета / Мохамад Абу Таам Гани, А. А. Смирнов, Н. С. Якименко, С. А. Смирнов //

Збірник наукових праць Харківського університету Повітряних Сил. – Харків: ХУПС, 2014. – Вип. 4 (41). – С. 48-52.

24. Смирнов С. А. Исследование показателей качества функционирования интеллектуальных узлов коммутации в телекоммуникационных системах и сетях / Мохамад Абу Таам Гани, А. А. Смирнов, Н. С. Якименко, С. А. Смирнов // Наука і техніка Повітряних Сил Збройних Сил України: наук. журн. –Х.: ХУПС, 2014. – № 4(17). – С. 90-95.

25. Смирнов С. А. Усовершенствованный алгоритм управления доступом к «облачным» телекоммуникационным ресурсам / Мохамад Абу Таам Гани, А. А. Смирнов, Н. С. Якименко, С. А. Смирнов // Системи обробки інформації: зб. наук. праць. – Х.: ХУПС, 2015. –Вип. 1(126). – С. 150-153.

26. Smirnov S.A. Method of controlling access to intellectual switching nodes of telecommunication networks and systems / A.A. Smirnov, Mohamad Abou Taam, S.A. Smirnov // International Journal of Computational Engineering Research (IJCER). – Volume 5, Issue 5. – India. Delhi. – 2015. – P. 1-7.

27. Смирнов С. А. Анализ и исследование методов управления сетевыми ресурсами для обеспечения антивирусной защиты даннях / Мохамад Абу Таам Гани, А. А. Смирнов, С. А. Смирнов // Системи озброєння і військова техніка: наук. журн. – Х.: ХУПС, 2015. – № 3(43). – С. 100-107.

28. Смирнов С. А. Исследование эффективности метода управления доступом к облачным антивирусным телекоммуникационным ресурсам / Мохамад Абу Таам Гани, А. А. Смирнов, С. А. Смирнов // Наука і техніка Повітряних Сил Збройних Сил України: наук. журн. –Х.: ХУПС, 2015. – № 3(20). – С. 134-141.

29. Смирнов С. А. Комплекс GERT-моделей технологии облачной антивирусной защиты телекоммуникационной системы / А. А. Смирнов, А. К. Дидык, А. Н. Дреев, С. А. Смирнов // Безпека інформації: наук. - практ. журн. – К.: НАУ, 2015. – Т. 21, № 3. – С. 251-262.

30. Смирнов С. А. Метод безопасной маршрутизации метаданных в облачные антивирусные системы / А. А. Смирнов, А. К. Дидык, С. А. Смирнов // Системи озброєння і військова техніка: наук. журн. – Х.: ХУПС, 2016. – № 2 (46). – С. 146-149.

31. Смирнов С. А. Модели системы нейросетевых экспертов безопасной маршрутизации в облачных антивирусных системах / А. А. Смирнов, А. К. Дидык, А. Н. Дреев, С. А. Смирнов // Системи обробки інформації: зб. наук. праць. – Х.: ХУПС, 2016. – Вип. 3 (140). – С. 36-39.

32. Смирнов С. А. Метод безопасной маршрутизации на базовом множестве путей передачи метаданных в облачные антивирусные системы / В. Л. Бурячок, С. А. Смирнов // Системи управління, навігації та зв'язку. – Полтава, 2016. – Вип. 4(40). – С. 57-62.

33. Смирнов С. А. Способ контроля линий связи телекоммуникационной системы облачного антивируса / А. А. Смирнов, А. К. Дидык, А. Н. Дреев, С. А. Смирнов // Збірник наукових праць Харківського університету Повітряних Сил. – Харків: ХУПС, 2016. – № 2 (47). – С. 148-152.

34. Смирнов С. А. Дослідження та реалізація GERT-моделі технології розповсюдження комп'ютерних вірусів для захисту телекомунікаційних систем / В. Л. Бурячок, Мохамад Абу Таам Гани, С. А. Смирнов // Інформаційні технології та комп'ютерна інженерія: зб. тез доп. наук.-практ. конф., м. Кіровоград, 4 грудня 2014 р. – Кіровоград: КНТУ, 2014. – С. 168.

35. Смирнов С. А. Исследование математических моделей технологии распространения компьютерных виру сов / А. А. Смирнов, Мохамад Абу Таам Гани, С. А. Смирнов // Актуальні питання забезпечення кібернетичної безпеки та захисту інформації: зб. наук. праць міжнар. наук.-практ. конф., м. Київ, 25-28 лютого 2015 р. – К.: Європейський університет, 2015. – С. 90-91.

36. Смирнов С. А. Метод управления доступом к «облачным» ресурсам для защиты телекоммуникационных систем / Мохамад Абу Таам Гани, А. А. Смирнов, С. А. Смирнов // Всеукраїнська науково-практична конференція

«Інформаційна безпека держави, суспільства та особистості», м. Кіровоград, 16 квітня 2015 р.: зб. тез доп. – Кіровоград: КНТУ, 2015. – С. 50-52.

37. Смирнов С. А. Разработка метода управления доступом в интеллектуальных узлах коммутации / А. А. Смирнов, Мохамад Абу Таам Гани, С. А. Смирнов // Проблеми і перспективи розвитку ІТ-індустрії: зб. тез VII міжнар. наук.-практ. конф., м. Харків, 17-18 квітня 2015 р. – Х.: ХНЕУ, 2015. – С. 14.

38. Смирнов С.А. Реализация метода управления доступом в интеллектуальных узлах коммутации / А.А. Смирнов, Мохамад Абу Таам Гани, С.А. Смирнов // Збірник тез XVII міжнародного науково-практичного семінару «Комбінаторні конфігурації та їх застосування». м. Кіровоград. 17-18 квітня 2015 р. – Кіровоград: КНТУ. – 2015. – С. 91-92.

39. Смирнов С. А. Технология передачи сигнатур в облачные антивирусные системы для обеспечения защищенности телекоммуникационных сетей / А. А. Смирнов, С. А. Смирнов // Збірник тез V міжнародної науково-технічної конференції «ITSEC», Київ, 19-22 травня 2015 р. – К.: НАУ 2015. – С. 12-13.

40. Смирнов С. А. Реализация математической модели интеллектуального узла коммутации для обеспечения защищенности телекоммуникационной сети / Мохамад Абу Таам Гани, А. А. Смирнов, С. А. Смирнов // Інформаційна та економічна безпека (INFECO-2015): зб. тез II Міжнар. наук.-практ. Інтернет-конф., м. Харків, 21-22 травня 2015 р. – Х.: ХІБС УБС НБУ, 2015. – С. 20-24.

41. Смирнов С. А. Разработка математической модели технологии распространения компьютерных вирусов в информационно-телекоммуникационных сетях / Мохамад Абу Таам Гани, А. А. Смирнов, С. А. Смирнов // Сборник тезисов XI международной конференции «Стратегия качества в промышленности и образовании», г. Варна, Болгария, 01-06 июня 2015 г. – Варна: ТУВ, 2015. – С. 488-491.

42. Смирнов С. А. Метод управления доступом к облачным телекоммуни-кационным ресурсам для обеспечения защиты даннях / Мохамад Абу Таам Гани, А. А. Смирнов, С. А. Смирнов // Комп'ютерні технології та інформаційна безпека: зб. тез доп. міжнар. наук.-практ. конф., м. Кіровоград, 2-3 липня 2015 р. – Кіровоград: КНТУ, 2015. – С. 4-5.

43. Смирнов С. А. Имитационная модель системы управления доступом к облачным антивирусным телекоммуникационным ресурсам / Мохамад Абу Таам Гани, А. А. Смирнов, С. А. Смирнов // Збірник тез першої всеукраїнської науково-практичної конференції «Перспективні напрями захисту інформації» (м. Затока, 7-9 вересня 2015 р.). – Одеса: ОНАЗ, 2015. – С. 90-94.

44. Смирнов С. А. Разработка комплекса GERT-моделей технологии облачной антивирусной защиты телекоммуникационной системы / А. А. Смирнов, С. А. Смирнов, А. К. Дидык // Інформаційні технології та взаємодії» (IT & I): зб. тез II міжнар. наук. -практ. конф., м. Київ, 3-5 листопада 2015 р. – К.: КНУ ім. Тараса Шевченка, 2015. – С. 65-67.

45. Смирнов С. А. Разработка моделей телекоммуникационной системы формирования и обработки метаданных в облачных антивирусных системах / А. А. Смирнов, С. А. Смирнов, А. К. Дидык // Информационные и телекоммуникационные технологии: образование, наука, практика: сб. тезисов II междунар. научно-практ. конф., г. Алматы, Казахстан, 3-4 декабря 2015 г. – Алматы: КазНИТУ им. К.И. Сатпаева, 2015. – С. 309-313.

46. Смирнов С. А. GERT-модели технологии облачной антивирусной защиты / А. А. Смирнов, С. А. Смирнов, А. К. Дидык // Безпека українського суспільства в концепції вступу в постіндустріальне суспільство ЄС: зб. тез Круглого столу, м. Київ, 16 грудня 2015 р. – К.: Європейський університет, 2015. – С.41-43.

47. Смирнов С. А. Алгоритмы формирования множества маршрутов передачи метаданных в облачные антивирусные системы / А. А. Смирнов, С. А. Смирнов, А. К. Дидык // Актуальні питання забезпечення

кібернетичної безпеки та захисту інформації: зб. наук. праць II Міжнар. наук.-практ. конф., м. Київ, 24-27 лютого 2016 р. – К.: Європейський університет, 2016. – С. 140-142.

48. Смирнов С. А. Разработка и реализация метода безопасной маршрутизации метаданных в облачные антивирусные системы / А. А. Смирнов, С. А. Смирнов, А. К. Дидык // Securitea informationala 2015-2016: Conferenta internationala (editia a XII-a), Chisinau, Moldova, 3 martie 2016. – Chisinau: ADSEM, 2016. – С. 90-96.

49. Смирнов С. А. Алгоритм формирования базового множества маршрутов передачи метаданных в облачные антивирусные системы / А. А. Смирнов, С. А. Смирнов, А. К. Дидык // Інформатика та системні науки (ІСН-2016): зб. тез VII всеукр. наук.-практ. конф., м. Полтава, 10-12 березня 2016 р. – Полтава: ПУЕТ, 2016. – С. 261-263.

50. Смирнов С. А. Система обработки и формирования начального состояния маршрутизации метаданных в облачные антивирусные системы / А. А. Смирнов, С. А. Смирнов, А. К. Дидык // Проблеми кібербезпеки інформаційно-телекомунікаційних систем: зб. тез наук. -практ. конф., м. Київ, 10-11 березня 2016 р. – К.: КНУ ім. Тараса Шевченка, 2016. – С. 81-82.

51. Смирнов С. А. Алгоритм безопасной маршрутизации на базовом множестве путей передачи метаданных в программный сервер облачной антивирусной системы / А. А. Смирнов, С. А. Смирнов, А. К. Дидык // Інформаційна безпека та комп'ютерні технології (IS&CT): зб. тез міжнар. наук. -практ. конф., м. Кіровоград, 24-25 березня 2016 р. – Кіровоград: КНТУ, 2016. – С. 73.

52. Смирнов С. А. Исследование способа контроля линий связи телекоммуникационной системы для облачных антивирусов / А. А. Смирнов, С. А. Смирнов, А. К. Дидык // Збірник тез першої міжнародної науково-практичної конференції «Проблеми науково-технічного та правового

забезпечення кібербезпеки у сучасному світі» (ПНПЗК-2016), м. Харків, 30 березня - 1 квітня 2016 р. – Х.: НТУ «ХПІ», 2016. – С. 14.

53. Смирнов С. А. Разработка способа контроля линий связи телекоммуникационной системы для облачных антивирусов / А. А. Смирнов, С. А. Смирнов, А. К. Дидык // Матеріали XVIII міжнародного науково-практичного семінару «Комбінаторні конфігурації та їх застосування» (м. Кіровоград, 15-16 квітня 2016 р.). –Кіровоград: КНТУ, 2016. – С. 182-186.

54. Смирнов С. А. Разработка и исследование способа контроля линий связи телекоммуникационных сетей для облачных антивирусных систем / А. А. Смирнов, С. А. Смирнов, А. К. Дидык // Проблеми і перспективи розвитку ІТ-індустрії: VIII міжнар. наук.-практ. конф., м. Харків, 28-29 квітня 2016 р.: зб. тез. – Х.: ХНЕУ, 2016. – С. 48.

55. Смирнов С. А. Модель системы нейросетевых экспертов безопасной маршрутизации для облачных антивирусных систем / А. А. Смирнов, С. А. Смирнов, А. К. Дидык // Інформаційна та економічна безпека (INFECO-2016): зб. тез III міжнар. наук.-практ. конф., м. Харків, 28-30 кві. 2016 р. – Х.: ХННІ ДВНЗ «УБС», 2016. – С. 178-182.

56. Смирнов С. А. Метод безопасной маршрутизации метаданных в облачные антивирусные системы / А. А. Смирнов, С. А. Смирнов, А. К. Дидык // Сборник тезисов XII международной конференции «Стратегия качества в промышленности и образовании» (г. Варна, Болгария, 30 мая - 02 июня 2016 г.). – Варна: ТУВ, 2016. – С. 581-585.

57. Смирнов С. А. Оценка эффективности метода безопасной маршрутизации метаданных в облачные антивирусные системы / А. А. Смирнов, С. А. Смирнов, А. С. Коваленко // РадіоЕлектроніка та ІнфоКомунікації: зб. тез першої наук. - техн. конф., м. Київ, 11-16 вересня 2016 р. – К.: НТУУ «КПІ», 2016. – С. 17.

Додаток А
(обов'язковий)

Технічне завдання

Зміст

					КБР-123.21.0037.00.00.ТЗ		
Вим.	Арк.	№ документа	Підпис	Дата			
Розробив		Олійник А.О.			Програмне забезпечення системи аналізу мережевого трафіку	Літ. Б	Аркуш 1 / Аркушів 6
Перевірив		Доренський О.П.					
Н. Контр.		Гермак В.С.				ЦНТУ КІ-18-3СК	
Затв.		Смірнов О.А.					

1 Найменування та область застосування

Це технічне завдання розповсюджується на розробку системи аналізу мережевого трафіку.

2 Підстава для розробки

Підставою для розробки служить завдання на кваліфікаційну бакалаврську роботу, видане на кафедрі кібербезпеки та програмного забезпечення (нак. № 204-02 від 28.12.2020 року).

3 Мета та призначення розробки

Метою кваліфікаційної бакалаврської роботи є розробка програмного забезпечення системи аналізу мережевого трафіку.

4 Джерела розробки

Джерелом цієї кваліфікаційної бакалаврської роботи є стосовна до теми література і існуючі аналоги.

5 Технічні вимоги

5.1 Склад продукції

Складниками розробки є:

– вибір і обґрунтування методів реалізації проекту;

– розробка програмної частин системи, а також розробка взаємодії системи з ОС та з користувачем;

– розробка програми, що реалізує спроектовані алгоритми роботи системи.

5.2 Показники призначення

Система повинна забезпечувати:

– системи аналізу мережевого трафіку;

– цілісність даних у процесі роботи та при зберіганні;

– простий, інтуїтивно зрозумілий інтерфейс.

5.3 Вимоги до функціональних характеристик

Розроблене програмне забезпечення не повинно мати обмежень на версію драйверів та операційної системи.

5.4 Вимоги до архітектури

Компонент, що розробляється повинен використовувати системні засоби та апаратні засоби, що на даному етапі розвитку обчислювальної техніки найбільше поширені.

5.5 Вимоги до надійності

Програмні модулі написані по всім правилам, які стосуються стандартних викликів процедур, функцій, методів і форм, визначених технічною документацією на середовище розробки.

5.6 Умови експлуатації

Робочі місця користувачів ПЗ повинні задовольняти наступним умовам експлуатації:

– температура повітря: 19-20 град. по Цельсію;

– відносна вологість повітря до 80%;

– атмосферний тиск 107 кПа.

5.7 Вимоги до складу та параметрів технічних засобів

Програмне забезпечення повинно бути реалізоване на ПЕОМ архітектури IBM PC, працювати в ОС Windows XP/Vista/7/8/10 і з сумісними з цією платформою пристроями і прикладним програмним забезпечення.

5.8 Вимоги до інформаційної і програмної сумісності

Переносність програмного забезпечення повинна бути забезпечена за рахунок його реалізації стандартного інтерфейсу взаємодії з ОС, що працюють під управлінням ОС Windows XP/Vista/7/8/10.

5.8.1 Обладнання

Комп'ютер Intel® Celeron/8 Mb/1.2 Gb/SVGA 14" 1Mb або сумісні з ним.

5.8.2 Мова програмування

Середовище Embarcadero Delphi.

5.8.3 Вхідні дані

Опис алгоритму роботи запропонованої системи.

5.8.4 Вихідні дані

Робоча програма.

6 Вимоги до програмної документації

Програмна продукція повинна бути представлена у виді опису структури даних, схем та опису алгоритму, а також текстів вихідних модулів програмного забезпечення згідно ЄСПД .

7 Перелік документів, що розробляються

– Структурна схема системи – 1 аркуш.
– Функціональна схема системи – 1 аркуш.
– Діаграма процесів – 1 аркуш.
– Блок-схема алгоритму роботи програми – 2 аркуша.
– Пояснювальна записка – 71 аркуш.

8 Етапи розробки

8.1 Збір і обробка інформації по темі кваліфікаційної бакалаврської роботи. Постановка задачі на виконання кваліфікаційної бакалаврської роботи (складання ТЗ).

8.2 Проведення досліджень або експериментальних робіт для уточнення основних положень кваліфікаційної бакалаврської роботи.

8.3 Розробка функціональних схем, блок схем алгоритмів роботи програмного забезпечення.

8.4 Побудова схем взаємодії даних.

8.5 Створення прототипу ПЗ.

8.6 Віднаходження ПЗ, аналіз отриманих результатів.

8.7 Оформлення пояснювальної записки і виконання робіт по графічній частині.

11 Порядок контролю та приймання

11.1 Подання кваліфікаційної бакалаврської роботи на попередній захист 22.05.2020 р.

11.2 Подання кваліфікаційної бакалаврської роботи на захист 3.06.2020 р.

Додаток Б

(обов’язковий)

Міністерство освіти і науки України

Центральноукраїнський національний технічний університет

ЗАТВЕРДЖУЮ

Керівник кваліфікаційної бакалаврської роботи

_____ Доренський О.П.

Програмне забезпечення системи аналізу мережевого трафіку

Лістинг програми

Код документу 12

Носій: CD/DVD-диск

Загальна кількість аркушів: 36

Літера: РП

Кропивницький – 2021 року

Основна програма

Файл NetworkTrafficAnalysis.dpr основної програми

```
//Файл основної програми до якого підключаються відповідні //бібліотеки

program NetworkTrafficAnalysis;

uses
  Forms,
  IPHelper in ' IPHelper.pas' ,
  IPHLPAPI in ' IPHLPAPI.pas' ,
  MainFormUnit in ' MainFormUnit.pas'  {MainForm},
  TrafficUnit in ' TrafficUnit.pas' ;

{$R *.RES}

begin
  Application.Initialize;
  Application.CreateForm(TMainForm, MainForm);
  Application.Run;
end.
```

unit IPHelper – файл прогнозування трафіку

```
interface

uses
  Windows, Messages, SysUtils, Classes, Dialogs, IpHlpApi;

const
  NULL_IP       = '   0.  0.  0.  0' ;

//------перетворення визначених імен портів у сервісні імена------

type
  TWellKnownPort = record
    Prt: DWORD;
    Srv: string[20];
  end;

  PTTcpConnStatus = ^TTcpConnStatus;
  TTcpConnStatus = record
    LocalIP      : string;
    LocalPort    : string;
    RemoteIP     : string;
    RemotePort   : string;
    Status       : string;
  end;


const
    // для самих розповсюджених сервісів...
  WellKnownPorts: array[1..29] of TWellKnownPort
  = (
    ( Prt: 0; Srv: ' LOOPBACK' ),
    ( Prt: 7; Srv: ' ECHO'  ),      {Пінгування    }
    ( Prt: 9; Srv: ' DISCRD'  ),    { Відмова}
    ( Prt: 13; Srv: ' DAYTIM'  ),   {Час}
    ( Prt: 17; Srv: ' QOTD'  ),     {Вказник на час}
    ( Prt: 19; Srv: ' CHARGEN'  ),  {Генерація символів}
    ( Prt: 20; Srv: ' FTP '  ),     { File Transfer Protocol}
    ( Prt: 21; Srv: ' FTPC'  ),     { File Transfer Control Protocol}
    ( Prt: 23; Srv: ' TELNET'  ),   {TelNet}
    ( Prt: 25; Srv: ' SMTP'  ),     { Simple Mail Transfer Protocol}
    ( Prt: 37; Srv: ' TIME'  ),     { Часовий протокол }
    ( Prt: 43; Srv: ' WHOIS' ),     { WHO IS service  }
    ( Prt: 53; Srv: ' DNS '  ),     { Domain Name Service }
    ( Prt: 67; Srv: ' BOOTPS'  ),   { BOOTP Сервер }
    ( Prt: 68; Srv: ' BOOTPC'  ),   { BOOTP Клієнт }
    ( Prt: 69; Srv: ' TFTP'  ),     { стандартний FTP  }
    ( Prt: 70; Srv: ' GOPHER'  ),   { Gopher       }
    ( Prt: 79; Srv: ' FING'  ),     { Finger       }
    ( Prt: 80; Srv: ' HTTP'  ),     { HTTP         }
    ( Prt: 88; Srv: ' KERB'  ),     { Kerberos     }
    ( Prt: 109; Srv: ' POP2'  ),    { Post Office Protocol Version 2 }
    ( Prt: 110; Srv: ' POP3'  ),    { Post Office Protocol Version 3 }
    ( Prt: 119; Srv: ' NNTP'  ),    { Network News Transfer Protocol }
    ( Prt: 123; Srv: ' NTP '  ),    { Network Time protocol          }
    ( Prt: 135; Srv: ' LOCSVC' ),   { Локальні сервіси             }
    ( Prt: 137; Srv: ' NBNAME'  ),  { NETBIOS Імя сервісу         }
    ( Prt: 138; Srv: ' NBDGRAM'  ), { NETBIOS Сервіс датаграм      }
    ( Prt: 139; Srv: ' NBSESS'  ),  { NETBIOS Сеccіійний сервіс        }
    ( Prt: 161; Srv: ' SNMP'  )     { Simple Netw. Management Protocol }
    );


//----------перетворення ICMP кодів помилок у рядок-------------------------
```

```
const
  ICMP_ERROR_BASE = 11000;
  IcmpErr : array[1..22] of string =
  (
   ' IP_BUFFER_TOO_SMALL' ,' IP_DEST_NET_UNREACHABLE' , '
IP_DEST_HOST_UNREACHABLE' ,
   ' IP_PROTOCOL_UNREACHABLE' , ' IP_DEST_PORT_UNREACHABLE' , ' IP_NO_RESOURCES'
,
   ' IP_BAD_OPTION' ,' IP_HARDWARE_ERROR' , ' IP_PACKET_TOO_BIG' , '
IP_REQUEST_TIMED_OUT' ,
   ' IP_BAD_REQUEST' ,' IP_BAD_ROUTE' , ' IP_TTL_EXPIRED_TRANSIT' ,
   ' IP_TTL_EXPIRED_REASSEM' ,' IP_PARAMETER_PROBLEM' , ' IP_SOURCE_QUENCH' ,
   ' IP_OPTION_TOO_BIG' , ' IP_BAD_DESTINATION' ,' IP_ADDRESS_DELETED' ,
   ' IP_SPEC_MTU_CHANGE' , ' IP_MTU_CHANGE' , ' IP_UNLOAD'
  );


//----------перетворення різних величин до рядку ----------------//

  ARPEntryType  : array[1..4] of string = ( ' Other' , ' Invalid' ,
    ' Dynamic' , ' Static'
    );

  TCPConnState  :      // стани підключення TCP
    array[1..12] of string =
    ( ' closed' , ' listening' , ' syn_sent' ,
    ' syn_rcvd' , ' established' , ' fin_wait1' ,
    ' fin_wait2' , ' close_wait' , ' closing' ,
    ' last_ack' , ' time_wait' , ' delete_tcb'
    );

  TCPToAlgo     : array[1..4] of string =  // алгоритми часу TCP
    ( ' Const.Timeout' , ' MIL-STD-1778' ,
    ' Van Jacobson' , ' Other'  );

  IPForwTypes   : array[1..4] of string =   // IP пересилання методів
    ( ' other' , ' invalid' , ' local' , ' remote'  );

  IPForwProtos  : array[1..18] of string =  // IP пересилання протоколів
    ( ' OTHER' , ' LOCAL' , ' NETMGMT' , ' ICMP' , ' EGP' ,
    ' GGP' , ' HELO' , ' RIP' , ' IS_IS' , ' ES_IS' ,
    ' CISCO' , ' BBN' , ' OSPF' , ' BGP' , ' BOOTP' ,
    ' AUTO_STAT' , ' STATIC' , ' NOT_DOD'  );


//---------------експортуємі данні-----------------------------------------
-

// дані перетворюються у Tstrings для представлення на дісплей
procedure Get_AdaptersInfo( List: TStrings );
procedure Get_NetworkParams( List: TStrings );
procedure Get_ARPTable( List: TStrings );
procedure Get_TCPTable( List: TStrings );
procedure Get_TCPStatistics( List: TStrings );
procedure Get_UDPTable( List: TStrings );
procedure Get_UDPStatistics( List: TStrings );
procedure Get_IPAddrTable( List: TStrings );
procedure Get_IPForwardTable( List: TStrings );
procedure Get_IPStatistics( List: TStrings );
function Get_RTTAndHopCount( IPAddr: DWORD; MaxHops: Longint;
  var RTT: longint; var HopCount: longint ): integer;
procedure Get_ICMPStats( ICMPIn, ICMPOut: TStrings );
procedure Get_IfTable( NameList, ItemList: TStrings );
procedure Get_IfTableMIB( var MIBIfArray: TMIBIfArray );
procedure Get_IPAddrTableMIB( var IPAddrTable:TMibIPAddrArray  );



procedure Get_RecentDestIPs( List: TStrings );
```

```
// додаємо функцію
procedure Get_OpenConnections( List: TList );



// утіліти перетворення
function MacAddr2Str( MacAddr: TMacAddress; size: integer ): string;
function IpAddr2Str( IPAddr: DWORD ): string;
function Str2IpAddr( IPStr: string ): DWORD;
function Port2Str( nwoPort: DWORD ): string;
function Port2Wrd( nwoPort: DWORD ): DWORD;
function Port2Svc( Port: DWORD ): string;
function ICMPErr2Str( ICMPErrCode: DWORD) : string;

implementation

var
  RecentIPs     : TStringList;

//--------------Основні утіліти-----------------------------------------------

{ перетворюємо наступний токен у рядок, потім зчитуємо рядок та видаляємо його }
function NextToken( var s: string; Separator: char ): string;
var
  Sep_Pos       : byte;
begin
  Result := ' ' ;
  if length( s ) > 0 then begin
    Sep_Pos := pos( Separator, s );
    if Sep_Pos > 0 then begin
      Result := copy( s, 1, Pred( Sep_Pos ) );
      Delete( s, 1, Sep_Pos );
    end
    else begin
      Result := s;
      s := ' ' ;
    end;
  end;
end;

//------------------------------------------------------------------------------
{ перетворюємо цифрові MAC-адреса у ww-xx-yy-zz строку }
function MacAddr2Str( MacAddr: TMacAddress; size: integer ): string;
var
  i             : integer;
begin
  if Size = 0 then
  begin
    Result := ' 00-00-00-00-00-00' ;
    EXIT;
  end
  else Result := ' ' ;
  //
  for i := 1 to Size do
    Result := Result + IntToHex( MacAddr[i], 2 ) + ' -' ;
  Delete( Result, Length( Result ), 1 );
end;

//------------------------------------------------------------------------------
{ перетворення IP-адреси в мережний байт типу DWORD у крапкову десяткову строку}
function IpAddr2Str( IPAddr: DWORD ): string;
var
  i             : integer;
begin
  Result := ' ' ;
  for i := 1 to 4 do
  begin
    Result := Result + Format( ' %3d.' , [IPAddr and $FF] );
```

```
    IPAddr := IPAddr shr 8;
  end;
  Delete( Result, Length( Result ), 1 );
end;

//---------------------------------------------------------------------------
{ перетворення крапкової десяткової IP-адреси у мережний байт типу DWORD}
function Str2IpAddr( IPStr: string ): DWORD;
var
  i             : integer;
  Num           : DWORD;
begin
  Result := 0;
  for i := 1 to 4 do
  try
    Num := ( StrToInt( NextToken( IPStr, ' .'  ) ) ) shl 24;
    Result := ( Result shr 8 ) or Num;
  except
    Result := 0;
  end;

end;

//---------------------------------------------------------------------------
{ перетворення номера порту у мережний байт типу DWORD }
function Port2Wrd( nwoPort: DWORD ): DWORD;
begin
  Result := Swap( WORD( nwoPort ) );
end;

//---------------------------------------------------------------------------
{ перетворення номера порту у мережний байт типу string }
function Port2Str( nwoPort: DWORD ): string;
begin
  Result := IntToStr( Port2Wrd( nwoPort ) );
end;

//---------------------------------------------------------------------------
{ перетворення номера порту у сервіс ID }
function Port2Svc( Port: DWORD ): string;
var
  i             : integer;
begin
  Result := Format( ' %4d' , [Port] ); // у випадку відсутності порту
  for i := Low( WellKnownPorts ) to High( WellKnownPorts ) do
    if Port = WellKnownPorts[i].Prt then
    begin
      Result := WellKnownPorts[i].Srv;
      BREAK;
    end;
end;

//---------------------------------------------------------------------------
{ general,  fixed network parameters }
procedure Get_NetworkParams( List: TStrings );
var
  InfoSize      : Longint;
  ErrorCode     : DWORD;
  pBuf          : PChar;
begin
  if not Assigned( List ) then EXIT;
  List.Clear;
  InfoSize := 0;
  ErrorCode := GetNetworkParams( PTFixedInfo(pBuf), @InfoSize );
  GetMem( pBuf, InfoSize );
  ErrorCode := GetNetworkParams( PTFixedInfo(pBuf), @InfoSize );
  if ErrorCode = ERROR_SUCCESS then
    with PTFixedinfo(pBuf)^ do
    begin
```

```
      List.Add( ' HOSTNAME          : '  + string( Імя хосту ) );
      List.Add( ' DOMAIN            : '  + string( імя домену ) );
      List.Add( ' SCOPE             : '  + string( ScopeID ) );
      List.Add( ' NETBIOS NODE TYPE : '  + NETBIOSTypes[NodeType] );
      List.Add( ' ROUTING ENABLED   :'  + IntToStr( Дозволена маршрутизація ) );
      List.Add( ' PROXY   ENABLED   :'  + IntToStr( Дозволений Proxy ) );
      List.Add( ' DNS     ENABLED   :'  + IntToHex( Дозволений DNS,8 ) );
    end
  else
    List.Add( SysErrorMessage( ErrorCode ) );
  FreeMem(pBuf);
end;

//---------------------------------------------------------------------------
function ICMPErr2Str( ICMPErrCode: DWORD) : string;
var
 i : integer;
begin
   Result := ' UnknownError : '  + IntToStr( ICMPErrCode );
   dec( ICMPErrCode, ICMP_ERROR_BASE );
   if ICMPErrCode in [Low(ICMpErr)..High(ICMPErr)] then
     Result := ICMPErr[ ICMPErrCode];
end;

//---------------------------------------------------------------------------
//процедура отримання трафіку та занесення у таблицю
procedure Get_IfTable( NameList, ItemList: TStrings );
var
  IfRow         : TMibIfRow;
  i,
    Error,
    TableSize   : integer;
  pBuf          : PChar;
  NumEntries    : DWORD;
  sDescr,
    Temp        : string;
begin
  if (not Assigned( NameList ))
  or (not Assigned( ItemList ))  then EXIT;
  NameList.Clear;
  ItemList.Clear;
  TableSize := 0;
   // перший виклик: необхідно отримати розмір пам' яті
  Error := GetIfTable( PTMibIfTable( pBuf ), @TableSize, false );
  if Error <> ERROR_INSUFFICIENT_BUFFER then
    EXIT;
  GetMem( pBuf, TableSize );

   // отримуємо таблицю показчиків
  Error := GetIfTable( PTMibIfTable( pBuf ), @TableSize, false );
  if Error = NO_ERROR then
  begin
    NumEntries := PTMibIfTable( pBuf )^.dwNumEntries;
    if NumEntries > 0 then
    begin
      inc( pBuf, SizeOf( NumEntries ) );
      for i := 1 to NumEntries do
      begin
        IfRow := PTMibIfRow( pBuf )^;
        with IfRow do
        begin
          SetLength( sDescr, dwDescrLen );
          move( bDescr, sDescr[1], Length( sDescr ) );
          sDescr := trim( sDescr );
          NameList.Add( sDescr );
          ItemList.Add( Format( ' %0.8x|%2d| %16s| %4d| %8d| %8d| %8d' ,
            [dwIndex, dwType,
            MacAddr2Str( TMacAddress( bPhysAddr ), dwPhysAddrLen )
              , dwMTU, dwSpeed,
```

```
              dwInOctets, dwOutOctets,
              dwOPerStatus] )
              );
        end;
        inc( pBuf, SizeOf( IfRow ) );
      end;
    end
    else  begin
      NameList.Add( ' без даних' );
      ItemList.Add( ' немає даних'  );
    end;
  end
  else begin
    NameList.Add( ' Oops' );
    ItemList.Add( SysErrorMessage( GetLastError ) );
  end;
  dec( pBuf, SizeOf( DWORD ) + NumEntries * SizeOf( IfRow ) );
  FreeMem( pBuf );
end;


//-------------------------------------------------------------------------
//Занесення даних у таблицю MIB
procedure Get_IfTableMIB( var MIBIfArray: TMIBIfArray );
var
  i,
    Error,
    TableSize   : integer;
  pBuf          : PChar;
  NumEntries    : DWORD;
  sDescr,
    Temp        : string;
begin
  TableSize := 0;
   // перший виклик: необхідно отримати розмір пам' яті
  Error := GetIfTable( PTMibIfTable( pBuf ), @TableSize, false );
  if Error <> ERROR_INSUFFICIENT_BUFFER then
    EXIT;
  GetMem( pBuf, TableSize );

   // отримуємо таблицю показчиків
  Error := GetIfTable( PTMibIfTable( pBuf ), @TableSize, false );
  if Error = NO_ERROR then
  begin
    NumEntries := PTMibIfTable( pBuf )^.dwNumEntries;
    if NumEntries > 0 then
    begin
      SetLength( MIBIfArray, NumEntries );
      inc( pBuf, SizeOf( NumEntries ) );
      for i := 0 to pred(NumEntries) do
      begin
        MIBIfArray[i] := PTMibIfRow( pBuf )^;
        inc( pBuf, SizeOf( TMIBIfRow ) );
      end;
    end
  end;
  dec( pBuf, SizeOf( DWORD ) + NumEntries * SizeOf( TMIBIfRow ) );
  FreeMem( pBuf );
end;



//-------------------------------------------------------------------------
//Заносимо дінні про адреси у таблиці MIB
procedure Get_IPAddrTableMIB( var IPAddrTable:TMibIPAddrArray  );
var
  IPAddrRow     : TMibIPAddrRow;
  TableSize     : DWORD;
  ErrorCode     : DWORD;
```

```
  i             : integer;
  pBuf          : PChar;
  NumEntries    : DWORD;
begin
  TableSize := 0; ;
  // перший виклик: необхідно отримати розмір таблиці у БД
  ErrorCode := GetIpAddrTable( PTMibIPAddrTable( pBuf ), @TableSize, true );
  if Errorcode <> ERROR_INSUFFICIENT_BUFFER then
    EXIT;

  GetMem( pBuf, TableSize );
  // заносимо у таблицю
  ErrorCode := GetIpAddrTable( PTMibIPAddrTable( pBuf ), @TableSize, true );
  if ErrorCode = NO_ERROR then
  begin
    NumEntries := PTMibIPAddrTable( pBuf )^.dwNumEntries;
    if NumEntries > 0 then
    begin
      SetLength( IPAddrTable, NumEntries);
      inc( pBuf, SizeOf( DWORD ) );
      for i := 1 to NumEntries do
      begin
        IPAddrTable[ i-1 ] := PTMIBIPAddrRow( pBuf )^;
        inc( pBuf, SizeOf( TMIBIPAddrRow ) );
      end;
    end;
  end;

  // відновлюємо показчик !
  dec( pBuf, SizeOf( DWORD ) + NumEntries * SizeOf( IPAddrRow ) );
  FreeMem( pBuf );
end;



//* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *//
procedure Get_AdaptersInfo( List: TStrings );
var
  Error,
  BufLen      : DWORD;
  P             : Pointer;
  AdapterInfo   : PTIP_ADAPTER_INFO;
  Descr,
  LocalIP,
  GatewayIP,
  DHCPIP      : string;
begin
  if not Assigned( List ) then EXIT;
  List.Clear;
  BufLen := SizeOf( AdapterInfo^ );
  New( AdapterInfo );
  Error := GetAdaptersInfo( AdapterInfo, @BufLen );
  P := AdapterInfo;
  if Error = NO_ERROR then
  begin
    while P <> nil do
      with TIP_ADAPTER_INFO(P^) do
      begin
        SetLength( Descr, SizeOf( Description ) );
        Descr := Trim( string( Description ) );
        //
        if IPAddressList.IpAddress[1] <> #0 then
          LocalIP := IPAddressList.IpAddress
        else
          LocalIP := NULL_IP;
        //
        if GateWayList.IPAddress[1] <> #0 then
          GateWayIP := GatewayList.IPAddress
        else
```

```
          GateWayIP := NULL_IP;
        //
        if DHCPServer.IPAddress[1] <> #0 then
          DHCPIP := DHCPServer.IPAddress
        else
          DHCPIP := NULL_IP;

        List.Add( Descr );
        List.Add( Format(
          ' %8.8x|%6s|%16s|%2d|%16s|%16s|%16s' ,
          [Index, AdaptTypes[aType],
          MacAddr2Str( TMacAddress( Address ), AddressLength ),
            DHCPEnabled, LocalIP, GatewayIP, DHCPIP] )
            );
        List.Add( '   ' );
        P := Next;  //  TIP_ADAPTER_INFO(P^).Next  points to next entry
      end // with
    end // while
  else
    List.Add( SysErrorMessage( Error ) );
  Dispose( AdapterInfo );
end;



//---------------------------------------------------------------------------
{ записуємо час завантаження трафіка мережі IP }
function Get_RTTAndHopCount( IPAddr: DWORD; MaxHops: Longint; var RTT: Longint;
  var HopCount: Longint ): integer;
begin
  if not GetRTTAndHopCount( IPAddr, @HopCount, MaxHops, @RTT ) then
  begin
    Result := GetLastError;
    RTT := -1; // Розташування BAD_HOST_NAME,й.. т.і.
    HopCount := -1;
  end
  else
    Result := NO_ERROR;
end;



//---------------------------------------------------------------------------
{ ARP-таблиця записує відношення між IP та MAC-адресам.
}
procedure Get_ARPTable( List: TStrings );
var
  IPNetRow    : TMibIPNetRow;
  TableSize   : DWORD;
  NumEntries  : DWORD;
  ErrorCode   : DWORD;
  i           : integer;
  pBuf        : PChar;
begin
  if not Assigned( List ) then EXIT;
  List.Clear;
  // перший виклик: необхідно отримати розмір таблиці у БД
  TableSize := 0;
  ErrorCode := GetIPNetTable( PTMIBIpNetTable( pBuf ), @TableSize, false );
  //
  if ErrorCode = ERROR_NO_DATA then
  begin
    List.Add( '  ARP-cache empty.'  );
    EXIT;
  end;
  // заносимо у таблицю
  GetMem( pBuf, TableSize );
  ErrorCode := GetIpNetTable( PTMIBIPNetTable( pBuf ), @TableSize, false );
```

```
  if ErrorCode = NO_ERROR then
  begin
    NumEntries := PTMIBIPNetTable( pBuf )^.dwNumEntries;
    if NumEntries > 0 then //
    begin
      inc( pBuf, SizeOf( DWORD ) ); // записуємо розмір останьої таблиці
      for i := 1 to NumEntries do
      begin
        IPNetRow := PTMIBIPNetRow( PBuf )^;
        with IPNetRow do
          List.Add( Format( ' %8x | %12s | %16s| %10s' ,
                            [dwIndex, MacAddr2Str( bPhysAddr, dwPhysAddrLen ),
                            IPAddr2Str( dwAddr ), ARPEntryType[dwType]
                            ]));
        inc( pBuf, SizeOf( IPNetRow ) );
      end;
    end
    else
      List.Add( '  ARP-кеш пустий.'  );
  end
  else
    List.Add( SysErrorMessage( ErrorCode ) );

  // we _must_ restore pointer!
  dec( pBuf, SizeOf( DWORD ) + NumEntries * SizeOf( IPNetRow ) );
  FreeMem( pBuf );

end;


//---------------------------------------------------------------------------
procedure Get_TCPTable( List: TStrings );
var
  TCPRow        : TMIBTCPRow;
  i,
    NumEntries  : integer;
  TableSize     : DWORD;
  ErrorCode     : DWORD;
  DestIP        : string;
  pBuf          : PChar;
begin
  if not Assigned( List ) then EXIT;
  List.Clear;
  RecentIPs.Clear;
  // перший виклик: необхідно отримати розмір таблиці у БД
  TableSize := 0;
  ErrorCode := GetTCPTable( PTMIBTCPTable( pBuf ), @TableSize, true );
  if Errorcode <> ERROR_INSUFFICIENT_BUFFER then
    EXIT;

  // резервуємо пам' ять. Викликаємо знову
  GetMem( pBuf, TableSize );
  // заносимо у таблицю
  ErrorCode := GetTCPTable( PTMIBTCPTable( pBuf ), @TableSize, true );
  if ErrorCode = NO_ERROR then
  begin

    NumEntries := PTMIBTCPTable( pBuf )^.dwNumEntries;
    if NumEntries > 0 then
    begin
      inc( pBuf, SizeOf( DWORD ) ); // беремо розмір останьої таблиці
      for i := 1 to NumEntries do
      begin
        TCPRow := PTMIBTCPRow( pBuf )^; // беремо наступний запис з
навантаженістю мережного трафіку
        with TCPRow do
        begin
          if dwRemoteAddr = 0 then
            dwRemotePort := 0;
```

```
          DestIP := IPAddr2Str( dwRemoteAddr );
          List.Add(
            Format( ' %15s : %-7s|%15s : %-7s| %-16s' ,
            [IpAddr2Str( dwLocalAddr ),
            Port2Svc( Port2Wrd( dwLocalPort ) ),
              DestIP,
              Port2Svc( Port2Wrd( dwRemotePort ) ),
              TCPConnState[dwState]
              ] ) );
         //
            if (not ( dwRemoteAddr = 0 ))
            and ( RecentIps.IndexOf(DestIP) = -1 ) then
               RecentIPs.Add( DestIP );
        end;
        inc( pBuf, SizeOf( TMIBTCPRow ) );
      end;
    end;
  end
  else
    List.Add( SyserrorMessage( ErrorCode ) );
  dec( pBuf, SizeOf( DWORD ) + NumEntries * SizeOf( TMibTCPRow ) );
  FreeMem( pBuf );
end;


//---------------------------------------------------------------------------
// Опитуємо відкриті підключення до мережі
procedure Get_OpenConnections( List: TList );
var
  TCPRow        : TMIBTCPRow;
  i,
    NumEntries  : integer;
  TableSize     : DWORD;
  ErrorCode     : DWORD;
  DestIP        : string;
  pBuf          : PChar;
  CStat         : PTTcpConnStatus;
begin
  if not Assigned( List ) then EXIT;
  List.Clear;
  // перший виклик: необхідно отримати розмір таблиці у БД
  TableSize := 0;
  ErrorCode := GetTCPTable( PTMIBTCPTable( pBuf ), @TableSize, true );
  if Errorcode <> ERROR_INSUFFICIENT_BUFFER then
    EXIT;

  // резервуємо пам' ять. Викликаємо знову
  GetMem( pBuf, TableSize );
  // заносимо у таблицю
  ErrorCode := GetTCPTable( PTMIBTCPTable( pBuf ), @TableSize, true );
  if ErrorCode = NO_ERROR then
  begin
    NumEntries := PTMIBTCPTable( pBuf )^.dwNumEntries;
    if NumEntries > 0 then
    begin
      inc( pBuf, SizeOf( DWORD ) ); // беремо розмір останьої таблиці
      for i := 1 to NumEntries do
      begin
        TCPRow := PTMIBTCPRow( pBuf )^; // беремо наступний запис з
навантаженістю мережного трафіку
        with TCPRow do
          if dwState in [2,5] then   //
          begin
            New( CStat );
            CStat^.LocalIP   := IPAddr2Str( dwLocalAddr );
            CStat^.LocalPort := Port2Svc( Port2Wrd( dwLocalPort ));
            if dwRemoteAddr <> 0 then
            begin
              CStat^.RemoteIP     := IPAddr2Str( dwRemoteAddr );
```

```
            CStat^.RemotePort   := Port2Svc( Port2Wrd( dwRemotePort ));
          end
          else begin
            CStat^.RemoteIP   := ' ...' ;
            CStat^.RemotePort := ' ...' ;
          end;
          CStat^.Status       := TCPConnState[dwState];
          List.Add( CStat );
        end;
      inc( pBuf, SizeOf( TMIBTCPRow ) );
    end;
  end;
  dec( pBuf, SizeOf( DWORD ) + NumEntries * SizeOf( TMibTCPRow ) );
  FreeMem( pBuf );
end;



//-------------------------------------------------------------------------
//Записуємо статистику трафіка по TCP
procedure Get_TCPStatistics( List: TStrings );
var
  TCPStats      : TMibTCPStats;
  ErrorCode     : DWORD;
begin
  if not Assigned( List ) then EXIT;
  List.Clear;
  ErrorCode := GetTCPStatistics( @TCPStats );
  if ErrorCode = NO_ERROR then
    with TCPStats do
    begin
      List.Add( ' Алгоритм повторної передачі  :'  + TCPToAlgo[dwRTOAlgorithm]
);
      List.Add( ' Мініміальний час            :'  + IntToStr( dwRTOMin ) + '  ms'
);
      List.Add( ' Максимальний час            :'  + IntToStr( dwRTOMax ) + '  ms'
);
      List.Add( ' Максимальна кількість підключень :'  + IntToStr(
dwRTOAlgorithm ) );
      List.Add( ' Активні підключення             :'  + IntToStr( dwActiveOpens
) );
      List.Add( ' Пасивні підключення            :'  + IntToStr( dwPassiveOpens
) );
      List.Add( ' Помилка невдалого відкриття     :'  + IntToStr( dwAttemptFails
) );
      List.Add( ' Скидання встановленого підключення  :'  + IntToStr(
dwEstabResets ) );
      List.Add( ' Поточне встановлене підключення.:'  + IntToStr( dwCurrEstab )
);
      List.Add( ' Отриманий сегмент        :'  + IntToStr( dwInSegs ) );
      List.Add( ' Відправлений сегмент            :'  + IntToStr( dwOutSegs ) );
      List.Add( ' Переправлений сегмент   :'  + IntToStr( dwReTransSegs ) );
      List.Add( ' Помилка входження         :'  + IntToStr( dwInErrs ) );
      List.Add( ' Скидання виходу          :'  + IntToStr( dwOutRsts ) );
      List.Add( ' Сукупні зв'язки    :'  + IntToStr( dwNumConns ) );
    end
  else
    List.Add( SyserrorMessage( ErrorCode ) );

end;

//-------------------------------------------------------------------------

//Запис часу та завантаженості трафіку по UDP
procedure Get_UDPTable( List: TStrings );
var
  UDPRow        : TMIBUDPRow;
  i,
```

```
    NumEntries  : integer;
  TableSize     : DWORD;
  ErrorCode     : DWORD;
  pBuf          : PChar;
begin
  if not Assigned( List ) then EXIT;
  List.Clear;

  // перший виклик: необхідно отримати розмір таблиці у БД
  TableSize := 0;
  ErrorCode := GetUDPTable( PTMIBUDPTable( pBuf ), @TableSize, true );
  if Errorcode <> ERROR_INSUFFICIENT_BUFFER then
    EXIT;

  // резервуємо пам’ ять. Викликаємо знову
  GetMem( pBuf, TableSize );

  // заносимо у таблицю
  ErrorCode := GetUDPTable( PTMIBUDPTable( pBuf ), @TableSize, true );
  if ErrorCode = NO_ERROR then
  begin
    NumEntries := PTMIBUDPTable( pBuf )^.dwNumEntries;
    if NumEntries > 0 then
    begin
      inc( pBuf, SizeOf( DWORD ) ); // беремо розмір останьої таблиці
      for i := 1 to NumEntries do
      begin
        UDPRow := PTMIBUDPRow( pBuf )^; // беремо наступний запис з
навантаженістю мережного трафіку
        with UDPRow do
          List.Add( Format( ‘ %15s : %-6s’ ,
            [IpAddr2Str( dwLocalAddr ),
            Port2Svc( Port2Wrd( dwLocalPort ) )
              ] ) );
        inc( pBuf, SizeOf( TMIBUDPRow ) );
      end;
    end
    else
      List.Add( ‘ без даних.’  );
  end
  else
    List.Add( SyserrorMessage( ErrorCode ) );
  dec( pBuf, SizeOf( DWORD ) + NumEntries * SizeOf( TMibUDPRow ) );
  FreeMem( pBuf );
end;

//---------------------------------------------------------------------------
procedure Get_IPAddrTable( List: TStrings );

//Запис часу та завантаженості трафіку по IP

var
  IPAddrRow     : TMibIPAddrRow;
  TableSize     : DWORD;
  ErrorCode     : DWORD;
  i             : integer;
  pBuf          : PChar;
  NumEntries    : DWORD;
begin
  if not Assigned( List ) then EXIT;
  List.Clear;
  TableSize := 0; ;
  // перший виклик: необхідно отримати розмір таблиці у БД
  ErrorCode := GetIpAddrTable( PTMibIPAddrTable( pBuf ), @TableSize, true );
  if Errorcode <> ERROR_INSUFFICIENT_BUFFER then
    EXIT;

  GetMem( pBuf, TableSize );
  // заносимо у таблицю
```

```
  ErrorCode := GetIpAddrTable( PTMibIPAddrTable( pBuf ), @TableSize, true );
  if ErrorCode = NO_ERROR then
  begin
    NumEntries := PTMibIPAddrTable( pBuf )^.dwNumEntries;
    if NumEntries > 0 then
    begin
      inc( pBuf, SizeOf( DWORD ) );
      for i := 1 to NumEntries do
      begin
        IPAddrRow := PTMIBIPAddrRow( pBuf )^;
        with IPAddrRow do
          List.Add( Format( ' %8.8x|%15s|%15s|%15s|%8.8d' ,
            [dwIndex,
            IPAddr2Str( dwAddr ),
              IPAddr2Str( dwMask ),
              IPAddr2Str( dwBCastAddr ),
              dwReasmSize
              ] ) );
        inc( pBuf, SizeOf( TMIBIPAddrRow ) );
      end;
    end
    else
      List.Add( ' без даних.'  );
  end
  else
    List.Add( SysErrorMessage( ErrorCode ) );

  // відновлюємо показчик !
  dec( pBuf, SizeOf( DWORD ) + NumEntries * SizeOf( IPAddrRow ) );
  FreeMem( pBuf );
end;


(*
//---------------------------------------------------------------------------
//Запис IP адресів до MIB

procedure Get_IPAddrTableMIB( var IPAddrTable:TMibIPAddrArray  );
var
  IPAddrRow     : TMibIPAddrRow;
  TableSize     : DWORD;
  ErrorCode     : DWORD;
  i             : integer;
  pBuf          : PChar;
  NumEntries    : DWORD;
begin
  TableSize := 0; ;
  // перший виклик: необхідно отримати розмір таблиці у БД
  ErrorCode := GetIpAddrTable( PTMibIPAddrTable( pBuf ), @TableSize, true );
  if Errorcode <> ERROR_INSUFFICIENT_BUFFER then
   EXIT;

  GetMem( pBuf, TableSize );
  // заносимо у таблицю
  ErrorCode := GetIpAddrTable( PTMibIPAddrTable( pBuf ), @TableSize, true );
  if ErrorCode = NO_ERROR then
  begin
    NumEntries := PTMibIPAddrTable( pBuf )^.dwNumEntries;
    if NumEntries > 0 then
    begin
      SetLength( IPAddrTable, NumEntries);
      inc( pBuf, SizeOf( DWORD ) );
      for i := 1 to NumEntries do
      begin
        IPAddrTable[ i-1 ] := PTMIBIPAddrRow( pBuf )^;
        inc( pBuf, SizeOf( TMIBIPAddrRow ) );
      end;
    end;
  end;
```

```
  // відновлюємо показчик !
  dec( pBuf, SizeOf( DWORD ) + NumEntries * SizeOf( IPAddrRow ) );
  FreeMem( pBuf );
end;
*)

//----------------------------------------------------------------------------
{ отримуємо данні з таблиць маршрутизації }
procedure Get_IPForwardTable( List: TStrings );
var
  IPForwRow     : TMibIPForwardRow;
  TableSize     : DWORD;
  ErrorCode     : DWORD;
  i             : integer;
  pBuf          : PChar;
  NumEntries    : DWORD;
begin
  if not Assigned( List ) then EXIT;
  List.Clear;
  TableSize := 0;

  // перший виклик: необхідно отримати розмір таблиці у БД
  ErrorCode := GetIpForwardTable( PTMibIPForwardTable( pBuf ), @TableSize, true
    );
  if Errorcode <> ERROR_INSUFFICIENT_BUFFER then
    EXIT;

  // заносимо у таблицю
  GetMem( pBuf, TableSize );
  ErrorCode := GetIpForwardTable( PTMibIPForwardTable( pBuf ), @TableSize, true
    );
  if ErrorCode = NO_ERROR then
  begin
    NumEntries := PTMibIPForwardTable( pBuf )^.dwNumEntries;
    if NumEntries > 0 then
    begin
      inc( pBuf, SizeOf( DWORD ) );
      for i := 1 to NumEntries do
      begin
        IPForwRow := PTMibIPForwardRow( pBuf )^;
        with IPForwRow do
          List.Add( Format(
            ' %15s|%15s|%15s|%8.8x|%7s|   %5.5d|    %7s|        %2.2d' ,
            [IPAddr2Str( dwForwardDest ),
            IPAddr2Str( dwForwardMask ),
             IPAddr2Str( dwForwardNextHop ),
             dwForwardIFIndex,
             IPForwTypes[dwForwardType],
             dwForwardNextHopAS,
             IPForwProtos[dwForwardProto],
             dwForwardMetric1
             ] ) );
        inc( pBuf, SizeOf( TMibIPForwardRow ) );
      end;
    end
    else
      List.Add( ' без даних.'  );
  end
  else
    List.Add( SysErrorMessage( ErrorCode ) );
  dec( pBuf, SizeOf( DWORD ) + NumEntries * SizeOf( TMibIPForwardRow ) );
  FreeMem( pBuf );
end;

//----------------------------------------------------------------------------

// Надання статистики по IP
procedure Get_IPStatistics( List: TStrings );
```

```
var
  IPStats       : TMibIPStats;
  ErrorCode     : integer;
begin
  if not Assigned( List ) then EXIT;
  ErrorCode := GetIPStatistics( @IPStats );
  if ErrorCode = NO_ERROR then
  begin
    List.Clear;
    with IPStats do
    begin
      if dwForwarding = 1 then
        List.add( ' Пересилання дозволено       : '  + ' Так'  )
      else
        List.add( ' Пересилання дозволено       : '  + ' Ні'  );
      List.add( ' Вбудований TTL              : '  + inttostr( dwDefaultTTL ) );
      List.add( ' Отримана датаграмма       : '  + inttostr( dwInReceives ) );
      List.add( ' Помилки заголовку     (У)  : '  + inttostr( dwInHdrErrors ) );
      List.add( ' Помилка адреси    (У)  : '  + inttostr( dwInAddrErrors ) );
      List.add( ' Невідомий протокол  (У)  : '  + inttostr( dwInUnknownProtos )
);
      List.add( ' Відхілені датаграми     : '  + inttostr( dwInDiscards ) );
      List.add( ' Датаграми встановлені    : '  + inttostr( dwInDelivers ) );
      List.add( ' Зовнішній запит             : '  + inttostr( dwOutRequests ) );
      List.add( ' Маршрут відхилений      : '  + inttostr( dwRoutingDiscards )
);
      List.add( ' Немає маршруту          (Out): '  + inttostr( dwOutNoRoutes )
);
      List.add( ' Перебирання часу     : '  + inttostr( dwReasmTimeOut ) );
      List.add( ' Перебирання запитів     : '  + inttostr( dwReasmReqds ) );
      List.add( ' Повний перебор : '  + inttostr( dwReasmOKs ) );
      List.add( ' Помилка перебору     : '  + inttostr( dwReasmFails ) );
      List.add( ' Повна фрагментація: '  + inttostr( dwFragOKs ) );
      List.add( ' Помилка фрагментації    : '  + inttostr( dwFragFails ) );
      List.add( ' Датаграму фрагментовано     : '  + inttostr( dwFRagCreates ) );
      List.add( ' Кількість інтерфейсів     : '  + inttostr( dwNumIf ) );
      List.add( ' Кількість IP-адрес  : '  + inttostr( dwNumAddr ) );
      List.add( ' Маршрут у таблиці маршрутизатора  : '  + inttostr( dwNumRoutes
) );
    end;
  end
  else
    List.Add( SysErrorMessage( ErrorCode ) );
end;

//---------------------------------------------------------------------------
// Надання статистики по UDP

procedure Get_UdpStatistics( List: TStrings );
var
  UdpStats      : TMibUDPStats;
  ErrorCode     : integer;
begin
  if not Assigned( List ) then EXIT;
  ErrorCode := GetUDPStatistics( @UdpStats );
  if ErrorCode = NO_ERROR then
  begin
    List.Clear;
    with UDPStats do
    begin
      List.add( ' Датаграми (У)     : '  + inttostr( dwInDatagrams ) );
      List.add( ' Датаграми (З)    : '  + inttostr( dwOutDatagrams ) );
      List.add( 'Немає портів           : '  + inttostr( dwNoPorts ) );
      List.add( ' Помилки     (У)     : '  + inttostr( dwInErrors ) );
      List.add( ' UDP список портів  : '  + inttostr( dwNumAddrs ) );
    end;
  end
  else
    List.Add( SysErrorMessage( ErrorCode ) );
```

```
end;

//---------------------------------------------------------------------------
// Надання статистики по ICMP

procedure Get_ICMPStats( ICMPIn, ICMPOut: TStrings );
var
  ErrorCode     : DWORD;
  ICMPStats     : PTMibICMPInfo;
begin
  if ( ICMPIn = nil ) or ( ICMPOut = nil ) then EXIT;
  ICMPIn.Clear;
  ICMPOut.Clear;
  New( ICMPStats );
  ErrorCode := GetICMPStatistics( ICMPStats );
  if ErrorCode = NO_ERROR then
  begin
    with ICMPStats.InStats do
    begin
      ICMPIn.Add( ' Повідомлення прийнято    : '  + IntToStr( dwMsgs ) );
      ICMPIn.Add( ' Помилка               : '  + IntToStr( dwErrors ) );
      ICMPIn.Add( ' Розташування недосягнене    : '  + IntToStr( dwDestUnreachs
) );
      ICMPIn.Add( ' Час перевищений        : '  + IntToStr( dwTimeEcxcds ) );
      ICMPIn.Add( ' Проблеми з параметрами      : '  + IntToStr( dwParmProbs )
);
      ICMPIn.Add( ' Джерело відключене        : '  + IntToStr( dwSrcQuenchs ) );
      ICMPIn.Add( ' Перепризначення          : '  + IntToStr( dwRedirects ) );
      ICMPIn.Add( ' Exo запит        : '  + IntToStr( dwEchos ) );
      ICMPIn.Add( ' Exo повтор         : '  + IntToStr( dwEchoReps ) );
      ICMPIn.Add( ' Запит мітки часу   : '  + IntToStr( dwTimeStamps ) );
      ICMPIn.Add( ' Повтор мітки часу    : '  + IntToStr( dwTimeStampReps ) );

      ICMPIn.Add( ' Запит адреси маски : '  + IntToStr( dwAddrMasks ) );
      ICMPIn.Add( ' Повтор адреси маски   : '  + IntToStr( dwAddrReps ) );
    end;
     //
    with ICMPStats^.OutStats do
    begin
      ICMPOut.Add( 'Повідомлення відправлено: '  + IntToStr( dwMsgs ) );
      ICMPOut.Add( 'Помилка               : '  + IntToStr( dwErrors ) );
      ICMPOut.Add( 'Розташування недосягнене    : '  + IntToStr( dwDestUnreachs
) );
      ICMPOut.Add( 'Час перевищений        : '  + IntToStr( dwTimeEcxcds ) );
      ICMPOut.Add( 'Проблеми з параметрами      : '  + IntToStr( dwParmProbs )
);
      ICMPOut.Add( 'Джерело відключене        : '  + IntToStr( dwSrcQuenchs ) );
      ICMPOut.Add( 'Перепризначення          : '  + IntToStr( dwRedirects ) );
      ICMPOut.Add( 'Exo запит        : '  + IntToStr( dwEchos ) );
      ICMPOut.Add( 'Exo повтор         : '  + IntToStr( dwEchoReps ) );
      ICMPOut.Add( 'Запит мітки часу   : '  + IntToStr( dwTimeStamps ) );
      ICMPOut.Add( 'Повтор мітки часу    : '  + IntToStr( dwTimeStampReps ) );
      ICMPOut.Add( 'Запит адреси маски: '  + IntToStr( dwAddrMasks ) );
      ICMPOut.Add( 'Повтор адреси маски   : '  + IntToStr( dwAddrReps ) );
    end;
  end
  else
    IcmpIn.Add( SysErrorMessage( ErrorCode ) );
  Dispose( ICMPStats );
end;

//---------------------------------------------------------------------------
procedure Get_RecentDestIPs( List: TStrings );
begin
  if Assigned( List ) then
    List.Assign( RecentIPs )
end;

initialization
```

```
  RecentIPs := TStringList.Create;
finalization
  RecentIPs.Free;
end.
```

unit IPHLPAPI – бібліотека API функцій для роботи з IP мережею

```
interface
uses
  Windows, winsock;
const

  VERSION       = '1.3';


const
  ANY_SIZE      = 1;
  MAX_ADAPTER_DESCRIPTION_LENGTH = 128; // arb.
  MAX_ADAPTER_NAME_LENGTH = 256; // arb.
  MAX_ADAPTER_ADDRESS_LENGTH = 8; // arb.
  DEFAULT_MINIMUM_ENTITIES = 32; // arb.
  MAX_HOSTNAME_LEN = 128; // arb.
  MAX_DOMAIN_NAME_LEN = 128; // arb.
  MAX_SCOPE_ID_LEN = 256; // arb.

 // Типи ( NETBIOS)
  BROADCAST_NODETYPE = 1;
  PEER_TO_PEER_NODETYPE = 2;
  MIXED_NODETYPE = 4;
  HYBRID_NODETYPE = 8;

  NETBIOSTypes  : array[0..8] of string[20] =
    ( 'UNKNOWN', 'BROADCAST', 'PEER_TO_PEER', '', 'MIXED', '', '', '', 'HYBRID'
      );

 // Типи адаптерів
  IF_OTHER_ADAPTERTYPE = 0;
  IF_ETHERNET_ADAPTERTYPE = 1;
  IF_TOKEN_RING_ADAPTERTYPE = 2;
  IF_FDDI_ADAPTERTYPE = 3;
  IF_PPP_ADAPTERTYPE = 4;
  IF_LOOPBACK_ADAPTERTYPE = 5;
  IF_SLIP_ADAPTERTYPE = 6;
 //
  AdaptTypes    : array[0..6] of string[10] =
    ( 'інший', 'ethernet', 'tokenring', 'FDDI', 'PPP', 'loopback', 'SLIP' );


  MAX_INTERFACE_NAME_LEN = 256; { mrapi.h }
  MAXLEN_PHYSADDR = 8; { iprtrmib.h }
  MAXLEN_IFDESCR = 256; { --"---     }

//-----------------------------------------------------------------------

type
  TMacAddress = array[1..MAX_ADAPTER_ADDRESS_LENGTH] of byte;

  //------Структура IP-адрес -------------------------------------------------

  PTIP_ADDRESS_STRING = ^TIP_ADDRESS_STRING;
  TIP_ADDRESS_STRING = array[0..15] of char; //  IP в xxx.xxx.xxx.xxx рядок
  //
  PTIP_ADDR_STRING = ^TIP_ADDR_STRING;
  TIP_ADDR_STRING = packed record //
    Next: PTIP_ADDR_STRING;
    IpAddress: TIP_ADDRESS_STRING;
    IpMask: TIP_ADDRESS_STRING;
    Context: DWORD;
  end;

//----------Fixed Info структура-----------------------------------------
```

```
  PTFixedInfo = ^TFixedInfo;
  TFixedInfo = packed record
    HostName: array[0..MAX_HOSTNAME_LEN + 4] of char;
    DomainName: array[0..MAX_DOMAIN_NAME_LEN + 4] of char;
    CurrentDNSServer: PTIP_ADDR_STRING;
    DNSServerList: TIP_ADDR_STRING;
    NodeType: UINT;
    ScopeID: array[0..MAX_SCOPE_ID_LEN + 4] of char;
    EnableRouting: UINT;
    EnableProxy: UINT;
    EnableDNS: UINT;
  end;

//----------INTERFACE структура-------------------------------------------------


  PTMibIfRow = ^TMibIfRow;
  TMibIfRow = packed record
    wszName: array[1..MAX_INTERFACE_NAME_LEN] of WCHAR;
    dwIndex: DWORD;
    dwType: DWORD;
    dwMTU: DWORD;
    dwSpeed: DWORD;
    dwPhysAddrLen: DWORD;
    bPhysAddr: array[1..MAXLEN_PHYSADDR] of byte;
    dwAdminStatus: DWORD;
    dwOperStatus: DWORD;
    dwLastChange: DWORD;
    dwInOctets: DWORD;
    dwInUcastPkts: DWORD;
    dwInNUCastPkts: DWORD;
    dwInDiscards: DWORD;
    dwInErrors: DWORD;
    dwInUnknownProtos: DWORD;
    dwOutOctets: DWORD;
    dwOutUCastPkts: DWORD;
    dwOutNUCastPkts: DWORD;
    dwOutDiscards: DWORD;
    dwOutErrors: DWORD;
    dwOutQLen: DWORD;
    dwDescrLen: DWORD;
    bDescr: array[1..MAXLEN_IFDESCR] of char; //byte;
  end;

  TMIBIfArray = array of TMIBIFRow;


 //
  PTMibIfTable = ^TMIBIfTable;
  TMibIfTable = packed record
    dwNumEntries: DWORD;
    Table: array[0..ANY_SIZE - 1] of TMibIfRow;
  end;

 //------ADAPTER INFO структура-----------------------------------------------

  TTIME_T = array[1..325] of byte; //

  PTIP_ADAPTER_INFO = ^TIP_ADAPTER_INFO;
  TIP_ADAPTER_INFO = packed record
    Next: PTIP_ADAPTER_INFO;
    ComboIndex: DWORD;
    AdapterName: array[1..MAX_ADAPTER_NAME_LENGTH + 4] of char;
    Description: array[1..MAX_ADAPTER_DESCRIPTION_LENGTH + 4] of char;
    AddressLength: UINT;
    Address: array[1..MAX_ADAPTER_ADDRESS_LENGTH] of byte;
    Index: DWORD;
    aType: UINT;
```

```
    DHCPEnabled: UINT;
    CurrentIPAddress: PTIP_ADDR_STRING;
    IPAddressList: TIP_ADDR_STRING;
    GatewayList: TIP_ADDR_STRING;
    DHCPServer: TIP_ADDR_STRING;
    HaveWINS: BOOL;
    PrimaryWINSServer: TIP_ADDR_STRING;
    SecondaryWINSServer: TIP_ADDR_STRING;
    LeaseObtained: TTIME_T; //??
    LeaseExpires: TTIME_T; //??
  end;

//----------------TCP структура-----------------------------------------------

  PTMibTCPRow = ^TMibTCPRow;
  TMibTCPRow = packed record
    dwState: DWORD;
    dwLocalAddr: DWORD;
    dwLocalPort: DWORD;
    dwRemoteAddr: DWORD;
    dwRemotePort: DWORD;
  end;
  //
  PTMibTCPTable = ^TMibTCPTable;
  TMibTCPTable = packed record
    dwNumEntries: DWORD;
    Table: array[0..0] of TMibTCPRow;
  end;
  //
  PTMibTCPStats = ^TMibTCPStats;
  TMibTCPStats = packed record
    dwRTOAlgorithm: DWORD;
    dwRTOMin: DWORD;
    dwRTOMax: DWORD;
    dwMaxConn: DWORD;
    dwActiveOpens: DWORD;
    dwPassiveOpens: DWORD;
    dwAttemptFails: DWORD;
    dwEstabResets: DWORD;
    dwCurrEstab: DWORD;
    dwInSegs: DWORD;
    dwOutSegs: DWORD;
    dwRetransSegs: DWORD;
    dwInErrs: DWORD;
    dwOutRsts: DWORD;
    dwNumConns: DWORD;
  end;

//---------UDP структура-----------------------------------------------------

  PTMibUDPRow = ^TMibUDPRow;
  TMibUDPRow = packed record
    dwLocalAddr: DWORD;
    dwLocalPort: DWORD;
  end;
  //
  PTMibUDPTable = ^TMIBUDPTable;
  TMIBUDPTable = packed record
    dwNumEntries: DWORD;
    UDPTable: array[0..ANY_SIZE - 1] of TMibUDPRow;
  end;
 //
  PTMibUdpStats = ^TMIBUdpStats;
  TMIBUdpStats = packed record
    dwInDatagrams: DWORD;
    dwNoPorts: DWORD;
    dwInErrors: DWORD;
    dwOutDatagrams: DWORD;
    dwNumAddrs: DWORD;
```

```
  end;

//-----------IP структура-------------------------------------------------------

 //
  PTMibIPNetRow = ^TMibIPNetRow;
  TMibIPNetRow = packed record
    dwIndex: DWord;
    dwPhysAddrLen: DWord;
    bPhysAddr: TMACAddress;
    dwAddr: DWord;
    dwType: DWord;
  end;
  //
  PTMibIPNetTable = ^TMibIPNetTable;
  TMibIPNetTable = packed record
    dwNumEntries: DWORD;
    Table: array[0..ANY_SIZE - 1] of TMibIPNetRow;
  end;
  //
  PTMibIPStats = ^TMibIPStats;
  TMibIPStats = packed record
    dwForwarding: DWORD;
    dwDefaultTTL: DWORD;
    dwInReceives: DWORD;
    dwInHdrErrors: DWORD;
    dwInAddrErrors: DWORD;
    dwForwDatagrams: DWORD;
    dwInUnknownProtos: DWORD;
    dwInDiscards: DWORD;
    dwInDelivers: DWORD;
    dwOutRequests: DWORD;
    dwRoutingDiscards: DWORD;
    dwOutDiscards: DWORD;
    dwOutNoRoutes: DWORD;
    dwReasmTimeOut: DWORD;
    dwReasmReqds: DWORD;
    dwReasmOKs: DWORD;
    dwReasmFails: DWORD;
    dwFragOKs: DWORD;
    dwFragFails: DWORD;
    dwFragCreates: DWORD;
    dwNumIf: DWORD;
    dwNumAddr: DWORD;
    dwNumRoutes: DWORD;
  end;
  //
  PTMibIPAddrRow = ^TMibIPAddrRow;
  TMibIPAddrRow = packed record
    dwAddr: DWORD;
    dwIndex: DWORD;
    dwMask: DWORD;
    dwBCastAddr: DWORD;
    dwReasmSize: DWORD;
    Unused1,
      Unused2: WORD;
  end;

  TMibIPAddrArray = array of TMIBIPAddrRow;


  //
  PTMibIPAddrTable = ^TMibIPAddrTable;
  TMibIPAddrTable = packed record
    dwNumEntries: DWORD;
    Table: array[0..ANY_SIZE - 1] of TMibIPAddrRow;
  end;

  //
```

```
  PTMibIPForwardRow = ^TMibIPForwardRow;
  TMibIPForwardRow = packed record
    dwForwardDest: DWORD;
    dwForwardMask: DWORD;
    dwForwardPolicy: DWORD;
    dwForwardNextHop: DWORD;
    dwForwardIFIndex: DWORD;
    dwForwardType: DWORD;
    dwForwardProto: DWORD;
    dwForwardAge: DWORD;
    dwForwardNextHopAS: DWORD;
    dwForwardMetric1: DWORD;
    dwForwardMetric2: DWORD;
    dwForwardMetric3: DWORD;
    dwForwardMetric4: DWORD;
    dwForwardMetric5: DWORD;
  end;
  //
  PTMibIPForwardTable = ^TMibIPForwardTable;
  TMibIPForwardTable = packed record
    dwNumEntries: DWORD;
    Table: array[0..ANY_SIZE - 1] of TMibIPForwardRow;
  end;

//--------ICMP-структура-------------------------------------------------------

  PTMibICMPStats = ^TMibICMPStats;
  TMibICMPStats = packed record
    dwMsgs: DWORD;
    dwErrors: DWORD;
    dwDestUnreachs: DWORD;
    dwTimeEcxcds: DWORD;
    dwParmProbs: DWORD;
    dwSrcQuenchs: DWORD;
    dwRedirects: DWORD;
    dwEchos: DWORD;
    dwEchoReps: DWORD;
    dwTimeStamps: DWORD;
    dwTimeStampReps: DWORD;
    dwAddrMasks: DWORD;
    dwAddrReps: DWORD;
  end;

  PTMibICMPInfo = ^TMibICMPInfo;
  TMibICMPInfo = packed record
    InStats: TMibICMPStats;
    OutStats: TMibICMPStats;
  end;



  //------------------імпортовано з IPHLPAPI.DLL---------------------------------
  -

function GetAdaptersInfo( pAdapterInfo: PTIP_ADAPTER_INFO;
  pOutBufLen: PULONG ): DWORD;
stdcall; external 'IPHLPAPI.DLL';

function GetNetworkParams( FixedInfo: PTFixedInfo; pOutPutLen: PULONG ): DWORD;
stdcall; external 'IPHLPAPI.DLL';

function GetTcpTable( pTCPTable: PTMibTCPTable; pDWSize: PDWORD;
  bOrder: BOOL ): DWORD;
stdcall; external 'IPHLPAPI.DLL';

function GetTcpStatistics( pStats: PTMibTCPStats ): DWORD;
stdcall; external 'IPHLPAPI.DLL';
```

```
function GetUdpTable( pUdpTable: PTMibUDPTable; pDWSize: PDWORD;
  bOrder: BOOL ): DWORD;
stdcall; external 'IPHLPAPI.DLL';

function GetUdpStatistics( pStats: PTMibUdpStats ): DWORD;
stdcall; external 'IPHLPAPI.DLL';

function GetIpStatistics( pStats: PTMibIPStats ): DWORD;
stdcall; external 'IPHLPAPI.DLL';

function GetIpNetTable( pIpNetTable: PTMibIPNetTable;
  pdwSize: PULONG;
  bOrder: BOOL ): DWORD;
stdcall; external 'IPHLPAPI.DLL';

function GetIpAddrTable( pIpAddrTable: PTMibIPAddrTable;
  pdwSize: PULONG;
  bOrder: BOOL ): DWORD;
stdcall; external 'IPHLPAPI.DLL';

function GetIpForwardTable( pIPForwardTable: PTMibIPForwardTable;
  pdwSize: PULONG;
  bOrder: BOOL ): DWORD;
stdCall; external 'IPHLPAPI.DLL';

function GetIcmpStatistics( pStats: PTMibICMPInfo ): DWORD;
stdCall; external 'IPHLPAPI.DLL';
function GetRTTAndHopCount( DestIPAddress: DWORD; HopCount: PULONG;
  MaxHops: ULONG; RTT: PULONG ): BOOL;
stdCall; external 'IPHLPAPI.DLL';
function GetIfTable( pIfTable: PTMibIfTable; pdwSize: PULONG;
  bOrder: boolean ): DWORD;
stdCall; external 'IPHLPAPI.DLL';
function GetIfEntry( pIfRow: PTMibIfRow ): DWORD;
stdCall; external 'IPHLPAPI.DLL';

implementation
end.
```

unit TrafficUnit – визначення параметрів трафіку;

```
interface

uses SysUtils, Windows, IPHelper, IPHLPAPI;


type
  TTraffic = Class;

  TNewInstanceEvent = procedure(Sender :TTraffic) of object;
  TFreezeEvent = procedure(Sender :TTraffic) of object;

  TTraffic = Class
  private
    FIP: string;
    FMac: string;
    FInPerSec: Dword;
    FInTotal: Dword;
    FPeakInPerSec: Dword;
    FInterfaceIndex: DWord;
    FActiveCountIn: Dword;
    FSecondsActive: Cardinal;
    FPrevCountIn: DWord;
    FDescription: string;
    FOutTotal: Dword;
    FPeakOutPerSec: Dword;
    FOutPerSec: Dword;
    FPrevCountOut: DWord;
    FActiveCountOut: Dword;
    FAverageInPerSec: Dword;
    FAverageOutPerSec: Dword;
    FStartedAt: TDateTime;
    FRunning: boolean;
    FOnFreeze: TFreezeEvent;
    FOnUnFreeze: TFreezeEvent;
    FConnected: boolean;
    FFound: boolean;
    FSpeed: DWord;

    function GetIPFromIFIndex(InterfaceIndex: Cardinal): string;
  public
    property Found : boolean read FFound write FFound;
    property Connected : boolean read FConnected;
    property Running : boolean read FRunning;
    property InterfaceIndex : DWord read FInterfaceIndex;
    property IP : string read FIP;
    property Mac : string read FMac;
    property Description : string read FDescription;
    property StartedAt : TDateTime read FStartedAt;
    property SecondsActive : Cardinal read FSecondsActive;
    property Speed : DWord read FSpeed;
    property ActiveCountIn : Dword read FActiveCountIn; { }
    property PrevCountIn : DWord read FPrevCountIn; { }
    property InPerSec : Dword read FInPerSec; { байт підраховано в останній
період }
    property AverageInPerSec : Dword read FAverageInPerSec; { У середньому }
    property InTotal : Dword read FInTotal; { загальна кількість байтів }
    property PeakInPerSec : Dword read FPeakInPerSec; { максимальне число
байтів}

    property ActiveCountOut : Dword read FActiveCountOut; { підраховує число
байтів при передачі }
    property PrevCountOut : DWord read FPrevCountOut; { попереднє підрахування
байтів }
    property OutPerSec : Dword read FOutPerSec; { Кількість байтів у останій
період }
    property AverageOutPerSec : Dword read FAverageOutPerSec; { }
```

```
    property OutTotal : Dword read FOutTotal; { Загальна кількість байтів
передано }
    property PeakOutPerSec : Dword read FPeakOutPerSec; { Максимальна кількість
байтів передано }

    procedure NewCycle(const InOctets, OutOctets, TrafficSpeed : Dword);
    procedure Reset;
    procedure Freeze;
    procedure UnFreeze;
    procedure MarkDisconnected;
    function GetStatus : string;
    function FriendlyRunningTime:string;
    constructor Create(const AMibIfRow : TMibIfRow; OnNewInstance :
TNewInstanceEvent);
  published
    property OnFreeze :TFreezeEvent read FOnFreeze write FOnFreeze;
    property OnUnFreeze :TFreezeEvent read FOnUnFreeze write FOnUnFreeze;
  end;

  function BytesToFriendlyString(Value : DWord) : string;
  function BitsToFriendlyString(Value : DWord) : string;

implementation

function BytesToFriendlyString(Value : DWord) : string;
const
 OneKB=1024;
 OneMB=OneKB*1024;
 OneGB=OneMB*1024;
begin
 if Value<OneKB
 then Result:=FormatFloat('#,##0.00 B',Value)
 else
  if Value<OneMB
  then Result:=FormatFloat('#,##0.00 KB', Value/OneKB)
  else
   if Value<OneGB
   then Result:=FormatFloat('#,##0.00 MB', Value/OneMB)
end;

function BitsToFriendlyString(Value : DWord) : string;
const
 OneKB=1000;
 OneMB=OneKB*1000;
 OneGB=OneMB*1000;
begin
 if Value<OneKB
 then Result:=FormatFloat('#,##0.00 bps',Value)
 else
  if Value<OneMB
  then Result:=FormatFloat('#,##0.00 Kbps', Value/OneKB)
  else
   if Value<OneGB
   then Result:=FormatFloat('#,##0.00 Mbps', Value/OneMB)
end;

constructor TTraffic.Create(const AMibIfRow: TMibIfRow; OnNewInstance :
TNewInstanceEvent);
var
 Descr: string;
begin
 inherited Create;

 FRunning:=true;
 FConnected:=true;

 self.FInterfaceIndex:=AMibIfRow.dwIndex;
 self.FIP:=GetIPFromIFIndex(self.InterfaceIndex);
```

```
 self.FMac:=MacAddr2Str(TMacAddress(AMibIfRow.bPhysAddr),
AMibIfRow.dwPhysAddrLen);

 SetLength(Descr, Pred(AMibIfRow.dwDescrLen));
 Move(AMibIfRow.bDescr, Descr[1], pred(AMibIfRow.dwDescrLen));
 self.FDescription:=Trim(Descr);

 self.FPrevCountIn:=AMibIfRow.dwInOctets;
 self.FPrevCountOut:=AMibIfRow.dwOutOctets;

 self.FStartedAt:=Now;
 self.FSpeed:=AMibIfRow.dwSpeed;

 FActiveCountIn:=0;
 FActiveCountOut:=0;
 FInTotal:=0;
 FOutTotal:=0;
 FInPerSec:=0;
 FOutPerSec:=0;
 FPeakInPerSec:=0;
 FPeakOutPerSec:=0;

  if Assigned(OnNewInstance)
 then OnNewInstance(self);
end;

procedure TTraffic.NewCycle(const InOctets, OutOctets, TrafficSpeed: Dword);
begin
 inc(self.FSecondsActive);
 if not Running
 then Exit;
 FSpeed:=TrafficSpeed;
 // прийнято
 self.FInPerSec:=InOctets-self.PrevCountIn;
 Inc(self.FInTotal, self.InPerSec);
 if InPerSec>0
 then Inc(FActiveCountIn);
 if InPerSec>PeakInPerSec
 then FPeakInPerSec:=InPerSec;
 try
  if ActiveCountIn<>0
  then self.FAverageInPerSec:=InTotal div ActiveCountIn
 except
  self.FAverageInPerSec:=0;
 end;
 FPrevCountIn:=InOctets;
 // передано
 self.FOutPerSec:=OutOctets-self.PrevCountOut;
 Inc(self.FOutTotal,self.OutPerSec);
 if OutPerSec>0
 then Inc(FActiveCountOut);
 if OutPerSec>PeakOutPerSec
 then FPeakOutPerSec:=OutPerSec;
 try
  if ActiveCountIn<>0
  then self.FAverageOutPerSec:=OutTotal div ActiveCountOut
 except
  self.FAverageOutPerSec:=0;
 end;
 FPrevCountOut:=OutOctets;
end;

function TTraffic.GetIPFromIFIndex(InterfaceIndex: Cardinal): string;
var
 i: integer;
 IPArr: TMIBIPAddrArray;
begin
 Result:='Не знайдено!'; //...
 Get_IPAddrTableMIB(IpArr); // беремо таблицю IP-адрес
```

```
 if Length(IPArr)>0
 then
  for i:=low(IPArr) to High(IPArr) do  // проглядаємо індекси у таблиці...
   if IPArr[i].dwIndex=InterfaceIndex
   then
    begin
     Result:=IPAddr2Str(IParr[i].dwAddr);
     Break;
    end;
end;

procedure TTraffic.Reset;
begin
 self.FPrevCountIn:=InPerSec;
 self.FPrevCountOut:=OutPerSec;

 self.FStartedAt:=Now;
 FSecondsActive:=0;

 FActiveCountIn:=0;
 FActiveCountOut:=0;
 FInTotal:=0;
 FOutTotal:=0;
 FInPerSec:=0;
 FOutPerSec:=0;
 FPeakInPerSec:=0;
 FPeakOutPerSec:=0;
end;

procedure TTraffic.Freeze;
begin
 FRunning:=false;
 if Assigned(FOnFreeze)
 then OnFreeze(Self);
end;

procedure TTraffic.UnFreeze;
begin
 FRunning:=true;
 if Assigned(FOnUnFreeze)
 then OnUnFreeze(Self);
end;

procedure TTraffic.MarkDisconnected;
begin
 self.FConnected:=false;
 self.FRunning:=false;
end;
function TTraffic.GetStatus: string;
begin
 if self.Connected
 then Result:='Підключено'
 else Result:='Не підключено';
 if self.Running
 then Result:=Result+', Запущено'
 else Result:=Result+', Не запущено';
end;
function TTraffic.FriendlyRunningTime: string;
var
 H,M,S: string;
 ZH,ZM,ZS: integer;
begin
 ZH:=SecondsActive div 3600;
 ZM:=Integer(SecondsActive) div (60-ZH*60);
 ZS:=Integer(SecondsActive)-(ZH*3600+ZM*60);
 H:=Format('%.2d',[ZH]);
 M:=Format('%.2d',[ZM]);
 S:=Format('%.2d',[ZS]);
 Result:=H+':'+M+':'+S;
```

```
end;
end.
```

unit MainFormUnit – Запуск основної форми програми;

```
interface

uses
  Windows, Graphics, ExtCtrls, Controls, StdCtrls, Buttons, Tabs,
  ComCtrls, Classes, SysUtils, Forms, dialogs,
  TrafficUnit, IPHelper, IPHLPAPI, ShellAPI;

type
  TMainForm = class(TForm)
    pnlMain: TPanel;
    pnlBottom: TPanel;
    pc: TPageControl;
    tsAbout: TTabSheet;
    tsTraffic: TTabSheet;
    ExitButton: TButton;
    TrafficTabs: TTabSet;
    GroupBox: TGroupBox;
    ledAdapterDescription: TLabeledEdit;
    UnFreezeButton: TBitBtn;
    FreezeButton: TBitBtn;
    ClearCountersButton: TBitBtn;
    ledMACAddress: TLabeledEdit;
    gbIN: TGroupBox;
    ledOctInSec: TLabeledEdit;
    ledAvgINSec: TLabeledEdit;
    ledPeakINSec: TLabeledEdit;
    ledTotalIN: TLabeledEdit;
    gbOUT: TGroupBox;
    ledOctOUTSec: TLabeledEdit;
    ledAvgOUTSec: TLabeledEdit;
    ledPeakOUTSec: TLabeledEdit;
    ledTotalOUT: TLabeledEdit;
    Timer: TTimer;
    gbTime: TGroupBox;
    ledStartedAt: TLabeledEdit;
    ledActiveFor: TLabeledEdit;
    RemoveInactiveButton: TBitBtn;
    StatusText: TStaticText;
    cbOnTop: TCheckBox;
    Panel3: TPanel;
    ProductName: TLabel;
    lblURL: TLabel;
    Label3: TLabel;
    ProgramIcon: TImage;
    StaticText1: TStaticText;
    ledSpeed: TLabeledEdit;
    procedure TimerTimer(Sender: TObject);
    procedure ClearCountersButtonClick(Sender: TObject);
    procedure cbOnTopClick(Sender: TObject);
    procedure FormDestroy(Sender: TObject);
    procedure TrafficTabsChange(Sender: TObject; NewTab: Integer;
      var AllowChange: Boolean);
    procedure ExitButtonClick(Sender: TObject);
    procedure FormCreate(Sender: TObject);
    procedure FreezeButtonClick(Sender: TObject);
    procedure UnFreezeButtonClick(Sender: TObject);
    procedure RemoveInactiveButtonClick(Sender: TObject);
    procedure lblURLClick(Sender: TObject);
    procedure StaticText1Click(Sender: TObject);
    procedure pcChange(Sender: TObject);
    procedure ledAdapterDescriptionChange(Sender: TObject);
  private
    procedure HandleNewAdapter(ATraffic : TTraffic);
    procedure HandleFreeze(ATraffic : TTraffic);
    procedure HandleUnFreeze(ATraffic : TTraffic);
    function LocateTraffic(AdapterIndex : DWord) : TTraffic;
    procedure ProcessMIBData;
```

```
    procedure ClearDisplay;
    procedure RefreshDisplay;
  public
    { }
  end;

var
  MainForm: TMainForm;
  ActiveTraffic : TTraffic;

implementation
{$R *.dfm}

procedure TMainForm.ClearDisplay;
var
 j:integer;
begin
 TrafficTabs.Tabs.Clear;
 StatusText.Caption:='';
 for j:=0 to GroupBox.ControlCount-1 do
  begin
   if GroupBox.Controls[j] is TCustomEdit
   then TCustomEdit(GroupBox.Controls[j]).Text:='';
  end;
end;

procedure TMainForm.TimerTimer(Sender: TObject);
begin
 Timer.Enabled:=false;
 ProcessMIBData;
 Timer.Enabled:=true;
end;

procedure TMainForm.ClearCountersButtonClick(Sender: TObject);
begin
 ActiveTraffic.Reset;
 RefreshDisplay;
end;

procedure TMainForm.cbOnTopClick(Sender: TObject);
begin
 if cbOnTop.Checked=true
 then FormStyle:=fsSTAYONTOP
 else FormStyle:=fsNORMAL;
end;
procedure TMainForm.FormDestroy(Sender: TObject);
var
 i: integer;
begin
 Timer.OnTimer:=nil;
 ActiveTraffic:=nil;
 for i:=0 to -1+TrafficTabs.Tabs.Count do
  TrafficTabs.Tabs.Objects[i].Free;
end;

procedure TMainForm.TrafficTabsChange(Sender: TObject; NewTab: Integer; var
AllowChange: Boolean);
begin
 if NewTab=-1
 then ActiveTraffic:=nil
 else ActiveTraffic:=TTraffic(TrafficTabs.Tabs.Objects[NewTab]);
 RefreshDisplay;
end;

procedure TMainForm.ExitButtonClick(Sender: TObject);
begin
 Close;
end;
```

```
procedure TMainForm.FormCreate(Sender: TObject);
begin
  Timer.Interval:=1000; // усі розрахунки за 1 сек.
 //
 ClearDisplay;
 ActiveTraffic:=nil;
 pcChange(Sender);
 Timer.Enabled:=True;
end;

procedure TMainForm.RefreshDisplay;
begin
 if not Assigned(ActiveTraffic)
 then
  begin
   ClearDisplay;
   Exit;
  end;
 with ActiveTraffic do
  begin
   FreezeButton.Visible:=Connected;
   UnFreezeButton.Visible:=Connected;
   ClearCountersButton.Visible:=Connected;
   RemoveInactiveButton.Visible:=not Connected;

   FreezeButton.Enabled:=Running;
   UnFreezeButton.Enabled:=not Running;

   ledAdapterDescription.Text:=Description;
   ledMACAddress.Text:=MAC;

   ledSpeed.Text:=BitsToFriendlyString(Speed);

   ledOctInSec.Text:=BytesToFriendlyString(InPerSec);
   ledPeakInSec.Text:=BytesToFriendlyString(PeakInPerSec);
   ledAvgINSec.Text:=BytesToFriendlyString(AverageInPerSec);
   ledTotalIN.Text:=BytesToFriendlyString(InTotal);

   ledOctOUTSec.Text:=BytesToFriendlyString(OutPerSec);
   ledPeakOUTSec.Text:=BytesToFriendlyString(PeakOutPerSec);
   ledAvgOUTSec.Text:=BytesToFriendlyString(AverageOutPerSec);
   ledTotalOUT.Text:=BytesToFriendlyString(OutTotal);

   self.ledStartedAt.Text:=DateTimeToStr(StartedAt);
   self.ledActiveFor.Text:=FriendlyRunningTime;

   StatusText.Caption:=GetStatus;
  end;
end;

procedure TMainForm.ProcessMIBData;
var
 MibArr: IpHlpAPI.TMIBIfArray;
 i: integer;
 ATraffic: TTraffic;
begin
 Get_IfTableMIB(MibArr);  // Беремо поточні MIB дані
 //Мітку не знайдено, або не підключено
 for i:= 0 to -1 + TrafficTabs.Tabs.Count do
  begin
   ATraffic:=TTraffic(TrafficTabs.Tabs.Objects[i]);
   if ATraffic.Connected
   then ATraffic.Found:=false;
  end;
  //процесс
 if Length(MibArr)>0
 then
  begin
   for i:=Low(MIBArr) to High(MIBArr) do
```

```
    begin
     ATraffic:=LocateTraffic(MIBArr[i].dwIndex);
     if Assigned(ATraffic)
     then
      begin
       //заново підключаємось
       ATraffic.NewCycle(MIBArr[i].dwInOctets, MIBArr[i].dwOutOctets,
MIBArr[i].dwSpeed);
      end
     else
      begin
       //новий запис у таблицю!
       ATraffic:=TTraffic.Create(MIBArr[i], HandleNewAdapter);
       ATraffic.Found:=true;
       ATraffic.OnFreeze:=HandleFreeze;
       ATraffic.OnUnFreeze:=HandleUnFreeze;
      end;
    end;
  end;
 //Мітка не знайдена
 for i:=0 to -1+TrafficTabs.Tabs.Count do
  if not TTraffic(TrafficTabs.Tabs.Objects[i]).Found
  then TTraffic(TrafficTabs.Tabs.Objects[i]).MarkDisconnected;
 RefreshDisplay;
end;

function TMainForm.LocateTraffic(AdapterIndex : DWord): TTraffic;
var
 j: cardinal;
 ATraffic: TTraffic;
begin
 Result:=nil;
 if TrafficTabs.Tabs.Count=0
 then Exit;

 for j:= 0 to -1+TrafficTabs.Tabs.Count do
  begin
   ATraffic:=TTraffic(TrafficTabs.Tabs.Objects[j]);
   if ATraffic.InterfaceIndex=AdapterIndex
   then
    begin
     Result:=ATraffic;
     Result.Found:=true;
     Break;
    end;
  end;
end;

procedure TMainForm.HandleNewAdapter(ATraffic: TTraffic);
begin
 //додаємо адаптер
 TrafficTabs.Tabs.AddObject(ATraffic.IP, ATraffic);
 // вибираємо
 TrafficTabs.TabIndex:=-1+TrafficTabs.Tabs.Count;
end;

procedure TMainForm.FreezeButtonClick(Sender: TObject);
begin
 ActiveTraffic.Freeze;
end;

procedure TMainForm.UnFreezeButtonClick(Sender: TObject);
begin
 ActiveTraffic.UnFreeze;
end;

procedure TMainForm.HandleFreeze(ATraffic: TTraffic);
begin
 self.FreezeButton.Enabled:=ATraffic.Running;
```

```
 self.UnFreezeButton.Enabled:=not ATraffic.Running;
end;

procedure TMainForm.HandleUnFreeze(ATraffic: TTraffic);
begin
 self.FreezeButton.Enabled:=ATraffic.Running;
 self.UnFreezeButton.Enabled:=not ATraffic.Running;
end;

procedure TMainForm.RemoveInactiveButtonClick(Sender: TObject);
begin
 if not ActiveTraffic.Connected
 then //точна перевірка
  begin
   ActiveTraffic.Free;
   ActiveTraffic:=nil;
   TrafficTabs.Tabs.Delete(TrafficTabs.TabIndex);
   TrafficTabs.SelectNext(False);
  end;
 RefreshDisplay;
end;

procedure TMainForm.lblURLClick(Sender: TObject);
begin
 ShellExecute(Handle, 'open','',nil,nil,SW_SHOWNORMAL);
end;

procedure TMainForm.StaticText1Click(Sender: TObject);
begin
 ShellExecute(Handle, 'open','',nil,nil,SW_SHOWNORMAL);
end;

procedure TMainForm.pcChange(Sender: TObject);
begin
 pnlBottom.Visible:=pc.ActivePage=tsTraffic;
end;

procedure TMainForm.ledAdapterDescriptionChange(Sender: TObject);
begin
  ledAdapterDescription.Hint:=ledAdapterDescription.Text;

ledAdapterDescription.ShowHint:=Canvas.TextWidth(ledAdapterDescription.Text)>led
AdapterDescription.ClientWidth;
end;

end.
```

unit about – підпрограмма про розробників та місце розроблення програми;

```
interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, ExtCtrls, StdCtrls;

type
  TForm2 = class(TForm)
    Image1: TImage;
    Label1: TLabel;
    Label2: TLabel;
    Label3: TLabel;
    Label4: TLabel;
    Label5: TLabel;
    Label6: TLabel;
    Button1: TButton;
    Label7: TLabel;
    procedure Button1Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form2: TForm2;

implementation

{$R *.dfm}

procedure TForm2.Button1Click(Sender: TObject);
begin
Form2.Close;
end;

end.
```