МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Центральноукраїнський національний технічний університет

Кафедра кібербезпеки та програмного забезпечення

На правах рукопису

Лукашов Анатолій Миколайович

Програмне забезпечення системи кібербезпеки NPMD та APM

Спеціальність: 125 «Кібербезпека»

Освітній ступінь: бакалавр

Науковий керівник:

Коваленко Олександр Володимирович_____

(підпис) (дата)

доктор технічних наук, доцент

ДОПУЩЕНО ДО ЗАХИСТУ

Завідувач кафедри

_____ О.А. Смірнов

(підпис) ПІБ

«____»_____2021 р.

Кропивницький – 2021

Міністерство освіти і науки України
Центральноукраїнський національний технічний університет

Центр *Заочної та дистанційної освіти*
Кафедра *Кібербезпеки та програмного забезпечення*
Освітній ступінь *бакалавр*
Спеціальність *125 Кібербезпека*

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

Лукашову Анатолію Миколайовичу
(прізвище, ім'я, по батькові)

1. Тема роботи *Програмне забезпечення системи кібербезпеки NPMD та APM*

керівник роботи *Коваленко Олександр Володимирович, докт. техн. наук, доцент*
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу № 135-02 від 24.12.2020 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 кібербезпеки NPMD та APM*

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

Лукашов А.М. Програмне забезпечення системи кібербезпеки NPMD та АРМ. 125 Кібербезпека. Центральноукраїнський національний технічний університет. Кропивницький. 2021.

В даній кваліфікаційній бакалаврській розроблено програмне забезпечення, яке призначено для системи кібербезпеки NPMD та АРМ.

Метою розробки є програмне забезпечення системи кібербезпеки NPMD та АРМ.

Результат роботи – програмна реалізація системи кібербезпеки NPMD та АРМ.

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 RAD Studio Delphi 10.

Ключові слова: кібербезпека, NPMD, АРМ

ABSTRACT

Lukashov A.M. NPMD and APM cybersecurity software. 125 Cybersecurity. Central Ukrainian National Technical University. Kropyvnytskyi. 2021

This bachelor's degree software has been developed for the NPMD and APM cybersecurity system.

The purpose of the development is the software of the cybersecurity system NPMD and APM.

The result is a software implementation of the cybersecurity system NPMD and APM.

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 is developed in the environment of RAD Studio Delphi 10.

Keywords: cybersecurity, NPMD, APM

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ

EOM	–	електронно-обчислювальна машина
ATM	–	Asynchronous Transfer Mode – асинхронний спосіб передачі даних
BER	–	Bit Error Rate – імовірність ушкодження одного біта
CBWFQ	–	class based weighted fair queueing
DiffServ	–	Differentiated Services – механізм який залежно від вимог до якості обслуговування записує в IP заголовки пакетів спеціальні мітки
DSCP	–	DiffServ CoSde Point
ICMP	–	InterNet CoSntrol Message Protocol – міжмережний протокол управляючих повідомлень
ISP	–	InterNet Service Provider – провайдер
LLQ	–	low latency queueing
MPLS-TE	–	Multiprotocol Label Switching Traffic Engineering
PQ	–	priority queueing
RIO	–	RED with Input Output
RTT	–	Round Trip Time – час між відправкою запиту та отриманням відповіді
STM	–	Synchronous Transfer Mode – синхронний спосіб передачі даних
QoS	–	Quality of Service – якість обслуговування
WFQ	–	Weighted Fair Queuing – механізм планування пакетних потоків даних
WRED	–	Weighted random early detection – взвішане значення довжини черги, у якості фактора, визначаючого ймовірність відкидання пакета

ВСТУП

Актуальність теми. Жодна компаній не може існувати без IT-систем – це може бути проста пошта, 1С, ERP система і т.д. Якщо ми починаємо говорити про продуктивність IT-систем для внутрішніх або зовнішніх замовників, то від швидкості їх роботи залежить ефективність співробітників і продуктивність праці, про яку зараз популярно говорити. Протягом останніх 2-3 років керівників стало цікавити чому в IT вкладається багато грошей, а користувачі продовжують скаржитися на низьку продуктивність або якість сервісу. Вирішувати дану проблему можна по різному, наприклад, брати простий сніффер (аналізатор протоколів) начебто Wireshark або задуматися про повноцінні системи моніторингу від різних виробників.

Моніторинг продуктивності мережі (Network Performance Monitoring, NPMD)

Моніторинг продуктивності мережі (Network Performance Monitoring) – це програмні або апаратні компоненти, що підключаються до мережі, з метою контролю над продуктивністю, доступністю мережних компонентів і переданим трафіком. Ці застосунки накопичують статистику в часі й включають набір засобів оцінки базової продуктивності мережі (baselining), контролю над відхиленням від порогів, аналізу мережного трафіку, аналізу трендів, систему звітів, а в деяких випадках і біллінг.

Моніторинг продуктивності застосунків (Application Performance Monitoring, APM)

Моніторинг продуктивності застосунків (Application Performance Monitoring) – це програмні або апаратні компоненти, що підключаються до мережі з метою оцінки її роботи з п'яти функціональним вимірам:

– моніторинг роботи кінцевих користувачів або досвіду користувача (EUM);

– розкриття, моделювання й відображення архітектури застосунків під час роботи;

– профілювання користувацьких транзакцій;

– поглиблений компонентний моніторинг у контексті застосунку;

– аналітика.

Application Aware Network Performance Monitoring (AANPM)

У реальному світі градації складніше й завдання клієнтів більш різні, тому застосунки не завжди чітко лягають у ті границі, що їм відводять аналітики. В останній рік помітне змішання двох класів в один за назвою Application Aware Network Performance Monitoring (AANPM), тому що в подібних системах важливо для ухвалення рішення кореляція даних, а проблеми продуктивності застосунків на практиці часто сусідять із проблемами продуктивності мережі.

Мета й завдання дослідження. Метою роботи є програмне забезпечення системи кібербезпеки NPMD та APM.

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

– Огляд існуючих систем кібербезпеки NPMD та APM.

– Дослідження системи кібербезпеки NPMD та APM.

– Програмна реалізація системи кібербезпеки NPMD та APM.

Практична цінність отриманих результатів полягає в тому, що розроблені алгоритми дозволяють успішно вирішувати задачі NPMD та APM.

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи кібербезпеки NPMD та APM, є актуальною задачею, яка потребує вирішення у даній кваліфікаційній бакалаврській роботі.

1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ

1.1 Призначення системи

При виникненні проблем у роботі застосунків і сервісів дуже складно визначити дійсну причину гальмування. І найпоширеніша відповідь – це мережа працює повільно, канали вузькі, оператори зв'язку не забезпечують SLA. Адже нікому не цікаво переглядати тисячі пакетів з аналізатора протоколів (сніффера). Але дуже часто проблема виявляється саме в роботі самого застосунку або сервера, на якому він розгорнутий. Чому аналіз і в иявлення проблем із продуктивністю серверів досить складне завдання?

По-перше, відсутність фахівців, які розбираються в пакетах і протоколах і можуть набудовувати фільтри для пошуку збійних пакетів, що викликають великий час відгуку. Навіть, якщо знайшли сесію з більшим часом відгуку, з'ясувати реальну проблему, чим було це викликане також не тривіальне завдання.

Ідемо до користувача й починаємо захват трафіку на його стороні. Часто проблема вже пройшла, і чекати на неї виникнення знову доводиться довго. Бідний користувач робить ті ж самі дії, але система працює стабільно. Негатив з боку IT-фахівця й утішні слова в душі убік користувача забезпечені. Вихід можна поставити захват трафіку на стороні сервера, запустивши dumpcap на тривалий час зі збереженням трафіку на USB диск. Але давайте спробуємо подивитися файл із захопленим трафіком.

1.2 Область застосування

Шукаємо у файлі DNS запит до сервера, на якому працює додаток. Оцінюємо час відгуку DNS сервера й переконуємося, що воно прийнятне для вашої мережі (звичайний час відгуку не повинний бути більш 150 мс). Якщо користувач часто звертається до цього сервера, то шукаємо пакет запитом на встановлення TCP сесії (TCP SYN).

Коли ми аналізуємо час відгуку застосунків, будьте впевнені, що аналізатор налаштований коректно й використовує дельту в часі між пакетами.

Для аналізу швидкості встановлення сесії із сервером необхідно скористатися фільтром TCP Stream для вибірки пакетів, які ставляться до даної TCP сесії (правий клич миші на будь-якому пакеті в TCP сесії й вибираємо TCP Stream фільтр). Основна мета наших дій це зрівняти час передачі по мережі (Network roundtrip time) згодом відгуку застосунку (server response time).

Як тільки фільтр налаштований, і ми бачимо пакети, що ставляться до сесії, оцініте часову дельту між пакетом користувача з TCP SYN і пакетом з TCP SYN-ACK відправленим сервером користувачеві. Цей час можна використовувати як відправну крапку для пошуку джерела проблеми. Наприклад час відображається в пакеті номер 7 і становить 134 мс.

Як тільки користувач установив TCP з'єднання, він запросить необхідні дані. У прикладі вище клієнт відправив запит до Web серверу HTTP GET. Звертаємося до колонки дельта й бачимо, що сервер відповів через 125 мс. з TCP ACK, що підтверджує одержання сервером запиту, але не сама відповідь. І потім сервер витратив 4,85 секунд на відправлення пакета з необхідними даними й потім швидко передав швидкості, що залишилися пакети на, каналу (мережа літає). Порівнюючи час 4.85 секунди із часом установлення з'єднання – 134 мс. ми явно можемо зробити вивід – що час відгуку сервера дуже велике.

Ґрунтуючись на даній інформації дуже просто зрозуміти, що робити далі. Якщо час відгуку сервера суттєво вище часу встановлення з'єднання (connection

setup time) і немає повторних передач (TCP retransmissions), то проблема на стороні сервера. Мережа в даному прикладі ні при чому.

Якщо затримок у даній транзакції не виявлене, то потрібно переміщатися до іншого запиту, уважно відслідковуючи кількість часу, яка була витрачене сервером на відповідь користувача. Поступово одержавши досвід захвата трафіку на стороні клієнта, можна переходити до аналізу трафіку на стороні сервера, тому що цей аналіз дозволить зрозуміти, чим був зайнятий сервер, відповідаючи на запит нашого користувача протягом 4,38 секунд.

Шукати проблеми із продуктивністю мережі або сервера або застосунку можна й за допомогою Wireshark, але для вирішення даного завдання буде потрібно вправність, підключення в декількох місцях у мережі й бажаний одночасний захват трафіку з наступною синхронізацією файлів за часом. А якщо корпоративні застосунки багаторівневі й серверів багато, те автоматизовані комплекси моніторингу або комерційні аналізатори протоколів суттєво заощадять час.

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи кібербезпеки NPMD та APM, 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

Огляд застосунків AppNeta: AppView PathView, FlowView

Компанія заснована AppNeta в 2011 році на основі придбаних розробок і клієнтської бази компанії Apparent Networks. Основний фокус виробника – застосунки для моніторингу продуктивності й доступності застосунків. Компанія має кілька застосунків, які відповідають за продуктивність різних сегментів IT-інфраструктури. Обсяг продажів компанії по оцінках аналітиків становить від 5 до 10 мільйонів доларів. Клієнтська база – близько 200 корпоративних замовників.

Застосунки компанії працюють на основі апаратних і віртуальних програмних сенсорів. У продуктовий портфель AppNeta входять: AppView, PathView, FlowView, TraceView. Застосунку не підтримують традиційні технології SNMP/CLI/NetFlow/WMI і т.д. для моніторингу стану елементів IT-інфраструктури, тому стане прекрасним доповненням до існуючих у корпоративних замовників класичних систем по керуванню мережею (NMS).

Застосунок AppView

Застосунок AppView дозволяє контролювати доступність і продуктивність Saas застосунків. Основний метод роботи – це генерація синтетичних запитів для оцінки продуктивності Web застосунків, таких як: Salesforce, Servicenow, Microsoft Dynamics і Office 365, Google Docs, Netsuite і т.д. Застосунку досить простої в настроюванні, дозволяє виконувати не тільки синтетичні запити й відкриття Web ресурсів, але й виконувати скрипти й до 20 різних команд до ресурсу. Результати зберігаються протягом 30 днів з рівнем деталізації до 5 хвилин, рік зберігається інформація про тренди із продуктивністю ресурсів.

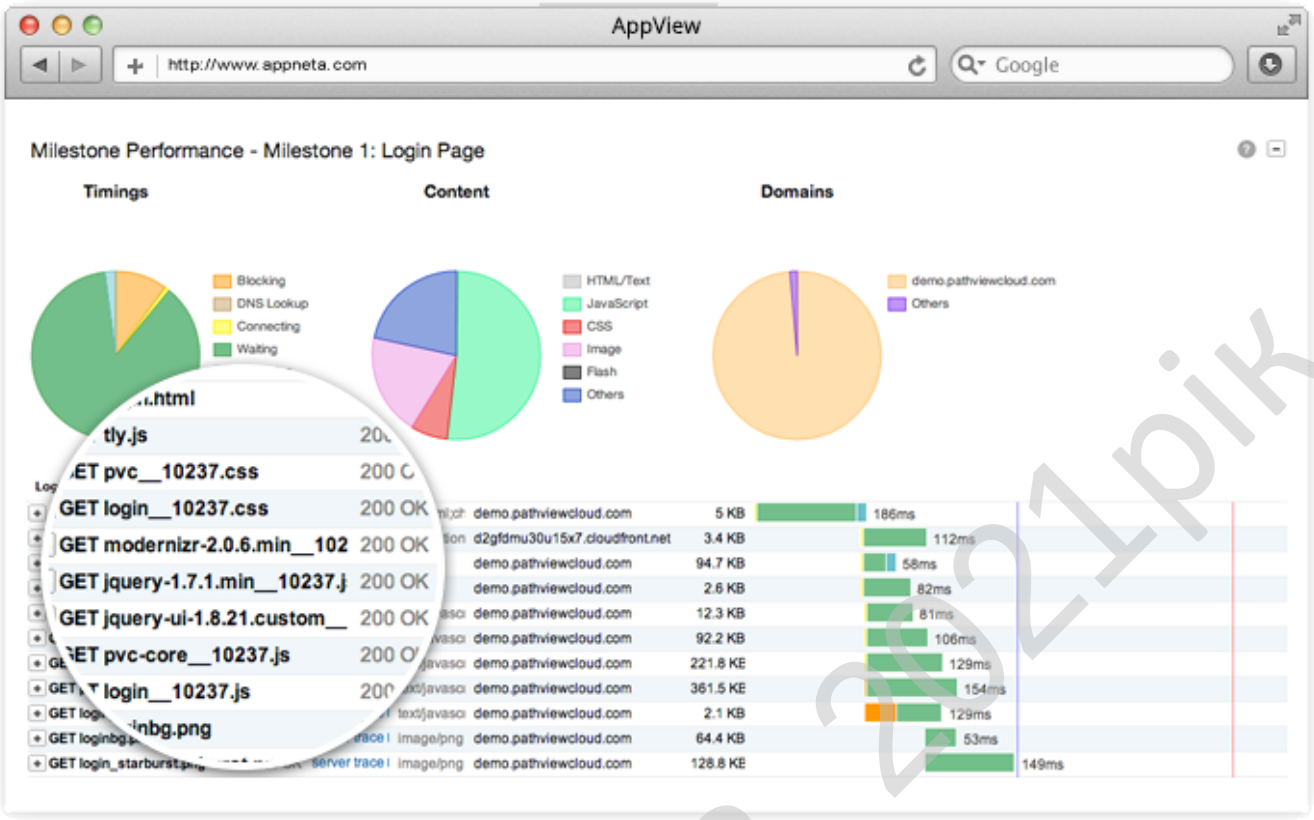

Рисунок 2.1 – Інтерфейс користувача AppView

Застосунок FlowView

Наступний основний застосунок – FlowView, яке надає інформацію, також з точністю до 5 хвилин за останні 90 днів, про використання каналів зв'язку. Чим завантажені канали, який обсяг трафіку і які найбільше часто використовувані застосунки. Застосунок працює на основі апаратних сенсорів (на швидкості до 10 Гбіт/с), які підключаються до маршрутизатора й збирають статистику про використання каналів зв'язку. Бібліотека включає близько 1000 передналаштованих застосунків, а також дозволяє створювати свої застосунки для контролю. Здобуваючи додаткову ліцензію, користувачі зможуть здійснювати захват трафіку для його наступного аналізу.

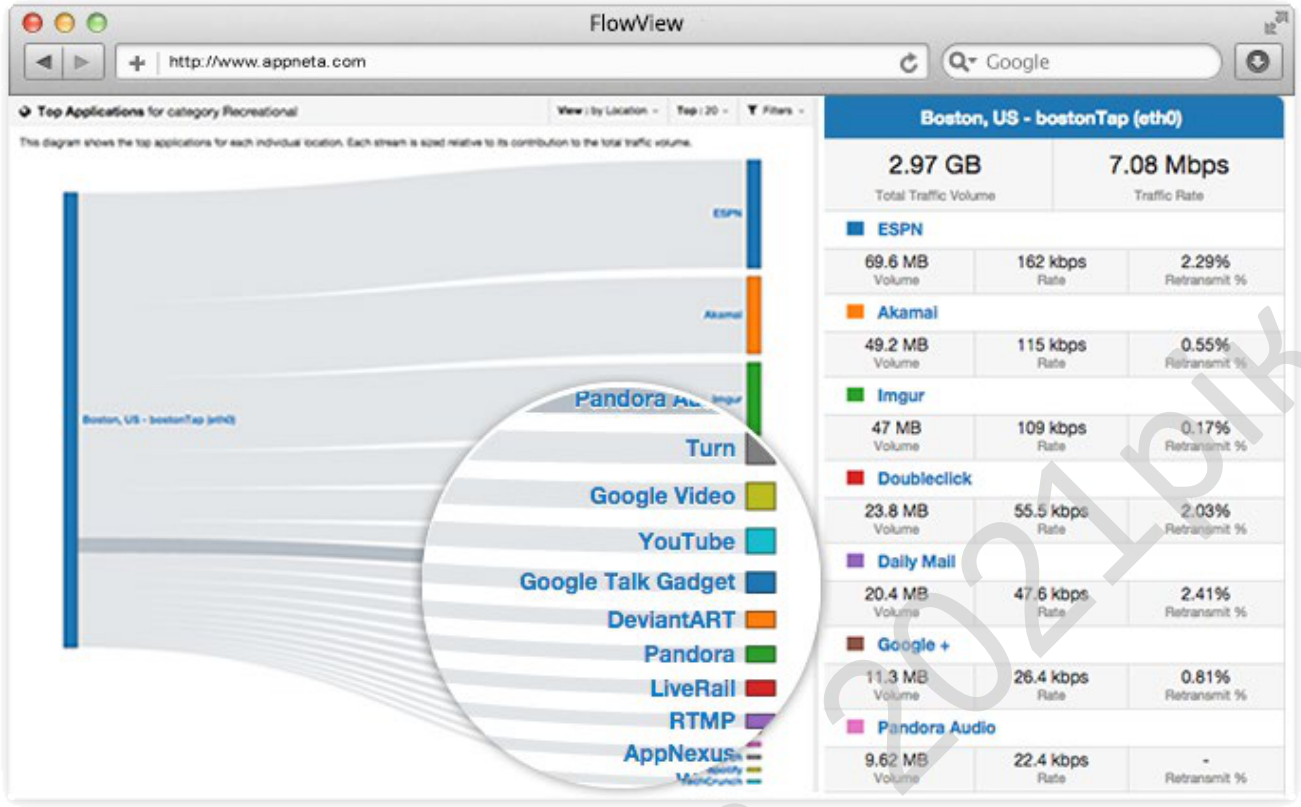

Рисунок 2.2 – Інтерфейс користувача FlowView

Застосунок PathView

Застосунок PathView дозволяє побудувати систему моніторингу доступності каналів зв’язку й виміряти втрати, джиттер, затримку, час передачі по мережі з точністю до 1 хвилини, а також контролювати будь-які зміни MTU і QoS для каналів зв'язку. Для роботи системи необхідно встановити пробники на кожному вилученому майданчику.

PathView використовує метод дисперсійного аналізу для виміру ємності каналів зв’язку. Пробник відсилає два тестові пакети невеликої довжини й вимірює затримку між приходом першого й другого. Далі розмір пакета ділиться на обмірювану дельту, і одержуємо пропускну здатність каналу в бітах у секунду. Цей метод використовується при вимірі каналу між двома зондами. Якщо ми прагнемо перевірити канал, на іншому кінці якого відсутній зонд, то

використовуються стандартні ICMP пакети. Далі пробник вимірює час між приходом відповідей на Ping запит.

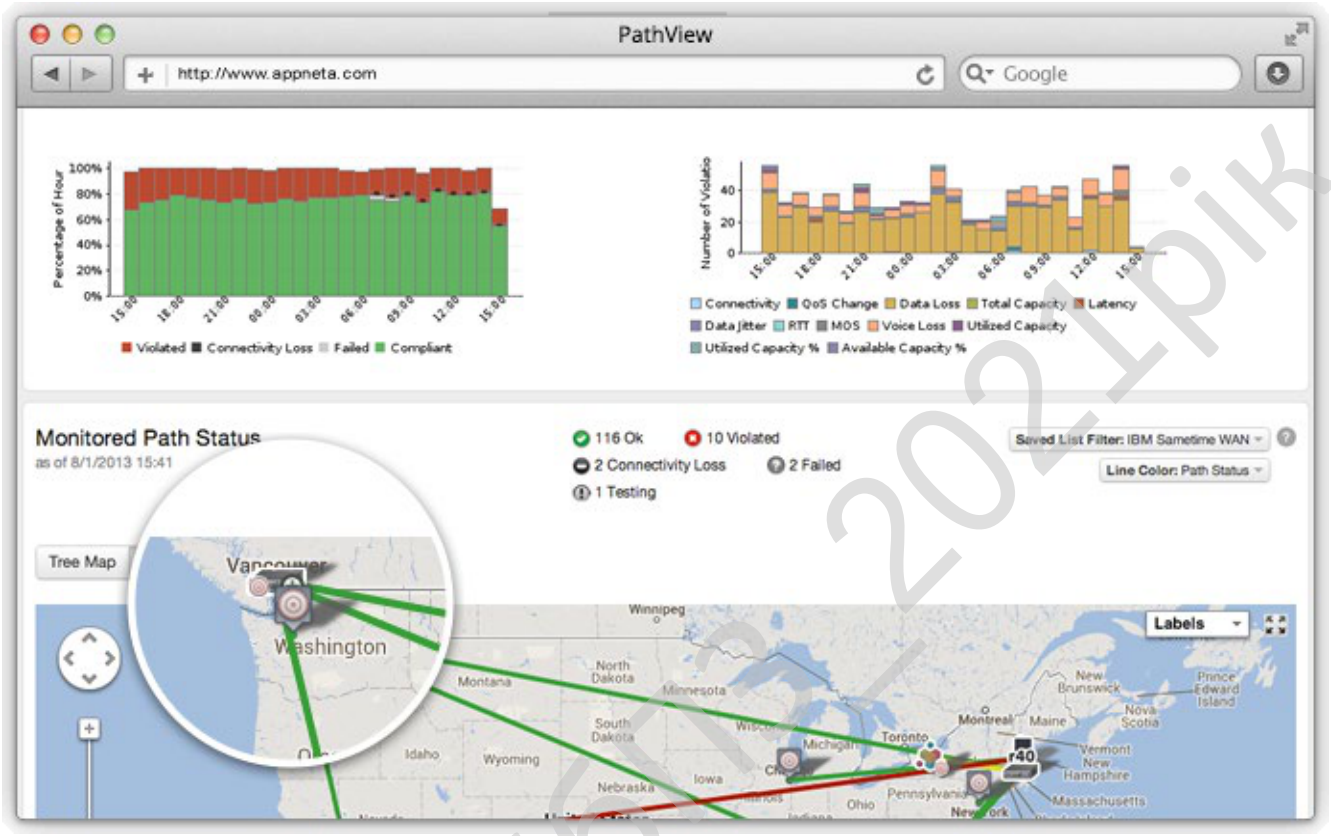

Рисунок 2.3 – Інтерфейс користувача PathView

Переваги й недоліки застосунків AppNeta

Переваги застосунків AppNeta:

– Повний спектр застосунків, які дозволяють побудувати систему моніторингу, як застосунків, так і мережі.

– Застосунок поставляється як абонентський сервіс на основі Saas хмари, що не вимагає більших первісних інвестицій в устаткування.

– Можливості PathView, включаючи розподілену модель моніторингу, дозволяє побудувати повноцінну систему моніторингу всіх сервісів для територіально розподілених мереж.

– Підхід AppNeta відрізняється від класичних систем моніторингу продуктивності мережі й застосунків, орієнтованих на технічних фахівців і детальний аналіз трафіку.

Недоліки застосунків AppNeta:

– Часто міняється керівництво компанії й це впливає на зсув фокуса компанії.

– Різні застосунки хоч і зв'язані логічно один з одним, але не мають єдиного інтерфейсу, що приводить до необхідності перемикатися між вікнами й будувати цілісну картину в голові фахівця, який проводить аналіз.

Огляд застосунків Viavi Solutions: Observer Matrix і Observer Taps

Viavi Solutions Inc. вийшла на ринок застосунків для моніторингу продуктивності мережі (NPMD) після придбання в 2013 році компанії Network Instruments і зайняла по праву лідируюче місце в магічному квадранті Gartner. Лінійка застосунків Viavi Observer широко відома серед професіоналів, як надійний застосунок для глибокого аналізу пакетів.

Платформа Observer полягає й декількох модулів, які роблять застосунок життєздатним і закінченим для глибокого аналізу пакетів з метою аналізу продуктивності мереж (NPMD).

Viavi Observer Matrix і Observer Taps

Observer Matrix і Observer Taps є застосунками для збору трафіку й потоків даних у різних місцях мережі. Після підключення відгалужувачів усі дані передаються в чотири різні застосунки для їхнього аналізу:

– Observer Analyzer.

– Observer Gigastor.

– Observer Probes.

– Observer Sightops.

Узагальнена статистика далі може бути передана в єдиний портал Observer Apex для побудови звітів, аналізу статистики й настроювання тригерів для повідомлень і аналітики. Платформа може працювати у вигляді програмно-

апаратного комплексу під операційною системою Windows або у вигляді віртуальних машин.

Які модулі входять у платформу Observer?

Observer Apex надає централізований портал для аналізу продуктивності мережі, що одержує метадані від інших модулів платформи Observer.

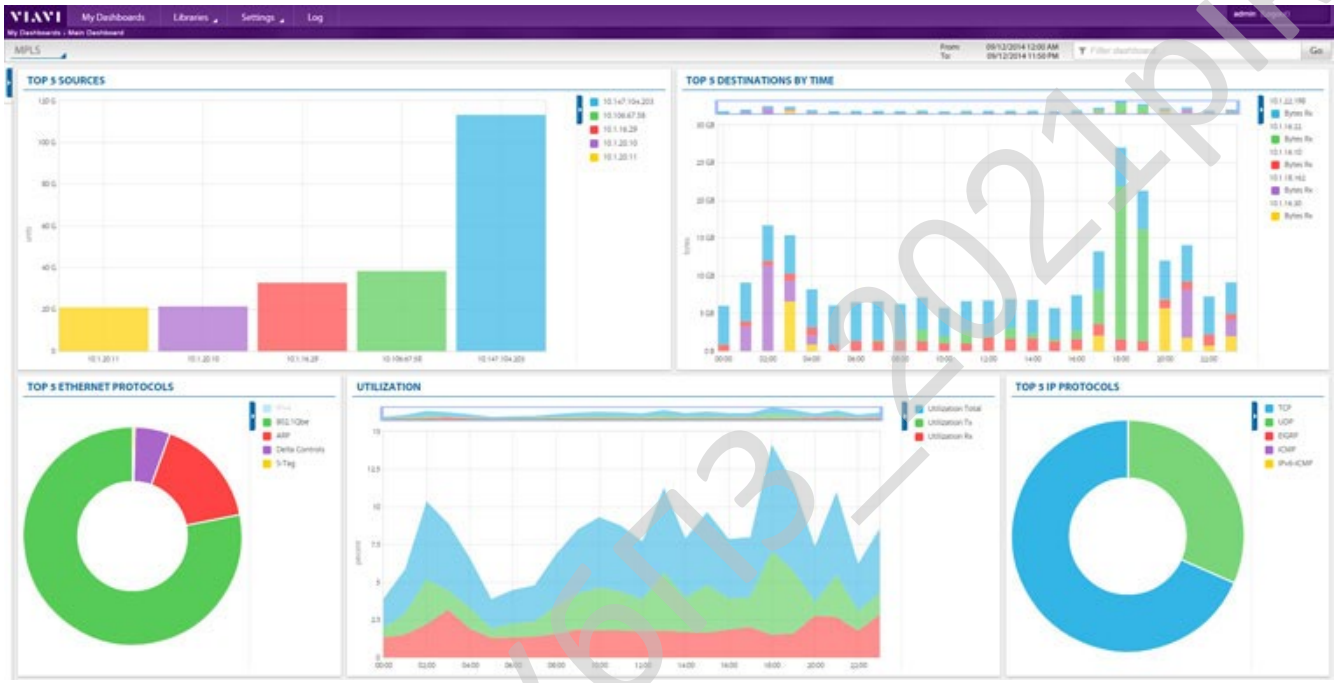

Рисунок 2.4 – Інтерфейс користувача Observer

Він агрегує дані й у зручному форматі надає графічні звіти для оцінки стану IT-інфраструктури й сервісів. За допомогою інтерфейсу Restful API він може інтегруватися з високорівневими системами моніторингу мережі й бізнеси-аналітики.

Apex пропонує користувачам наочні схеми, які показують зв'язки між користувачами, серверами й додатками, що дозволяє локалізувати збійна ділянка суттєво швидше.

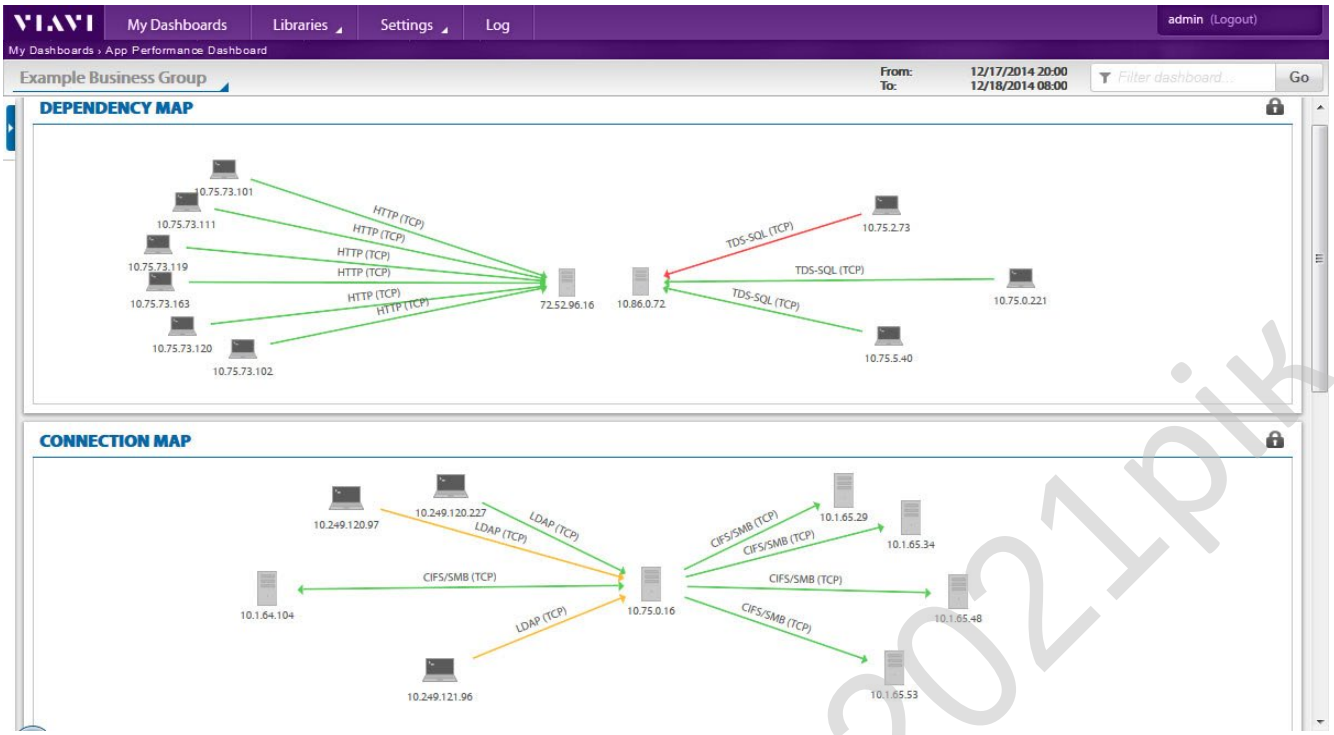

Рисунок 2.5– Інтерфейс користувача Apex

Для того щоб Apex побачив метадані необхідно налаштувати трендинг у модулях Observer і далі в самому Apex налаштувати необхідні віджети.

Observer Analyzer є робочою конячкою всієї платформи Observer. Застосунок поєднує в собі можливості моніторингу мережі з такими функціями як аналіз застосунків, декодування трафіку, розпізнавання голосових і відео потоків. Довгий час Observer Analyzer поставлявся як класичний аналізатор протоколів із вбудованою експертною системою, яка суттєво розширює можливість безкоштовних аналізаторів, типу Wireshark.

Застосунок дуже корисний для IT-фахівців, тому що дозволяє оцінити вплив на продуктивність мережі нових релізів програмного забезпечення, застосунків для передачі відео й голосу (telepresence) на продуктивність, а також для розуміння, у чому причина зниження продуктивності – у мережі або в додатках. Observer Analyzer працює як у реальних, так і віртуальних середовищах і підтримує хмарні технології.

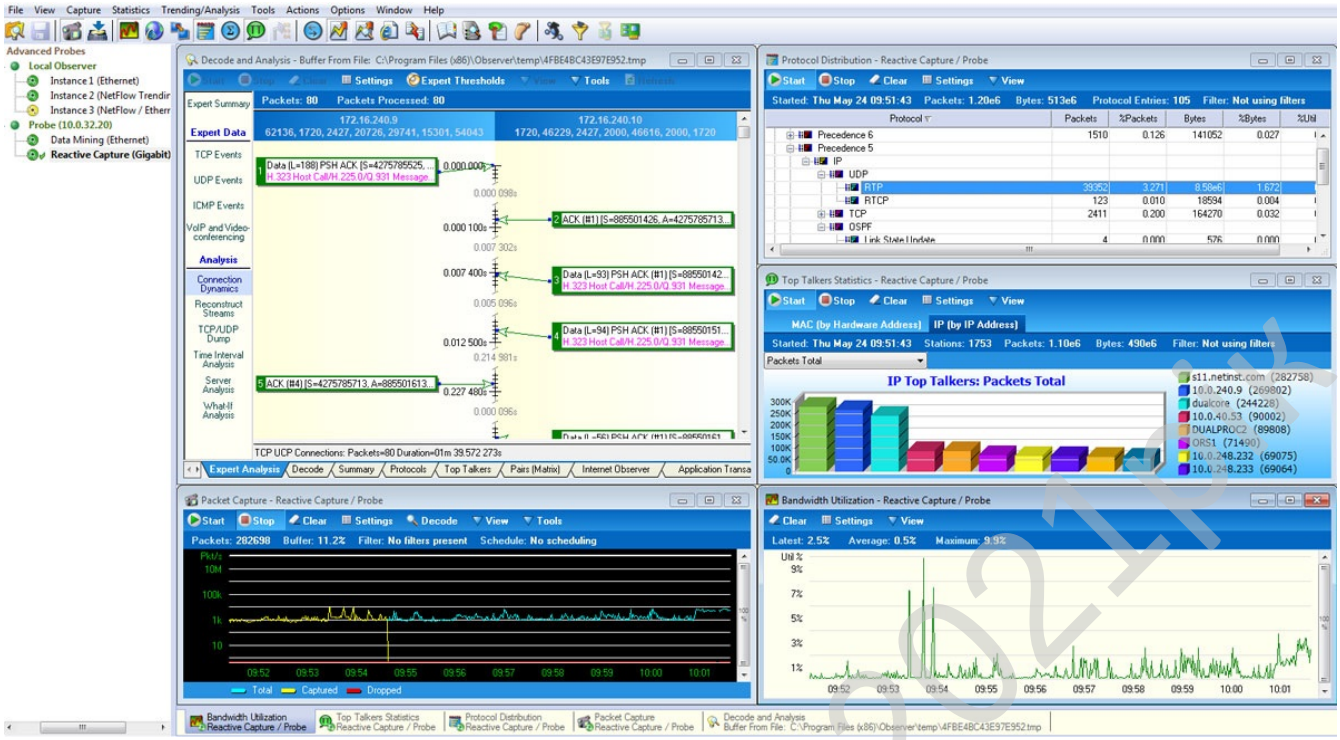

Рисунок 2.6– Інтерфейс користувача Observer Analyzer

Observer Gigastor – апаратні буфери для тривалого захвата, індексування й зберігання даних і застосунку завдань безпеки або аналізу продуктивності. Застосунок може поставлятися с інтерфейсами на 1 Гбіт/сек, 10 Гбіт/сек і 40 Гбіт/сек і розміром від 2U до 5U з максимальним обсягом зберігання до 1,15 Пбайт. Це єдиний застосунок на ринку з таким доступним обсягом зберігання трафіку. Для завдань моніторингу досить зберігати тільки заголовки (без корисного навантаження), що суттєво розширює обсяги зберігання даних.

Огляд Observer Gigastor

Observer Sightops – засіб автоматизації для контролю стану й продуктивності гібридних IT, коли інфраструктура складається із власних і хмарних ресурсів.

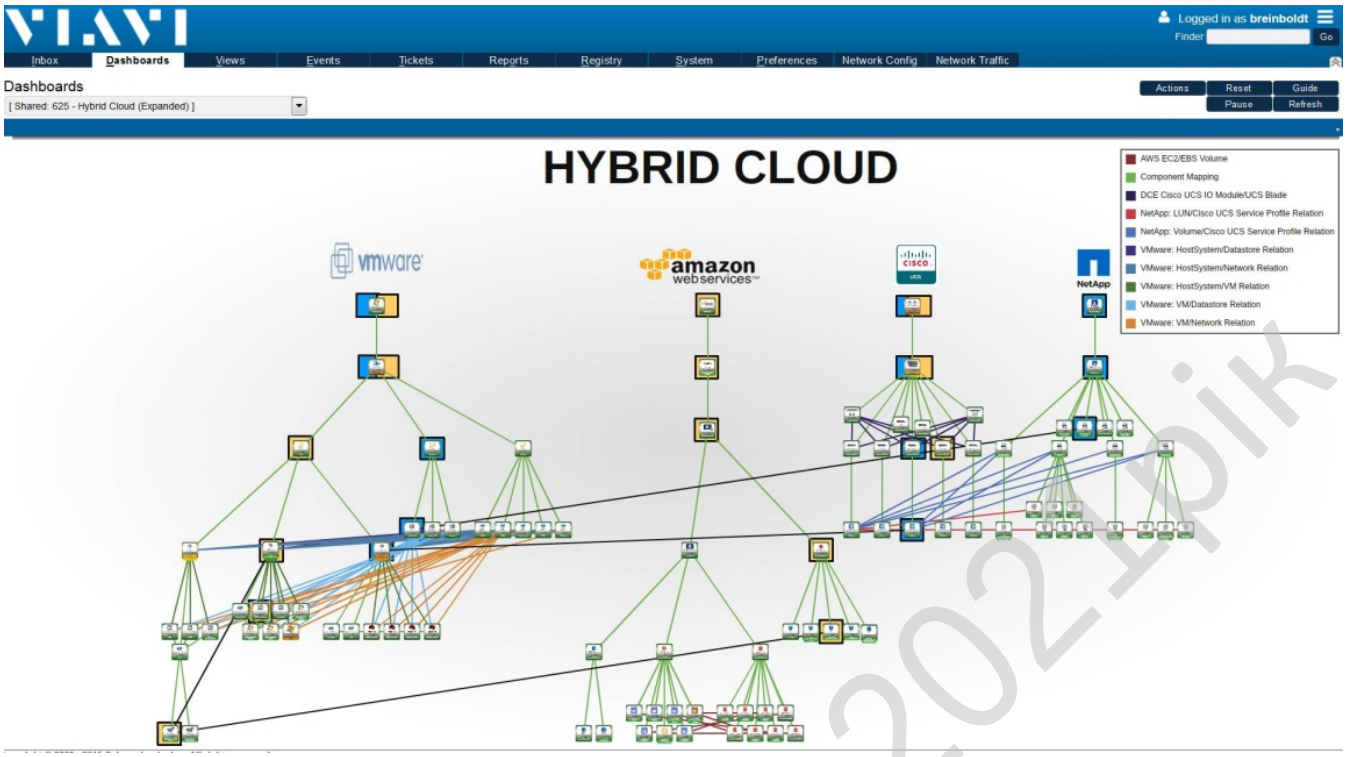

Рисунок 2.7 – Інтерфейс користувача Observer Sightops

Дозволяє контролювати такі доступні ресурси, як: дисковий простір, завантаження пам'яті, процесора і т.д. Завдяки власним алгоритмам може будувати карти взаємодії між пристроями й аналізувати вплив окремих елементів гібридної IT-інфраструктури на продуктивність сервісу. При спільній інтеграції з Viavi Gigastor дозволяє дістати з нього трафік, який ставиться до взаємодії між певними пристроями для вирішення проблем.

Observer Matrix і TAP – застосунку для інтелектуального й пасивного відгалуження трафіку на швидкості до 10G. Лінійка включає один інтелектуальний відгалужувач Matrix з можливістю балансування трафіку, агрегування й реплікації потоків, а також кілька моделей пасивних мідних і оптичних відгалужувачів.

Продуктовий портфель Viavi Solutions має всі необхідні модулі для побудови повноцінної системи для аналізу продуктивності мережі на основі моніторингу за реальним мережним трафіком, що дозволяє по оцінках аналітиків продавати до 150 млн. доларів у рік застосунків даного класу.

Застосунок Gigastor Software від компанії Viavi Solutions контролює роботу віртуальних і хмарних ресурсів

Рисунок 2.8 – Інтерфейс користувача Gigastor Software від компанії Viavi Solutions

Компанія Viavi Solutions Inc., раніше відома як VIAVI випустила у світло новий застосунок – «Gigastor Software Edition». Це програмне забезпечення призначене для роботи із програмно-конфігуруємими мережами (SDN). Воно дозволяє аналізувати продуктивність і поліпшувати роботу IT інфраструктури під завдання користувачів при роботі у віртуальних і хмарних середовищах.

Програмно-конфігуруємі мережі (SDN) – це сучасний підхід до побудови мереж, у якому весь інтелект мережі винесений на окрему апаратну/програмну базу, а все керування трафіком відбувається на основі спеціальних протоколів, які оперують поняттям «потік» (Flow) і можуть робити різні дії з ним (дозволити,

заборонити, перенаправляти, переписати поля в пакетах і т.д.). Фактично, на контролері визначається політика керування мережею на основі заданих правил, а також роботи спеціалізованих застосунків (наприклад, емулюючих роботу STP або протоколів маршрутизації). Потім кінцевий результат передається на комутатори по протоколу у вигляді Flow-таблиць, що містять інформацію про той куди, як і який трафік передавати. З одного боку, такий підхід дає більшу гнучкість у керуванні мережею, з іншого боку – суттєво спрощує адміністрування (і, почасти, архітектуру) мережі.

Застосунок Gigastor Software дозволяє захопити пакети обсягом від 250 Гбайт до 1 Тбайта для наступного аналізу збереженого трафіку. Замість того щоб витрачати час для підключення зондів або аналізаторів протоколів і намагатися відтворити проблеми, Gigastor Software дозволяє легко повернутися в минуле й подивитися, що відбувалося в момент, що цікавить нас. Можливість подорожі в часі особливо важлива у двох випадках:

- коли проблема складна, повторюється нерегулярно;
- коли має місце порушення безпеки.

Застосунок Gigastor Software дозволяє мережним інженерам контролювати й аналізувати трафік у віртуальних середовищах. Воно встановлюється на гіпервізор VMware ESX, а також може інтегруватися з будь-якими існуючими апаратними зондами Gigastor.

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 кібербезпеки NPMD та APM.

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 кібербезпеки, 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

NPMD

Застосунки NPMD дозволяють IT-фахівцям краще розуміти й усувати проблеми із продуктивністю застосунків, мережі й окремих компонентів IT-інфраструктури, що дозволяє розуміти якість надаваних сервісів і роботу застосунків з погляду їх користувачів. Таким чином, фокус даних застосунків спрямований саме на досвід користувачів сервісів, а не на стан серверів, комутаторів або віртуальних машин (класичні SNMP застосунки). Таким чином, за допомогою застосунків NPMD ми можемо зрозуміти, у чому основна причина погіршення сервісу або роботи застосунку, а також оцінити, як ухвалені рішення щодо оптимізації IT-інфраструктури вплинули на продуктивність ключових застосунків і сервісів.

Застосунки NPMD звичайно є частиною загальної стратегії побудови системи керування й моніторингу за роботою IT-інфраструктури й тому при виборі його рекомендую брати до уваги наступні фактори:

– Стратегія виробника застосунку, постійний його розвиток і вдосконалювання, тому що застосунки й IT-інфраструктура стають тільки складніше.

– Можливості інтеграції застосунків NPMD із системами OSS/BSS, як частина загальної стратегії розвитку системи керування й моніторингу IT-інфраструктури

– Швидкість впровадження, простота використання застосунку й зручність створення звітів для ухвалення рішення як IT-фахівцями, так і бізнесом.

– Підтримка застосунками NPMD можливості моніторингу хмарних технологій хааS, тому щ о усе більше компаній рухаються від приватних хмар убік гібридних хмар з метою оптимізації витрат.

– Багато застосунків на ринку поки не підтримують SDN мережі, і хто перший розв'яже це завдання – стане безумовним лідером

– Застосунок NPMD повинне не тільки мати можливість збору інформації на основі реального трафіку, NetFlow, SNMP і т.д., але й мати безумовну кореляцію даних.

– Не варто орієнтуватися на опис застосунку на сайті, робіть пілот і ви зможете переконатися в тому, що ви купуєте допомогу, а не черговий головний біль. Опис систем як близнюки брати, але сукупна вартість володіння, реальні функціональні можливості й зручність використання дуже сильно відрізняються.

APM

Вимоги до властивостей, якими повинні мати застосунки для аналізу продуктивності застосунків (APM), наступні:

– Моніторинг роботи застосунків кінцевих користувачів (End User Experience Monitoring) – захват даних для наскрізного контролю затримок у роботі застосунку, коректності і якості обробки запитів, яке випробовує її користувач. Контроль доступності застосунку, у тому числі й шляхом виконання синтетичних запитів, що імітують запити реальних користувачів.

– Розуміння й візуалізація топології застосунку – огляд і візуалізація апаратних і програмних компонентів, які задіяні в роботі застосунку й безлічі можливих шляхів, за допомогою яких можуть взаємодіяти компоненти при роботі застосунку.

– Профілювання користувацьких транзакцій – відстеження згрупованих по користувачеві подій, які ставляться до транзакцій користувача, які виникають у міру їх виконання й взаємодії апаратних і програмних компонентів.

– Глибокий аналіз прикладних компонентів – аналіз ресурсів і подій, які виникають на сервері при роботі застосунку.

– Аналіз отриманих даних – комбінація або використання наступних методик: комплексна обробка виявлених подій, статистичних закономірностей, індексація неструктурованого тексту, топологічний аналіз, багатомірні масиви даних з можливостями пошуку й аналізу накопичених даних по роботі застосунків.

3.2 Розробка структурної схеми

Для моніторингу продуктивності мережі (NPMD) і застосунків (APM) у даній роботі побудований застосунок у вигляді платформи NPMD_APM. По суті це веб-портал, до якого прикручуються модулі. Це робить складним розуміння сукупної вартості володіння застосунком, тому що застосунок дійсний може все, але це «все» виходить за досить великі гроші.

Під парасолькою NPMD_APM перебуває кілька модулів, який виконують різні функції NPMD і APM:

– моніторинг доступності інфраструктури;

– моніторинг застосунків;

– глибокий аналіз пакетів (DPI);

– глибокий аналіз потоків (Flow аналіз).

Які модулі входять в NPMD_APM?

NPMD_APM включає модулі як для ринку SMB, так і для великих корпорацій з територіально розподіленими мережами.

NPMD_APM Netexpress призначене для ринку SMB або для моніторингу вилучених офісів при розподіленій мережній інфраструктурі. Застосунок «усе в одному» дозволяє здійснювати: аналіз потоків даних, захвата трафіку, побудови звітів по використанню каналів зв’язку. Застосунок дозволяє визначити, які застосунки передаються по мережі, зв'язки між ними й з якими пріоритетами вони передаються.

Джерелом інформації може бути:

– живий трафік або NetFlow (V1, V5, V7, V9).

– IPFIX.

– Enhanced NetFlow.

– NBAR.

– sFlow.

– J-Flow.

– cFlow.

– Packeteer FDR.

– Citrix AppFlow.

– Palo Alto Networks.

– Cisco NBAR2.

– Cisco MediaNet.

– Cisco ASA NSEL.

– Flow Net.

Доступно п'ять конфігурацій застосунку передвстановленного на сервер або у вигляді віртуального образа під VMware залежно від кількості потоків у хвилину – від 15000 до 120 000. NPMD_APM AppResponse дозволяє забезпечити наскрізний моніторинг продуктивності бізнес застосунків для всіх користувачів з точністю до хвилини, а також моніторинг таких сервісів, як Voip і відео (Cisco Telepresence, Cisco Tandberg і Polycom). Не вимагає установки яких-небудь агентів, а весь аналіз і статистику будує на обробки живого трафіку. Якщо корпоративні користувачі використовують бази даних Oracle, SQL Server, DB2/UDB, Teradata, Sybase ASE і Informix, то купивши додаткову ліцензію можна оцінити вплив роботи бази даних на продуктивність сервісу, а також зрозуміти який запит до бази виконав користувач і викликав проблеми в роботі застосунку в цілому. Доступно кілька конфігурацій застосунку передвстановленного на сервер (поділ по типу й кількості мережних інтерфейсів і обсягу сховища) або у вигляді віртуального образа під VMware.

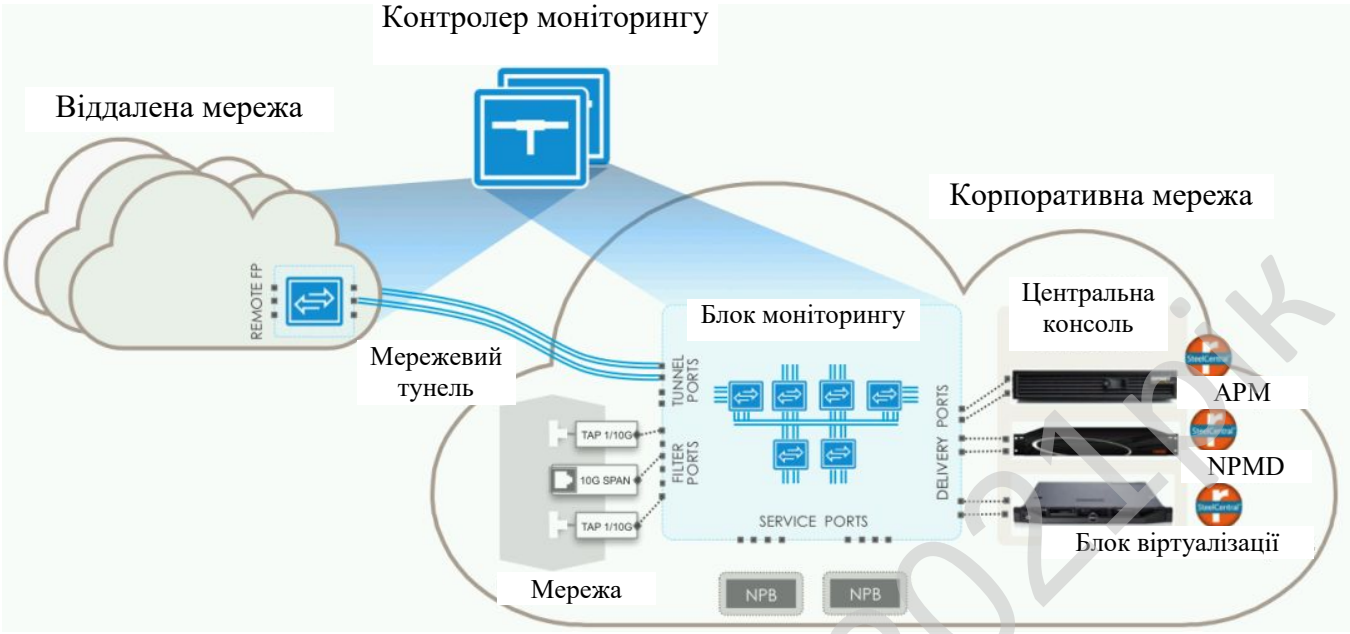

Рисунок 3.1 – Структурна схема системи

NPMD_APM AppInternals фокусується на аналізі користувацьких транзакцій для вирішення проблем у роботі самого застосунку на рівні коду. Легко справляється з великими обсягами даних і дозволяє застосунати глибинні проблеми в коді, які впливають на продуктивність сервісу.

NPMD_APM NetProfiler поєднує дані по потоках (Flow) з пакетним аналізом трафіку. Далі на основі отриманих даних від декількох джерел дозволяє зрозуміти, на скільки ефективно завантажені канали зв'язку, і як це впливає на продуктивність ключових сервісів. Джерелом даних для NetProfiler є:

- NPMD_APM NetShark – захват трафіку.
- NPMD_APM Flow Gateway – збір Flow. Підтримка: NetFlow (V1, V5, V7, V9), IPFIX, Enhanced NetFlow, NBAR, sFlow, J-Flow, cFlow, Packeteer FDR, Citrix AppFlow, Palo Alto Networks, Cisco NBAR2, Cisco MediaNet, Cisco ASA NSEL і Flow Net.
- Head – Заголовок оптимізації.
- Fusion – Функція оптимізації.

NPMD_APM NetProfiler використовує стандартні алгоритми ідентифікації застосунків і взаємодії між ними, що дозволяє оцінити базову продуктивність каналів зв'язку й реагувати на відхилення від неї.

NPMD_APM NetSensor – модуль, який на основі протоколу SNMP і WMI дозволяє контролювати стан окремих елементів IT інфраструктури, наприклад, доступність серверів, завантаження процесорів, пам'яті, статус інтерфейсів і т.д.

NPMD_APM NetShark – захват і зберігання поточного трафіку для наступного аналізу й аналізу трендів.

NPMD_APM Portal призначений для агрегації даних, отриманих від різних модулів у єдиному Web порталі. Екранні форми можуть бути налаштовані під окремих адміністраторів для застосунку з конкретних завдань.

Застосунок NPMD_APM – це велика парасолька, яка включає багато модулів. При виборі й аналізі застосунку обертайте на цю увагу, а краще тестуйте, щоб зрозуміти, що вам дійсно потрібно, а без чого можна обійтися.

3.3 Розробка функціональної схеми

Вимоги до властивостей, якими повинні мати застосунок для аналізу продуктивності застосунків містять у собі:

– Моніторинг роботи застосунків кінцевих користувачів (End User Experience Monitoring) – захват даних для наскрізного контролю затримок у роботі застосунку, коректності і якості обробки запитів, яке випробовує її користувач. Контроль доступності застосунку, у тому числі й шляхом виконання синтетичних запитів, що імітують запити реальних користувачів.

– Розуміння й візуалізація топології застосунку – огляд і візуалізація апаратних і програмних компонентів, які задіяні в роботі застосунку й множини можливих шляхів, за допомогою яких можуть взаємодіяти компоненти при роботі застосунку.

– Профілювання користувацьких транзакцій – відстеження згрупованих по користувачеві подій, які ставляться до транзакцій користувача, які виникають у міру їх виконання й взаємодії апаратних і програмних компонентів.

– Глибокий аналіз прикладних компонентів – аналіз ресурсів і подій, які виникають на сервері при роботі застосунку.

– Аналіз отриманих даних – комбінація або використання наступних методик: комплексна обробка виявлених подій, статистичних закономірностей, індексація неструктурованого тексту, топологічний аналіз, багатомірні масиви даних з можливостями пошуку й аналізу накопичених даних по роботі застосунків.

Функціонально система складається з наступних модулів:

1. Блок інтеграції й надання інформації.

2. Блок NPMD:

– Моніторинг мережного трафіку.

– Агрегація різних джерел даних (Flow).

– Аналітика, звіти.

3. Блок APM:

– Моніторинг функціонала застосунків.

– Відстеження досвіду користувачів.

– Кореляція подій, аналітика.

4. Блок оптимізації:

– Видимість WAN.

– Аналіз QoS.

5. Блок устаткування:

– Моніторинг SNMP.

– Керування конфігурацією.

6. Блок обробки трафіку:

– Захват пакетів і їх аналіз.

– Добування потоків з пакетів.

– Добування транзакцій з пакетів.

7. Блок роботи із клієнтами.

– Моніторинг користувацького досвіду.

8. Блок роботи із серверами:

– Моніторинг якості коду.

– Моніторинг ресурсу серверів.

– Кореляція послідовності викликів методів.

– Аналітика використання.

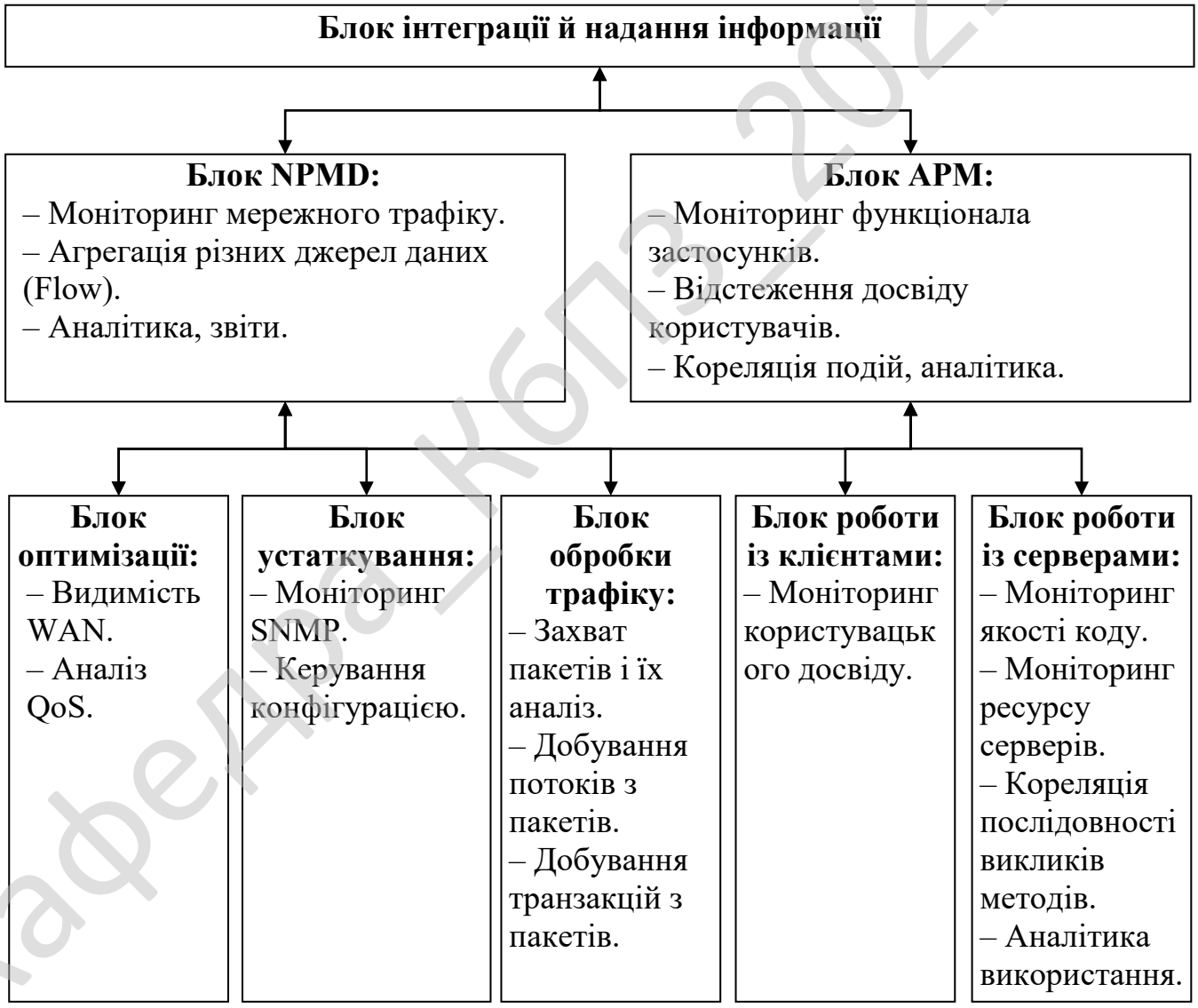

Рисунок 3.2 – Функціональна схема систем

Розглянемо які параметри і як виміряються при аналізі продуктивності сервісів і застосунків. Отже, більшість корпоративних застосунків як транспорту використовують протокол TCP, який орієнтований на встановлення з'єднання і його надійній підтримці з контролем доставки пакетів. Тому аналізуючи дельти за часом між певними пакетами можна зібрати всі основні метрики:

– CT (Connection Time) – час установлення з'єднання із сервером. Цей параметр наочно відбиває затримки в мережі, тому що пакети в цей момент передаються малої довжини й із пріоритетом, налаштованим для даного застосунку. Пакети при встановленні з'єднання обробляються сервером/клієнтом також з високим пріоритетом. Деякі системи моніторингу на цій стадії дають розгорнуту оцінку, і ділять час CT на складові: RTT і SRT і CD (Client Delay). Це дозволяє оцінити за часом SRT, чи зайнятий сервер і на скільки в момент установлення з'єднання.

– TTFB (Time To the First Byte) – час до відправлення першого байта. Не всі системи його вимірюють, але деякі користувачі знаходять у ньому інтерес. На наш погляд, воно включає багато складових і більше вносить сум'яття.

– DTT (Data Transfer Time) – час необхідний для передачі всіх даних від клієнта або сервера на його запит. Високі значення даного параметра треба аналізувати разом з оцінкою обсягу запиту, тому що якщо значення високе, те це звичайно перевантажений додаток. Чим? Або кількістю запитів або їх обсягом. Також необхідно при аналізі дивитися на наявність помилок на TCP рівні – Windows Zero Size, TCP Reset і т.д. Тому що ці TCP прапори суттєво впливають на швидкість передачі даних.

– RTT (Round Trip Time) – круговий час передачі по мережі. По суті, дельта між відправленням пакета з боку клієнта або сервера й одержанням підтвердження на цей пакет (ACK). Залежно від крапки захвата реального трафіку може відбивати час передачі до клієнта або до сервера. Частіше це дельта за часом між пакетом, який містить дані на запит і першим підтвердженням його одержання (ACK). Головний параметр для оцінки мережний складової IT-

інфраструктури, тому що цей чистий час передачі по мережі плюс невелика складова на рівні погрішності, – це Client Delay.

– SRT (Server Response Time) – час необхідний серверу на підготовку першого пакета з даними у відповідь на запит клієнта. Основний параметр, який показує затримку сервера й далі необхідно аналізувати, чим зайнятий сервер, використовуючи й інші застосунки, наприклад, на основі SMNP або SCOM.

– RR (Retransmission rates) – найбільш наочний індикатор, який показує втрати пакетів у каналах зв'язку або чергах активного встаткування. Багато фахівців при аналізі втрати пакетів виключають факт перевірки настроювань активного встаткування, а на ньому може бути відключена фрагментація пакетів, некоректно налаштований MTU, час життя і т.д.

Час відгуку сервісу для клієнта, це сума всіх складових і як видне на діаграмі становить 90 мс. Й наше завдання при аналізі продуктивності знайти слабку ланку й усунути його.

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.

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.

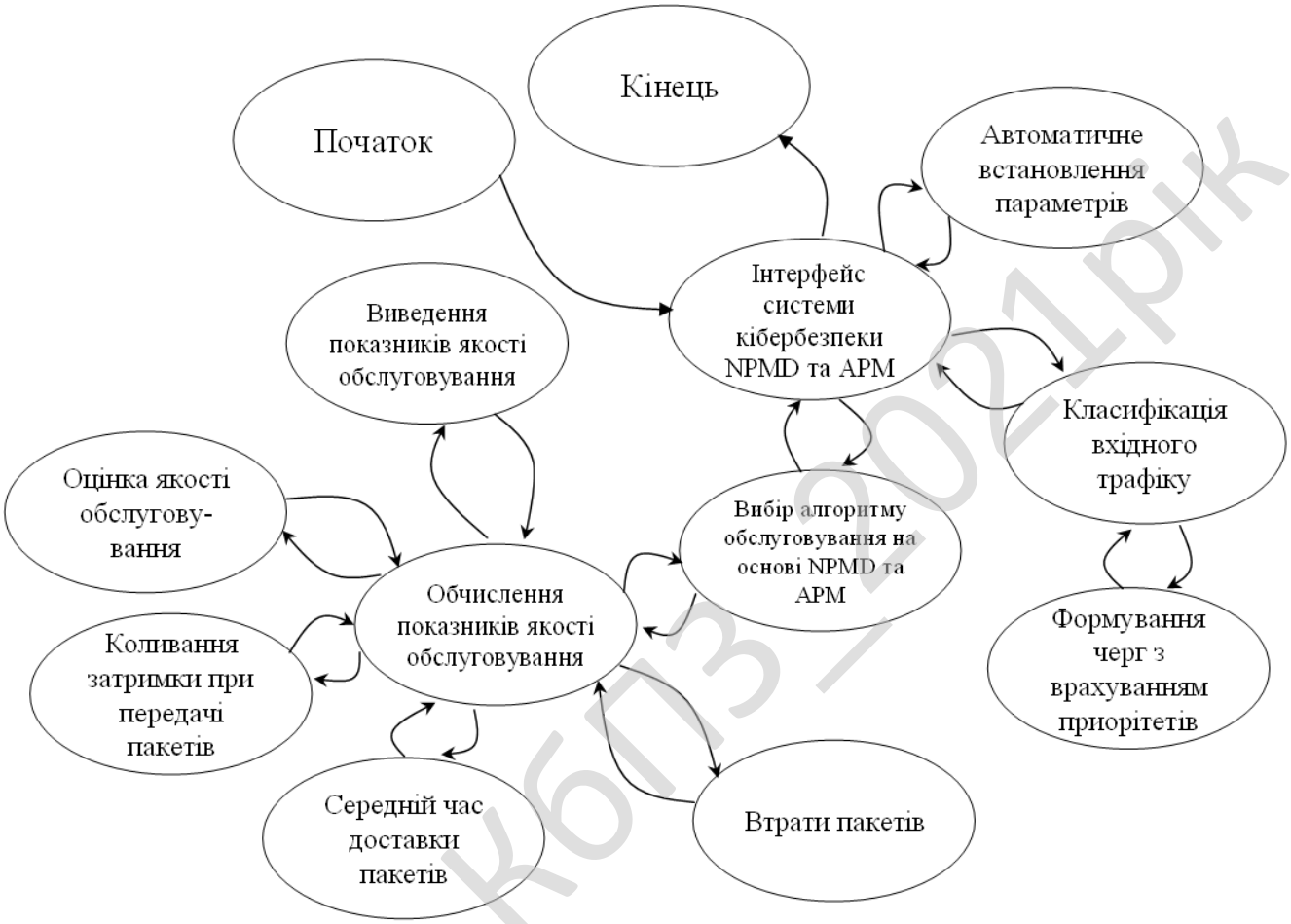

Рисунок 3.3 – Діаграма взаємодії процесів

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.

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

У розділі розглянемо реалізацію бакалаврської дипломної роботи та наведені приклади блок-схем та опис алгоритмів функціонування системи у вигляді частин вихідного коду.

Розглянемо реалізацію частини системи кібербезпеки який показує приклад відображення статистики з'єднання з Інтернет ресурсами в графічному вигляді за допомогою виконання команди PING і перевірки повертається параметра TTL. Вихідний код наступний.

```
unit fMNM;

interface
// Описова частина
uses
// підключення модулів
Windows, Messages, Graphics, Controls, Forms, Dialogs, StdCtrls, ExtCtrls,
SysUtils, Classes, IdIcmpClient, IdBaseComponent, IdComponent, IdRawBase,
IdRawClient, Spin, Grids, TeeProcs, TeEngine, Chart, shellapi,series,
Buttons, XPMan;

Type
// опис класу
  TfrmPing = class(TForm)
    lstReplies: TListBox;
    ICMP: TIdIcmpClient;
    Panel1: TPanel;
    btnPing: TButton;
    edtHost: TEdit;
    StringGrid1: TStringGrid;
    Timer1: TTimer;
    ComboBox1: TComboBox;
```

```
    Label1: TLabel;
    Button1: TButton;
    Label2: TLabel;
    Chart1: TChart;
    Series2: TLineSeries;
    Series3: TLineSeries;
    Memo1: TMemo;
    Label3: TLabel;
    StringGrid2: TStringGrid;
    BitBtn1: TBitBtn;
    Button2: TButton;
    Label4: TLabel;
    XPManifest1: TXPManifest;
    procedure btnPingClick(Sender: TObject);
    procedure ICMPReply(ASender: TComponent;
    const ReplyStatus: TReplyStatus);
    procedure FormCreate(Sender: TObject);
    procedure Timer1Timer(Sender: TObject);
    procedure Button1Click(Sender: TObject);
    procedure Label1Click(Sender: TObject);
    procedure Chart1ClickSeries(Sender: TCustomChart; Series: TChartSeries;
      ValueIndex: Integer; Button: TMouseButton; Shift: TShiftState; X,
      Y: Integer);
    procedure BitBtn1Click(Sender: TObject);
  private
  public
  end;

var
  frmPing: TfrmPing;
   i, j, tmp:integer;
   minTTL, maxTTL, meanTTL, minTrip, maxTrip, meanTrip :integer;
   sommeTTL, sommeTrip: integer;

implementation
{$R *.DFM}
// реалізація

// UDP запт
procedure TfrmPing.btnPingClick(Sender: TObject);
begin
Chart1.Title:=({'Ping Statistics for '+}edtHost.Text);
```

```
        Timer1.Interval:=(StrToInt(combobox1.text)*1000);
        If btnPing.caption='Start' then
        begin
          Timer1.Enabled:=True;
          btnPing.Caption:='Stop';
          Combobox1.Visible:=False;
          Label2.Visible:=False;
          edtHost.ReadOnly:=True;
          Label4.Visible:=False;
          end
          else
          begin
          Timer1.Enabled:=False;
          btnPing.Caption:='Start';
          Combobox1.Visible:=True;
          Label2.Visible:=True;
          edtHost.ReadOnly:=False;
          Label4.Visible:=True;
          end;
        end;

        // результат запиту
        procedure TfrmPing.ICMPReply(ASender: TComponent; const ReplyStatus:
TReplyStatus);
        var
          sTime: string;
        begin
        // TODO: check for error on ping reply (ReplyStatus.MsgType?)
        //-----------------------------------------------------
          lstReplies.Items.Add(Format('%d;%d',      //%s
            [
            ReplyStatus.TimeToLive,
        {     sTime,}
            ReplyStatus.MsRoundTripTime]));
        //-----------------------------------------------------
            StringGrid1.RowCount:=j+1;
            StringGrid1.Cells[0,j]:=DateTimeToStr(now);
            StringGrid1.Cells[1,j]:= (copy(lstReplies.Items.Text,0,
                                      pos(';',lstReplies.Items.Text)-1));
            StringGrid1.Cells[2,j]:= copy(lstReplies.Items.Text,pos(';',
                           lstReplies.Items.Text)+1,
                           length(lstReplies.Items.Text)-
```

```
                    (length(StringGrid1.Cells[1,j])+3));
//---------------------statistiques----------------------
    If StrToInt(StringGrid1.Cells[1,j])>maxTTL then
    begin
    maxTTL:=strToInt(StringGrid1.Cells[1,j]);
    StringGrid2.Cells[2,2]:=StringGrid1.Cells[1,j];
    StringGrid2.Cells[1,2]:=StringGrid1.Cells[0,j];
    end;
    If StrToInt(StringGrid1.Cells[1,j])<minTTL then
    begin
    minTTL:=strToInt(StringGrid1.Cells[1,j]);
    StringGrid2.Cells[2,1]:=StringGrid1.Cells[1,j];
    StringGrid2.Cells[1,1]:=StringGrid1.Cells[0,j];
    end;
    If StrToInt(StringGrid1.Cells[2,j])>maxTrip then
    begin
    maxTrip:=strToInt(StringGrid1.Cells[2,j]);
    StringGrid2.Cells[2,4]:=StringGrid1.Cells[2,j];
    StringGrid2.Cells[1,4]:=StringGrid1.Cells[0,j];
    end;
    If StrToInt(StringGrid1.Cells[2,j])<minTrip then
    begin
    minTrip:=strToInt(StringGrid1.Cells[2,j]);
    StringGrid2.Cells[2,5]:=StringGrid1.Cells[2,j];
    StringGrid2.Cells[1,5]:=StringGrid1.Cells[0,j];
    end;
    sommeTTL:=sommeTTL+strToInt(StringGrid1.Cells[1,j]);
    If j>1 then meanTTL:=round(sommeTTL/(j));
    StringGrid2.Cells[2,3]:=IntToStr(meanTTL);
    sommeTrip:=sommeTrip+strToInt(StringGrid1.Cells[2,j]);
    If j>1 then meanTrip:=round(sommeTrip/(j));
    StringGrid2.Cells[2,6]:=IntToStr(meanTrip);
//--------------------fin statistques--------------------
Memo1.Lines.Append(StringGrid1.Cells[0,j]+';'+StringGrid1.Cells[1,j]
+';'+StringGrid1.Cells[2,j]);
    Memo1.Lines.SaveToFile('Ping_log.csv');
    lstReplies.Clear;
    j:=j+1;
//----------------------Graph-----------------------------------
Series1.Clear;
Series2.Clear;
Series3.Clear;
```

```
With Chart1 do
begin
for I:=1 to Stringgrid1.RowCount -1 do
begin
Series1.Add(StrToDateTime(Stringgrid1.cells[0,I]));
Series2.Add(Strtoint(Stringgrid1.cells[1,I]));
Series3.Add(Strtoint(Stringgrid1.cells[2,I]));
end;
end;
//Series1.Active:=True;
Series2.Active:=True;
Series3.Active:=True;
//-------------------------Fin Graph-------------------------------
   end;
// Конструктор
procedure TfrmPing.FormCreate(Sender: TObject);
begin
j:=1;
Combobox1.ItemIndex:=0;
StringGrid1.ColWidths[0]:=105;
StringGrid1.Cells[0,0]:='Date_Time';
StringGrid1.Cells[1,0]:='TTL';
StringGrid1.Cells[2,0]:='TripT.';

StringGrid2.RowHeights[0]:=0;
StringGrid2.ColWidths[1]:=105;
StringGrid2.ColWidths[0]:=62;
StringGrid2.Cells[0,1]:='Min.TTL';
StringGrid2.Cells[0,2]:='Max.TTL';
StringGrid2.Cells[0,3]:='Mean TTL';
StringGrid2.Cells[0,4]:='Min.Trip T.';
StringGrid2.Cells[0,5]:='Max.Trip T.';
StringGrid2.Cells[0,6]:='Mean Trip T.';

minTTL:=256;
maxTTL:=0;
sommeTTL:=0;
//meanTTL:=0;
minTrip:=5000;
maxTrip:=0;
sommeTrip:=0;
meanTrip:=0;
```

```
end;

// Обробка таймерного виклику
procedure TfrmPing.Timer1Timer(Sender: TObject);
begin
  ICMP.OnReply := ICMPReply;
  ICMP.ReceiveTimeout := 5000;
  try
    ICMP.Host := edtHost.Text;
    begin
      ICMP.Ping;
      Application.ProcessMessages;
    end;
  finally btnPing.Enabled := True; end;
end;

procedure TfrmPing.Button1Click(Sender: TObject);
begin
Application.Terminate;
end;

procedure TfrmPing.Chart1ClickSeries(Sender: TCustomChart;
  Series: TChartSeries; ValueIndex: Integer; Button: TMouseButton;
  Shift: TShiftState; X, Y: Integer);
begin
  tmp:=Series2.GetCursorValueIndex;
if tmp<>-1 then
    begin
    Label3.Caption:='Selected Ping:   '+  (StringGrid1.Cells[0,tmp+1])
+'    TTL: '+StringGrid1.Cells[1,tmp+1]
+'    Trip T.: '+StringGrid1.Cells[2,tmp+1];
end;
tmp:=-1;
tmp:=Series3.GetCursorValueIndex;
if tmp<>-1 then
begin
  Label3.Caption:='Selected Ping: '+  (StringGrid1.Cells[0,tmp+1])+
  '   TTL: '+StringGrid1.Cells[1,tmp+1]
  +'   Trip T.: '+StringGrid1.Cells[2,tmp+1];
  tmp:=-1;
end;
end;
```

```
procedure TfrmPing.BitBtn1Click(Sender: TObject);
begin
J:=1;
minTTL:=256;
maxTTL:=0;
meanTTL:=0;
sommeTTL:=0;
minTrip:=5000;
maxTrip:=0;
meanTrip:=0;
sommeTrip:=0;
end;

end.
```

Були проведені розрахунки і підібрані набори тестових даних для перевірки правильності реалізації проектних рішень. Було створено блок-схеми роботи системи. Блок-схеми показують весь процес роботи системи з підсистемами та частково доказують правильність вибраних проектних рішень. Тому від точності і детальної блок-схеми залежить результат всієї програми.

При виборі початкової точки відліку при побудові схем було враховано, що виходячи з вибору мови програмування і інших технічних засобів, програма буде об’єктно-орієнтована що вимагає високого рівня декомпозиції задач на класи.

На рисунку 4.1 зображена основна блок-схема програми, на рисунку 4.2 зображено роботу підпрограми.

З яких видно що робота основної програми складається з початкових етапів ініціалізації ПЗ, перевірки наявності ресурсів системи, блоку початку основного циклу з чеканням запиту від користувача в якому відбувається виклик підсистеми та останньої стадії – перевірка поточного стану з завершенням роботи розробленого ПЗ.

При роботі підпрограми виконується основний функціонал системи з циклічними послідовностями, перевіркою поточного стану та поверненням в основну програму прапорів стану виконання.

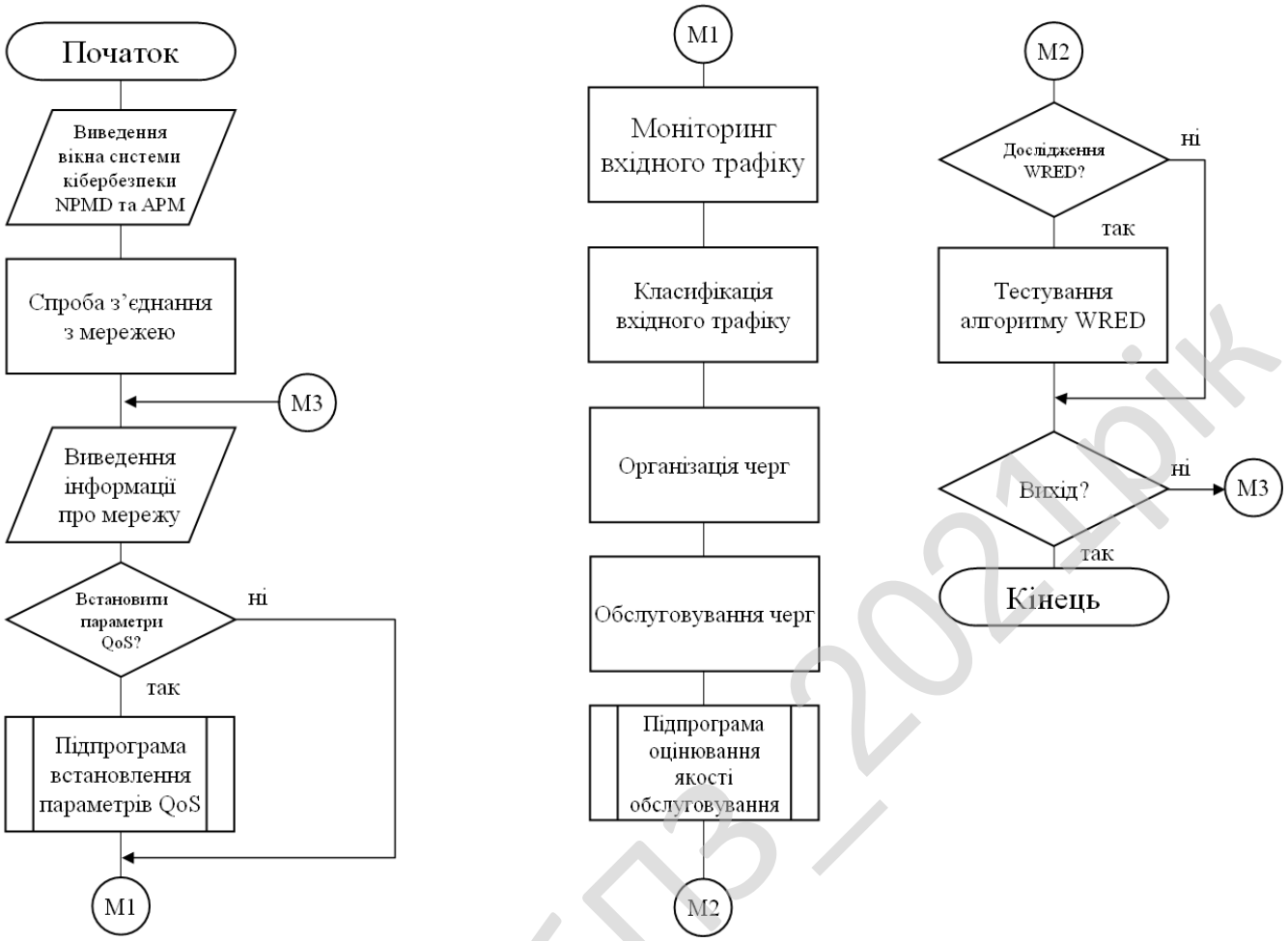

Рисунок 4.1 – Блок-схема основної програми

При розробці ПЗ було використано підходи ри́зик-ме́неджменту – це система управління ризиками, яка включає в себе стратегію та тактику управління, направлені на досягнення основних цілей. Ефективний ризик-менеджмент включає:

– систему управління;

– систему ідентифікації і вимірювання;

– систему супроводження (моніторингу та контролю).

Сучасна наука представляє ризик як вірогідну подію, в результаті настання якої можуть відбутися позитивні, нейтральні або негативні наслідки. Якщо ризик припускає наявність як позитивних, так і негативних результатів, він відноситься

до спекулятивних ризиків. Якщо ж наслідки негативні, або відсутні взагалі, такий ризик іменується чистим.

Рисунок 4.2 – Блок-схема роботи підпрограми

Мета ризик-менеджменту – підвищення конкурентоспроможності господарюючих суб'єктів за допомогою захисту від реалізації чистих ризиків.

Теорія ризик-менеджменту ґрунтується на трьох базових поняттях: корисності, регресії і диверсифікації.

У 1738 швейцарський математик Даніель Бернуллі доповнив теорію вірогідності методом корисності або привабливості того або іншого результату подій. Ідея Бернуллі полягала в тому, що в процесі ухвалення рішення люди приділяють більше уваги розміру наслідків різних результатів, ніж їх вірогідність.

В кінці XIX століття англійський дослідник Ф. Гальтон запропонував вважати регресію або повернення до середнього значення універсальною статистичною закономірністю. Суть регресії трактувалася ним як повернення явищ до норми з часом. Згодом було доведено, що правило регресії діє в найрізноманітніших ситуаціях, починаючи з азартних ігор та розрахунку вірогідності виникнення нещасних випадків, і закінчуючи прогнозуванням коливань економічних циклів.

У 1952 аспірант Університету Чикаго Гарі Марковіц в статті «Диверсифікація вкладень» («Portfolio Selection») математично обґрунтував стратегію диверсифікації інвестиційного портфеля, зокрема, він показав, як шляхом продуманого розподілу вкладень мінімізувати відхилення прибутковості від очікуваного показника. У 1990 Г. Марковіцу присуджена Нобелівська премія за розробку теорії і практики оптимізації портфеля фондових активів.

Етапи ризик-менеджменту

У ризик-менеджменті прийнято виділяти декілька ключових етапів:

– на першому етапі відбувається виявлення ризику з супутньою оцінкою вірогідності його реалізації і масштабу наслідків;

– на другому етапі здійснюється розробка ризик-стратегії з метою зниження вірогідності реалізації ризику і мінімізації можливих негативних наслідків;

– на третьому етапі вибираються методи і інструменти управління виявленим ризиком;

– на четвертому етапі проводиться безпосереднє управління ризиком;

– на завершальному етапі оцінюються досягнуті результати і коректується ризик-стратегія.

За ключовий етап ризик-менеджменту вважається етап вибору методів і інструментів управління ризиком.

Методи і інструментарій ризик-менеджменту

Базовими методами ризик-менеджменту є відмова від ризиків, зниження, передача і ухвалення.

Ризик-інструментарій значно ширший. Він включає політичні, організаційні, правові, економічні, соціальні інструменти, причому ризик-менеджмент як система допускає можливість одночасного застосування декількох методів і інструментів ризик-управління.

Найбільш часто вживаним інструментом ризик-менеджменту є страхування. Страхування припускає передачу відповідальності за відшкодування передбачуваного збитку сторонній організації (страхової компанії).

Прикладами інших інструментів можуть бути відмова від надмірно ризикової діяльності (метод відмови), профілактика або диверсифікація (метод зниження), аутсорсинг витратних ризикових функцій (метод передачі), формування резервів або запасів (метод ухвалення).

Була використана водоспадна (каскадна) модель життєвого циклу ПЗ (waterfall model) – послідовний метод розробки програмного забезпечення, названий так через діаграму схожу на водоспад.

Ця модель розробки запозичена з системної інженерії у виробництві та будівництві – областях, в яких зміни на пізніх етапах дуже дорогі, або неможливі. Наприклад, для створення складних інженерних конструкцій (споруд, літаків, мостів і т.п.). Зміни в проекті фундаменту будинку після того, як покладений дах коштують дуже дорого, тому перфекціонізм на початкових етапах проектування просто необхідний. Інженери, які починали займатись розробкою програмного забезпечення перейшовши з інших галузей, просто адаптували звичну модель, тому що на ранніх етапах розвитку комп'ютерної техніки не було методологій створених саме для програмування. Проте, схожі методології застосовуються для програмного забезпечення й далі, у випадках коли вимоги фіксовані, і вимагається

висока якість та надійність, наприклад в системах для військових чи медичних потреб.

Перший формальний опис водоспадної моделі, після якої вона стала популярною був здійснений В. В. Ройсом у 1970. Попри те, що стаття містить переважно критику методу, на неї часто посилаються.

Переваги методу:

– Ніяких переробок.

– Гарна специфікація перетікає в гарну документацію.

– Зрозуміла модель.

– Розробники можуть мати низьку кваліфікацію.

Недоліки:

– Необхідний перфекціонізм на кожному етапі.

– Важко вносити зміни (якщо взагалі можливо).

– Надлишкове проектування.

– Поділ розробників на "perfect" та "code monkeys".

Модифікації. Через те що цей метод погано підходить для розробки саме ПЗ, частіше використовують його модифікації.

Найвідоміша модифікація – Sashimi. Названа так через японську страву сашимі (суші нарізане і сервіроване так, що складені рядочком шматочки накладаються один на одного). В моделі розробки Сашимі фази життєвого циклу йдуть одна за одною, але при цьому перекриваються одна з одною в часі.

4.2 Захист розробленого програмного забезпечення

Для захисту розробленого програмного забезпечення запропоновано використовувати алгоритм REDOC III, який оперує з 80-бітовим блоком. Довжина ключа може змінюватися й досягати 2560 байт (204800 біт). Алгоритм складається тільки з операцій XOR над байтами ключа й відкритого тексту, перестановки й підстановки не використовуються.

1. Створюють таблицю ключів з 256 10-байтових ключів, використовуючи секретний ключ.

2. Створюють два 10-байтових блоки масок M1 і M2. M1 являє собою результат операції XOR перших 128 10-байтових ключів, а M2 – результат операції XOR других 128 10-байтових ключів.

3. Для шифрування 10-байтового блоку:

a. Виконують операцію XOR з першим байтом блоку даних і першим байтом M1. Вибирають ключ у таблиці ключів, розрахованої в раунді 1. Використовують обчислене значення XOR як індекс таблиці. Виконують операцію XOR з кожним, крім першого, байтом блоку даних і відповідним байтом обраного ключа.

b. Виконують операцію XOR із другим байтом блоку даних і другим байтом M1. Вибирають ключ у таблиці ключів, розрахованої в раунді 1. Використовують обчислене значення XOR як індекс таблиці. Виконаєте операцію XOR з кожним, крім другого, байтом блоку даних і відповідним байтом обраного ключа.

c. Продовжують ці дії з усім блоком даних (з 3-10 байтами), поки не буде використаний кожний байт для вибору ключа з таблиці після виконання операції XOR з ним і відповідним значенням M1. Потім виконують операцію XOR з кожним, крім використаного для вибору ключа, байтом, і ключем.

d. Повторюють етапи a-c для M2.

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:

– Функціональних кнопок ПЗ які викликаються натисканням розділу деревовидної структури.

– Навігаційного меню яке викликається натисканням правої клавіші маніпулятора миші на вікні з наступними підрозділами: Налаштування параметрів моніторингу; Вид; Формат; Журнал роботи; Довідка.

– Верхнього меню: Файл; Моніторинг; Оцінка параметрів; Довідка.

– Розділу виведення результату роботи системи що реалізовано у вигляді відділених вікон що показуються при необхідності.

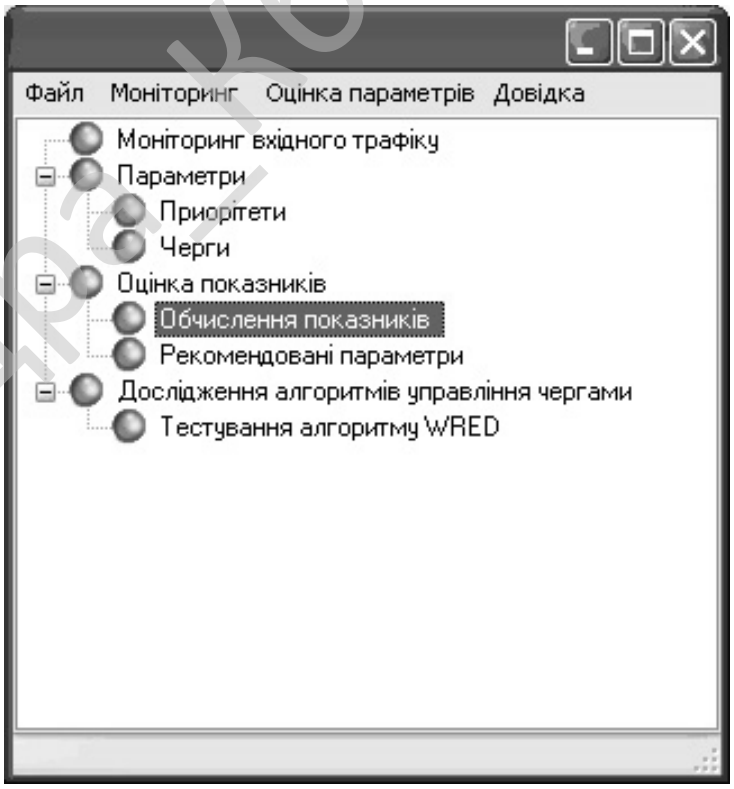

Рисунок 5.1 – Головне вікно ПЗ

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.

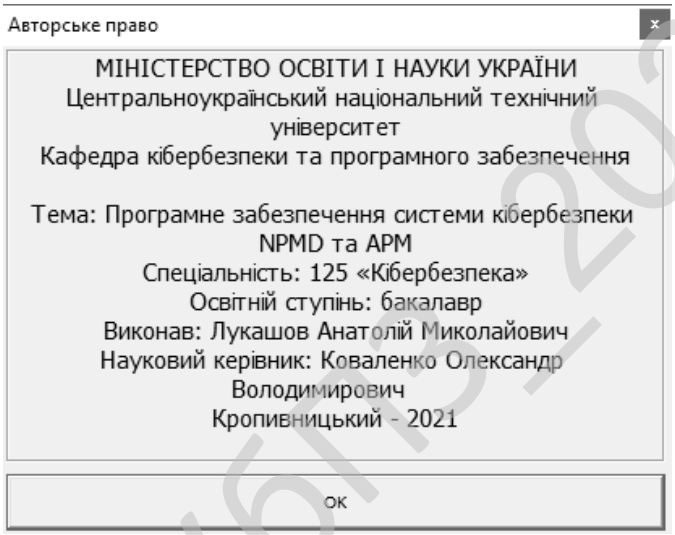

Рисунок 5.2 – Авторське право

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. Таким чином у результаті вищерозглянутого можна стверджувати що розроблено інтерфейс системи у відповідності з вибраною метою роботи. Система містить максимальний необхідний набір функцій придатних для виконання будь-яких дій для забезпечення повноцінної роботи програми. Далі розглянемо висновки та використані літературні джерела.

6 ОСНОВНІ ВИСНОВКИ

Програмне забезпечення, створене в результаті виконання кваліфікаційної бакалаврської роботи, призначено для системи кібербезпеки NPMD та АРМ.

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

Рішення завдання полягало у вирішенні наступних задач:

– Був проведений огляд існуючих систем кібербезпеки NPMD та АРМ.

– Досліджена система кібербезпеки NPMD та АРМ.

– На основі отриманих результатів досліджень створена програмна реалізація системи кібербезпеки NPMD та АРМ.

Розроблені під час виконання кваліфікаційної бакалаврської роботи алгоритми дозволяють успішно вирішувати завдання NPMD та АРМ.

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 RAD Studio Delphi 10. Дана мова програмування дозволяє найбільш ефективно обробляти дані призначені для системи кібербезпеки NPMD та АРМ. 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 XP/Vista/7/8/10.

Даються необхідні рекомендації з установки розробленого програмного забезпечення.

Для підвищення рівня безпеки запропоновано застосовувати алгоритм REDOC III.

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. Вишневский В.М. Теоретические основы проектирования компьютерных сетей / В.М. Вишневский. – М.: Техносфера, 2003. – 512 с.

2. Галкин В.А. Телекоммуникации и сети / В.А. Галкин, Ю.А.Григорьев. – М.: МГТУ имени Н.Э. Баумана, 2003. – 608 с.

3. Гмурман В.Е. Теория вероятностей и математическая статистика / Владимир Ефимович Гмурман. – М.: Высшая школа, 2003. – 479 с.

4. ГОСТ Р ИСО/МЭК 13335-1-2006 Информационная технология Методы и средства обеспечения безопасности Часть 1 Концепция и модели менеджмента безопасности информационных и телекоммуникационных технологий [Електронний ресурс]. – Режим доступу до ресурсу: http://www.rfcmd.ru/sphider/docs/InfoSec/GOST-R_ISO_IEC_13335-1-2006.htm

5. ГОСТ Р ИСО/МЭК 27033-1-2011 Информационная технология. Методы и средства обеспечения безопасности. Безопасность сетей. Часть 1. Обзор и концепции [Електронний ресурс]. – Режим доступу до ресурсу: http://protect.gost.ru/document.aspx?control=7&id=179072

6. ДСТУ 2481 – 94 Системи оброблення інформації інтелектуальні інформаційні технології. Терміни та визначення. – Х.: ДЕРЖСТАНДАРТ УКРАЇНИ, 1994. – 33 с.

7. ДСТУ В 3265 – 95. Зв'язок військовий. Терміни та визначення. – К.:УкрНДІССІ, 1995. – 23 с.

8. ДСТУ ISO 9000:2007 Системи управління якістю. Основні положення та словник термінів [Електронний ресурс]. – Режим доступу до ресурсу: http://document.ua/docs/tdoc14237.php

9. Дымарский Я.С. Управление сетями связи: Принципы, протоколы, прикладные задачи / Я.С. Дымарский, Н.П. Крутякова, Г.Г. Яновский. – М.: ИТЦ «Мобильные коммуникации», 2003. – 384 с.

10. Ершов В.А. Мультисервисные телекоммуникационные сети / В.А. Ершов, Н.А. Кузнецов – М.: Изд. МГТУ им. Н.Э. Баумана, 2003. – 432 с.

11. Зайченко Ю.П. Компьютерные сети / Ю.П. Зайченко. – К.: Слово, 2003. – 256 с.

12. Касперский Е. Компьютерное зловредство / Е. Касперский. – СПб.: Питер, 2007. – 208 с.

13. Касперский К. Техника сетевых атак. [Електронний ресурс]. – Режим доступу до ресурсу: http://rghost.ru/download/43730077/

14. 8e48b6263ce45c7dc2a65a7453383dc33b22486d/Крис%20Касперски%20-%20Техника%20сетевых%20атак.pdf

15. Касперский К. Техника и философия хакерских атак. / К. Касперский. – М.: Солон-Пресс, 2004 – 272 с.

16. Касперский К. Записки исследователя компьютерных вирусов. / К. Касперский. – СПб.: Питер, 2006. – 316 с.

17. Касперский К. Компьютерные вирусы изнутри и снаружи. / К. Касперский. – СПб.: Питер, 2006. – 526 с.

18. Кингман Дж. Пуассоновские процессы / Дж. Кингман М.:МЦНМО, 2007. – 136 с.

19. Клейнрок Л. Вычислительные системы с очередями / Л. Клейнрок – М.:Мир, 1979. – 600 с.

20. Кормен Т. Алгоритмы: построение и анализ / Томас Кормен, Чарльз Лейзерсон, Рональд Ривест, Клиффорд Штайн. – М.:"Вильямс", 2005. –1296 с.

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

22. Королев А.В. Адаптивная маршрутизация в корпоративных сетях / А.В. Королев, Г.А. Кучук, А.А. Пашнев. – Х.: ХВУ, 2003. – 224 с.

23. Кучерявый Е.А. Управление трафиком и качество обслуживания в сети Интернет / Евгений Андреевич Кучерявый. – СПб.: Наука и техника, 2004. – 336 с.

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

25. Лагутин В.С., Степанов С.Н. Телетрафик мультисервисных сетей связи / В.С. Лагутин, С.Н. Степанов. – М.: Радио и связь, 2000. – 320 с.

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

27. Мохамад Гани Абу Таам Разработка математической GERT-модели технологии распространения компьютерных вирусов в информационно-телекоммуникационных сетях / А.А.Смирнов, Мохамад Гани Абу Таам // Информационные системы в управлении, образовании, промышленности: монография / Под редакцией профессора В.С. Пономаренко. – Х.: Вид-во ТОВ «Щедра садиба плюс», 2014. – 498 с.

28. Мохамад Гани Абу Таам Метод управления доступом в интеллектуальных узлах коммутации / Мохамад Гани Абу Таам, А.А.Смирнов // Информационные технологии и защита информации в информационно-коммуникационных системах: монография / Под редакцией профессора В.С. Пономаренко. – Х.: Вид-во ТОВ «Щедра садиба плюс», 2015. – 486 с.

29. Мохамад Гани Абу Таам Математическая GERT-модель технологии передачи метаданных в облачные антивирусные системы / В.В.Босько, А.А.Смирнов, И.А.Березюк, Мохамад Гани Абу Таам // Збірник наукових праць "Системи обробки інформації". – Випуск 1(117). – Х.: ХУПС – 2014. – С. 137-141.

30. Мохамад Гани Абу Таам Структурно-логическая GERT-модель технологии распространения компьютерных вирусов / А.А.Смирнов,

И.А.Березюк, Мохамад Гани Абу Таам // Системи управління, навігації та зв'язку. – Випуск 1(29). – П.: ПНТУ. – 2014. – С. 120-125.

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

32. Мохамад Гани Абу Таам Математическая модель интеллектуального узла коммутации с обслуживанием информационных пакетов различного приоритета / Мохамад Гани Абу Таам, А.А. Смирнов, Н.С. Якименко, С.А. Смирнов // Збірник наукових праць Харківського університету Повітряних Сил. Випуск 4 (41). – Харків: ХУПС. – 2014. – С. 48-52.

33. Мохамад Гани Абу Таам Исследование показателей качества функционирования интеллектуальных узлов коммутации в телекоммуникационных системах и сетях / Мохамад Гани Абу Таам, А.А. Смирнов, Н.С. Якименко, С.А. Смирнов // Наука і техніка Повітряних Сил Збройних Сил України. – Випуск 4(17). – Харків: ХУПС. – 2014. – С.90-95.

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

35. Мохамад Гани Абу Таам Анализ и исследование методов управления сетевыми ресурсами для обеспечения антивирусной защиты данных / Мохамад Гани Абу Таам, А.А. Смирнов, С.А. Смирнов // Системи озброєння і військова техніка. – Випуск 3(43) – Х.: ХУПС – 2015. – С. 100-107.

36. Мохамад Гани Абу Таам Исследование эффективности метода управления доступом к облачным антивирусным телекоммуникационным ресурсам / Мохамад Гани Абу Таам, А.А. Смирнов, С.А. Смирнов // Наука і

техніка Повітряних Сил Збройних Сил України. – Випуск 3(19). – Х.: ХУПС. – 2015. – С. 134-141.

37. Mohamad Abou Taam 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.

38. Мохамад Гани Абу Таам GERT-модель технологии передачи данных в облачные антивирусные системы / А.А. Смирнов, В.В. Босько, Мохамад Гани Абу Таам // Збірник тез доповідей науково-практичної конференції «Застосування інформаційних технологій у підготовці та діяльності сил охорони правопорядку». м. Харків. 12-13 березня 2014 р. – Харків. АВВ МВС. – 2014. – С. 18-19.

39. Мохамад Гани Абу Таам Математическое моделирование технологии передачи сигнатур в облачные антивирусные системы / Мохамад Гани Абу Таам, А.А. Смирнов // Збірник тез VI міжнародної науково-практичної конференції “Проблеми і перспективи розвитку ІТ-індустрії”. м. Харків. 17-18 квітня 2014 р. – Харків: ХНЕУ. – 2014. – С. 260.

40. Мохамад Гани Абу Таам Анализ требований к качеству обслуживания в информационно-телекоммуникационных системах / А.А. Смирнов, Мохамад Гани Абу Таам // Збірник тез XVI міжнародного науково-практичного семінару «Комбінаторні конфігурації та їх застосування». м. Кіровоград. 11-12 квітня 2014 р. – Кіровоград: КНТУ. – 2014. – С. 124-126.

41. Мохамад Гані Абу Таам Дослідження та реалізація GERT-моделі технології розповсюдження комп’ютерних вірусів для захисту телекомунікаційних систем / Мохамад Гані Абу Таам, С.А. Смирнов // Збірник тез науково-практичної конференції «Інформаційні технології та комп’ютерна інженерія». м. Кіровоград. 4 грудня 2014 р. – Кіровоград: КНТУ. – 2014. – С. 168.

42. Мохамад Гани Абу Таам Исследование математических моделей технологии распространения компьютерных вирусов / А.А. Смирнов, Мохамад Гани Абу Таам, С.А. Смирнов // Збірник наукових праць міжнародної науково-практичної конференції «Актуальні питання забезпечення кібернетичної безпеки та захисту інформації». м. Київ. 25-28 лютого 2015 р. – Київ: Європейський університет. – 2015. – С. 90-91.

43. Мохамад Гани Абу Таам Метод управления доступом к «облачным» ресурсам для защиты телекоммуникационных систем / Мохамад Гани Абу Таам, А.А. Смирнов, С.А. Смирнов // Збірник тез всеукраїнської науково-практичної конференції «Інформаційна безпека держави, суспільства та особистості». м. Кіровоград. 16 квітня 2015. – Кіровоград: КНТУ. – 2015. – С. 50-52.

44. Мохамад Гани Абу Таам Разработка метода управления доступом в интеллектуальных узлах коммутации / А.А. Смирнов, Мохамад Гани Абу Таам, С.А. Смирнов // Збірник тез VII міжнародної науково-практичної конференції "Проблеми і перспективи розвитку ІТ-індустрії". м. Харків. 17-18 квітня 2015 р. – Харків: ХНЕУ. – 2015. – С. 14.

45. Мохамад Гани Абу Таам Реализация метода управления доступом в интеллектуальных узлах коммутации / А.А. Смирнов, Мохамад Гани Абу Таам // 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.

46. Мохамад Гани Абу Таам Реализация математической модели интеллектуального узла коммутации для обеспечения защищенности телекоммуникационной сети / Мохамад Гани Абу Таам, А.А. Смирнов, С.А. Смирнов // Збірник тез II Міжнародної науково-практичної Інтернет-конференції «Інформаційна та економічна безпека» (INFECO-2015)». м. Харків. 21-22 травня 2015 р. – Харків: ХІБС УБС НБУ. – 2015. – С. 20-24.

47. Мохамад Гани Абу Таам Разработка математической модели технологии распространения компьютерных вирусов в информационно-телекоммуникационных сетях / Мохамад Гани Абу Таам, 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

48. Мохамад Гани Абу Таам Метод управления доступом к облачным телекоммуникационным ресурсам для обеспечения защиты данных / Мохамад Гани Абу Таам, А.А. Смирнов, С.А. Смирнов // Збірник тез Міжнародної науково-практичної конференції «Комп'ютерні технології та інформаційна безпека». м. Кіровоград. 2-3 липня 2015 р. – Кіровоград: КНТУ. – 2015. – С. 4-5.

49. Мохамад Гани Абу Таам Имитационная модель системы управления доступом к облачным антивирусным телекоммуникационным ресурсам / Мохамад Гани Абу Таам, 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.

50. 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

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

52. 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.

Додаток А
(обов'язковий)

Технічне завдання

Зміст

1 Найменування та область застосування

Це технічне завдання розповсюджується на розробку системи кібербезпеки NPMD та АРМ.

2 Підстава для розробки

Підставою для розробки служить завдання на кваліфікаційну бакалаврську роботу, видане на кафедрі кібербезпеки та програмного забезпечення (нак. № 135-02 від 24.12.2020 року).

3 Мета та призначення розробки

Метою кваліфікаційної бакалаврської роботи є розробка програмного забезпечення системи кібербезпеки NPMD та АРМ.

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 кібербезпеки NPMD та APM;

– 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 RAD Studio Delphi 10.

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 – 61 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 22.05.2021 р.

11.2 Подання кваліфікаційної бакалаврської роботи на захист 14.06.2021 р.

Додаток Б

(обов’язковий)

Міністерство освіти і науки України

Центральноукраїнський національний технічний університет

ЗАТВЕРДЖУЮ

Керівник кваліфікаційної бакалаврської роботи

_____ Коваленко О.В.

Програмне забезпечення системи кібербезпеки NPMD та APM

Лістинг програми

Код документу 12

Носій: CD/DVD-диск

Загальна кількість аркушів: 49

Літера: РП

Кропивницький – 2021 року

Файл Main.pas основної програми

```
unit Main;

interface

// опис бібліотек

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls, ExtCtrls, ComCtrls, Stat,
  ShellAPI, ShlObj, ImgList, TCP_IP, About;

//опис типів

type
  TMainForm = class(TForm)
    gbxShares: TGroupBox;
    lbxShares: TListBox;
    gbxSessions: TGroupBox;
    lvSessions: TListView;
    bvlSessions: TBevel;
    gbxFiles: TGroupBox;
    btnGetShares: TButton;
    btnCloseShares: TButton;
    btnAddShares: TButton;
    btnCloseSession: TButton;
    btnGetSessions: TButton;
    bvlTopSessions: TBevel;
    plButtonFiles: TPanel;
    btnGetFiles: TButton;
    btnCloseFile: TButton;
    bvlLeftFiles: TBevel;
    plFiles: TPanel;
    lvFiles: TListView;
    bvlTopFiles: TBevel;
    gbxTraffic: TGroupBox;
    lvTraffic: TListView;
    bvlTraffic: TBevel;
    tmrTraffic: TTimer;
    Button1: TButton;
    rgScope: TRadioGroup;
    GroupBox1: TGroupBox;
    cbUsageAll: TCheckBox;
    cbUsageConnectable: TCheckBox;
    cbUsageContainer: TCheckBox;
    GroupBox2: TGroupBox;
    cbTypeAny: TCheckBox;
    cbTypeDisk: TCheckBox;
    cbTypePrint: TCheckBox;
    Net_NGN_Tree: TTreeView;
    ImageList1: TImageList;
    Button2: TButton;
    Button3: TButton;
    Button4: TButton;
    function  IsNT(var Value: Boolean): Boolean;
    procedure btnGetSharesClick(Sender: TObject);
    procedure btnCloseSharesClick(Sender: TObject);
    function  SelectDirectory: String;
    procedure btnAddSharesClick(Sender: TObject);
    function  CardinalToTimeStr(Value:Cardinal):String;
    procedure btnGetSessionsClick(Sender: TObject);
    procedure btnCloseSessionClick(Sender: TObject);
    procedure btnGetFilesClick(Sender: TObject);
    procedure btnCloseFileClick(Sender: TObject);
```

```
    procedure tmrTrafficTimer(Sender: TObject);
    procedure Button1Click(Sender: TObject);
    procedure Net_NGN_TreeCustomDrawItem(Sender: TCustomTreeView;
      Node: TTreeNode; State: TCustomDrawState; var DefaultDraw: Boolean);
    procedure Net_NGN_TreeDblClick(Sender: TObject);
    procedure Net_NGN_TreeGetImageIndex(Sender: TObject; Node: TTreeNode);
    procedure Button4Click(Sender: TObject);
    procedure Button2Click(Sender: TObject);
    procedure Button3Click(Sender: TObject);

//опис типів та записів


  private
    { Private declarations }
  public
    { Public declarations }
    SessionCloseKey: array [0..512] of SmallInt;
    procedure Open_Do_Close_Enum(const ParentNode: TTreeNode;
         ResScope, ResType, ResUsage: DWORD; const Net_NGN_ContainerToOpen:
PNet_NGN_Resource);
  //  function OpenEnum(const Net_NGN_ContainerToOpen: PNet_NGN_Resource;
 //        ResScope, ResType, ResUsage: DWORD): THandle;
 //   function EnumResources(const ParentNode: TTreeNode;
  //       ResScope, ResType, ResUsage: DWORD; hNet_NGN_Enum: THandle): UINT;
  end;

type
 TShareInfo2 = packed record
  shi2_net_NGN_name : PWChar;
  shi2_type: DWORD;
  shi2_remark :PWChar;
  shi2_permissions: DWORD;
  shi2_max_uses : DWORD;
  shi2_current_uses : DWORD;
  shi2_path : PWChar;
  shi2_passwd : PWChar;
 end;
 PShareInfo2 = ^ TShareInfo2;
 TShareInfo2Array = array [0..512] of TShareInfo2;
 PShareInfo2Array = ^ TShareInfo2Array;

type
 TShareInfo50 = packed record
  shi50_net_NGN_name : array [0..12] of Char;
  shi50_type : Byte;
  shi50_flags : Word;
  shi50_remark : PChar;
  shi50_path : PChar;
  shi50_rw_password : array [0..8] of Char;
  shi50_ro_password : array [0..8] of Char;
 end;

type
 TSessionInfo502 = packed record
   Sesi502_cname: PWideChar;
   Sesi502_username: PWideChar;
   Sesi502_num_opens: DWORD;
   Sesi502_time: DWORD;
   Sesi502_idle_time: DWORD;
   Sesi502_user_flags: DWORD;
   Sesi502_cltype_name: PWideChar;
   Sesi502_transport: PWideChar;
 End;
 PSessionInfo502 = ^TSessionInfo502;
 TSessionInfo502Array = array[0..512] of TSessionInfo502;
 PSessionInfo502Array = ^TSessionInfo502Array;

type
```

```
 TSessionInfo50 = packed record
   Sesi50_cname       : PChar;
   Sesi50_username    : PChar;
   sesi50_key         : Cardinal;
   sesi50_num_conns   : Word;
   sesi50_num_opens   : Word;
   sesi50_time        : Cardinal;
   sesi50_idle_time   : Cardinal;
   sesi50_protocol    : Byte;
   pad1               : Byte;
 end;

type
 TFileInfo3 = packed record
   fi3_id          : DWORD;
   fi3_permissions : DWORD;
   fi3_num_locks   : DWORD;
   fi3_pathname    : PWChar;
   fi3_username    : PWChar;
 end;
 PFileInfo3 = ^TFileInfo3;
 TFileInfo3Array = array[0..512] of TFileInfo3;
 PFileInfo3Array = ^TFileInfo3Array;

type
 TFileInfo50 = packed record
   fi50_id          : Cardinal;
   fi50_permissions : WORD;
   fi50_num_locks   : WORD;
   fi50_pathname    : PChar;
   fi50_username    : PChar;
   fi50_sharename   : PChar;
 end;

type
  TMibIfRow = packed record
    wszName           : array[0..255] of WideChar;
    dwIndex           : DWORD;
    dwType            : DWORD;
    dwMtu             : DWORD;
    dwSpeed           : DWORD;
    dwPhysAddrLen     : DWORD;
    bPhysAddr         : array[0..7] 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[0..255] of Char;
  end;
  TMibIfArray = array [0..512] of TMibIfRow;
  PMibIfRow = ^TMibIfRow;
  PMibIfArray = ^TMibIfArray;

type
 TMibIfTable = packed record
   dwNumEntries     : DWORD;
   Table            : TMibIfArray;
```

```
 end;
 PMibIfTable = ^TMibIfTable;
var
Net_NGN_ShareEnumNT:function (      servername:PWChar;
                          level:DWORD;
                          bufptr:Pointer;
                          prefmaxlen:DWORD;
                          entriesread,
                          totalentries,
                          resume_handle:LPDWORD): DWORD; stdcall;

var
Net_NGN_ShareEnum:function (  pszServer   : PChar;
                        sLevel            : Cardinal;
                        pbBuffer    : Pchar;
                        cbBuffer    : Cardinal;
                        pcEntriesRead,
                        pcTotalAvail: Pointer):DWORD; stdcall;

var
Net_NGN_ShareDelNT:function (servername: PWideChar;
                        net_NGN_name: PWideChar;
                        reserved: DWORD): LongInt; stdcall;

var
Net_NGN_ShareDel:function (  pszServer,
                        pszNet_NGN_Name:PChar;
                        usReserved:Word): DWORD; stdcall;

var
Net_NGN_ShareAddNT: function(servername: PWideChar;
                        level: DWORD;
                        buf: Pointer;
                        parm_err: LPDWORD): DWORD; stdcall;

var
Net_NGN_ShareAdd: function (  pszServer:Pchar;
                        sLevel:Cardinal;
                        pbBuffer:PChar;
                        cbBuffer:Word):DWORD; stdcall;

Var
Net_NGN_SessionEnumNT:function(servername,
                        UncClientName,
                        username:PWChar;
                        level:DWORD;
                        bufptr:Pointer;
                        prefmaxlen:DWORD;
                        entriesread,
                        totalentries,
                        resume_handle:LPDWORD):DWORD; stdcall;

var
Net_NGN_SessionEnum:function(pszServer:PChar;
                        sLevel: DWORD;
                        pbBuffer:Pointer;
                        cbBuffer:DWORD;
                        pcEntriesRead,
                        pcTotalAvial:Pointer):integer; stdcall;

var
Net_NGN_SessionDelNT:function(ServerName,
                        UncClientName,
                        username:PWChar):DWORD; stdcall;

var
Net_NGN_SessionDel:function( pszServer:PChar;
                        pszClientName: PChar;
                        sReserved: SmallInt):DWORD; stdcall;
```

```
var
Net_NGN_FileEnumNT:function(  servername,
                        basepath,
                        username:PWChar;
                        level:DWORD;
                        bufptr:Pointer;
                        prefmaxlen:DWORD;
                        entriesread,
                        totalentries,
                        resume_handle:LPDWORD):DWORD; stdcall;

var
Net_NGN_FileEnum:function(     pszServer,
                        pszBasePath:PChar;
                        sLevel:DWORD;
                        pbBuffer:Pointer;
                        cbBuffer:DWORD;
                        pcEntriesRead,
                        pcTotalAvail:pointer):integer; stdcall;

var
Net_NGN_FileClose:function(   ServerName:PWideChar;
                        FileId:DWORD):DWORD; stdcall;

var
Net_NGN_FileClose2:function(  pszServer:PChar;
                        ulFileId:LongWord):DWORD; stdcall;

var
GetIfTable:function(    pIfTable    : PMibIfTable;
                        pdwSize     : PULONG;
                        bOrder      : Boolean ): DWORD; stdcall;

var
  MainForm: TMainForm;

implementation

{$R *.dfm}

{ TMainForm }

//////////////////////////
//
//  Спочатку нам потрібно визначиться, під якою системою ми працюємо,
// щоб довідатися яку частину коду (для NT чи ні) використовувати в цей момент.
// Для цього напишемо невелику функцію, що і буде визначати тип системи.
//

function TMainForm.IsNT(var Value: Boolean): Boolean;
var Ver: TOSVersionInfo;
    BRes: Boolean;
begin
  Ver.dwOSVersionInfoSize := SizeOf(TOSVersionInfo);
  BRes := GetVersionEx(Ver);
  if not BRes then   //Перевірка
  begin
    Result := False; //Інформація не отримана
    Exit;            //ідемо
  end else
    Result := True;  //Інформація отримана

  case Ver.dwPlatformId of //визначаємося
    VER_PLATFORM_WIN32_NT       : Value := True;  //Windows NT тьа вище -
підходить
    VER_PLATFORM_WIN32_WINDOWS  : Value := False; //Windows 9 x-Me- підходить
    VER_PLATFORM_WIN32s         : Result := False //Windows 3.x- не підходить
  end;
```

```
end;

////////////////////////////
//
//  Одержання всіх відкритих загальних ресурсів досліджуємої мережі
//

procedure TMainForm.btnGetSharesClick(Sender: TObject);
var
  i:Integer;
  FLibHandle : THandle;
  ShareNT : PShareInfo2Array;     //<= Змінні
  entriesread,totalentries:DWORD; //<= для Windows NT
  Share : array [0..512] of TShareInfo50; //<= Змінні
  pcEntriesRead,pcTotalAvail:Word;        //<= для Windows 9 x-Me
  OS: Boolean;
begin
  lbxShares.Items.Clear;
  if not IsNT(OS) then Close; //Визначаємо тип системи

  if OS then begin  //Код для NT
    FLibHandle := LoadLibrary(' NET_NGN_API32.DLL' ); //Завантажуємо бібліотеку
    if FLibHandle = 0 then Exit;
    //Зв' язуємо функцію
    @Net_NGN_ShareEnumNT := GetProcAddress(FLibHandle,' Net_NGN_ShareEnum' );
    if not Assigned(Net_NGN_ShareEnumNT) then //Перевірка
    begin
      FreeLibrary(FLibHandle);
      Exit;
    end;
    ShareNT := nil; //Очищаємо покажчик на масив структур
    //Виклик функції
    if Net_NGN_ShareEnumNT(nil,2,@ShareNT,DWORD(-1),
                      @entriesread,@totalentries,nil) <> 0 then
    begin  //Якщо виклик невдалий вивантажуємо бібліотеку
      FreeLibrary(FLibHandle);
      Exit;
    end;
    if entriesread > 0 then //Обробка результатів
    for i:= 0 to entriesread- 1 do
      lbxShares.Items.Add(String(ShareNT^[i].shi2_net_NGN_name));
  end else begin //Код для 9 x-me
    FLibHandle := LoadLibrary(' SVRAPI.DLL' ); //Завантажуємо бібліотеку
    if FLibHandle = 0 then Exit;
    //Зв' язуємо функцію
    @Net_NGN_ShareEnum := GetProcAddress(FLibHandle,' Net_NGN_ShareEnum' );
    if not Assigned(Net_NGN_ShareEnum) then //Перевірка
    begin
      FreeLibrary(FLibHandle);
      Exit;
    end;
    //Виклик функції
    if Net_NGN_ShareEnum(nil,50,@Share,SizeOf(Share),
                  @pcEntriesRead,@pcTotalAvail)<> 0 then
    begin  //Якщо виклик невдалий вивантажуємо бібліотеку
      FreeLibrary(FLibHandle);
      Exit;
    end;
    if pcEntriesRead > 0 then //Обробка результатів
    for i:= 0 to pcEntriesRead- 1 do
      lbxShares.Items.Add(String(Share[i].shi50_net_NGN_name));
  end;
  FreeLibrary(FLibHandle); //Не забуваємо вивантажити бібліотеку
end;

////////////////////////////
//
//  Закриття загального ресурсу
//
```

```
procedure TMainForm.btnCloseSharesClick(Sender: TObject);
var
  OS:Boolean;
  FLibHandle : THandle;
  Name9x:array [0..12] of Char;
  NameNT:PWChar;
  i:Integer;
  ShareName: String;
begin
  if not IsNT(OS) then Close; //Визначаємо тип системи

  if lbxShares.Items.Count = 0 then Exit;
  for i:= 0 to lbxShares.Items.Count-1 do
    if lbxShares.Selected[i] then Break;  //Шукаємо обраний елемент
  if not lbxShares.Selected[i] then Exit; //Якщо не знайдений ідемо
  ShareName := lbxShares.Items.Strings[i];

  if OS then begin //Код для NT
    FLibHandle := LoadLibrary(' NET_NGN_API32.DLL' );
    if FLibHandle = 0 then Exit;
    @Net_NGN_ShareDelNT := GetProcAddress(FLibHandle,' Net_NGN_ShareDel' );
    if not Assigned(Net_NGN_ShareDelNT) then //Перевірка
    begin
      FreeLibrary(FLibHandle);
      Exit;
    end;
    i:= SizeOf(WideChar)*256;
    GetMem(NameNT,i);  //Виділяємо пам' ять під змінну
    StringToWideChar(ShareName,NameNT,i); //Перетворимо в PWideChar
    Net_NGN_ShareDelNT(nil,NameNT,0);   //Видаляємо ресурс
    FreeMem(NameNT);  //Звільняємо пам' ять
  end else begin  //Код для 9 х-ме
    FLibHandle := LoadLibrary(' SVRAPI.DLL' );
    if FLibHandle = 0 then Exit;
    @Net_NGN_ShareDel := GetProcAddress(FLibHandle,' Net_NGN_ShareDel' );
    if not Assigned(Net_NGN_ShareDel) then //Перевірка
    begin
      FreeLibrary(FLibHandle);
      Exit;
    end;
    FillChar(Name9x, SizeOf(Name9x), #0); //Очищаємо масив
    move(ShareName[1],Name9x[0],Length(ShareName)); //Заповнюємо масив
    Net_NGN_ShareDel(nil,@Name9x,0); //Видаляємо ресурс
  end;
  FreeLibrary(FLibHandle);
end;

//////////////////////////
//
//  Показ діалогу вибору директорії
//

function TMainForm.SelectDirectory: String;
var
  lpItemID : PItemIDList;
  BrowseInfo : TBrowseInfo;
  DisplayName : array[0..MAX_PATH] of Char;
  TempPath : array[0..MAX_PATH] of Char;
begin
  FillChar(BrowseInfo, sizeof(TBrowseInfo), #0);
  BrowseInfo.hwndOwner := Handle;
  BrowseInfo.pszDisplayName := @DisplayName;
  BrowseInfo.lpszTitle := ' Specify a directory' ;
  BrowseInfo.ulFlags := BIF_RETURNONLYFSDIRS;
  lpItemID := SHBrowseForFolder(BrowseInfo);
  if Assigned(lpItemId) then begin
    SHGetPathFromIDList(lpItemID, TempPath);
```

```
    GlobalFreePtr(lpItemID);
  end else Result := ' ' ;
  Result := String(TempPath);
end;

////////////////////////////
//
//  Додавання загального ресурсу
//

procedure TMainForm.btnAddSharesClick(Sender: TObject);
const
  STYPE_DISKTREE = 0;
  ACCESS_ALL =  258;
  SHI50F_FULL = 258;
var
  FLibHandle : THandle;
  Share9x : TShareInfo50;
  ShareNT : TShareInfo2;
  TmpDir, TmpName: String;
  TmpDirNT, TmpNameNT: PWChar;
  OS: Boolean;
  TmpLength: Integer;
begin
  TmpDir := SelectDirectory; //Визначаємо шлях до наступного ресурсу
  TmpName := InputBox(' Share name' ,' Enter name' ,' Test' ); //Визначаємо ім'
я під яким він буде видний у мережі
  if TmpDir = ' '  then Exit;

  if not IsNT(OS) then Close; //З' ясовуємо тип системи

  if OS then begin //Код для NT
    FLibHandle := LoadLibrary(' NET_NGN_API32.DLL' );
    if FLibHandle = 0 then Exit;
    @Net_NGN_ShareAddNT := GetProcAddress(FLibHandle,' Net_NGN_ShareAdd' );
    if not Assigned(Net_NGN_ShareAddNT) then
    begin
      FreeLibrary(FLibHandle);
      Exit;
    end;
    TmpLength := SizeOF(WideChar)*256; //Визначаємо необхідний розмір

    GetMem(TmpNameNT, TmpLength); //Конвертуємо в PWChar
    StringToWideChar(TmpName, TmpNameNT, TmpLength);
    ShareNT.shi2_net_NGN_name := TmpNameNT; //Ім' я

    ShareNT.shi2_type := STYPE_DISKTREE; //Тип ресурсу
    ShareNT.shi2_remark := ' ' ; //Коментар
    ShareNT.shi2_permissions := ACCESS_ALL; //Доступ
    ShareNT.shi2_max_uses := DWORD(-1); // Кількість максим. підключ.
    ShareNT.shi2_current_uses := 0; // Кількість тік підкл.

    GetMem(TmpDirNT, TmpLength);
    StringToWideChar(TmpDir, TmpDirNT, TmpLength);
    ShareNT.shi2_path := TmpDirNT; //Шлях до ресурсу

    ShareNT.shi2_passwd := ' ' ; //Пароль

    Net_NGN_ShareAddNT(nil,2,@ShareNT, nil); //Додаємо ресурс
    FreeMem (TmpNameNT); //звільняємо пам' ять
    FreeMem (TmpDirNT);
  end else begin
    FLibHandle := LoadLibrary(' SVRAPI.DLL' );
    if FLibHandle = 0 then Exit;
    @Net_NGN_ShareAdd := GetProcAddress(FLibHandle,' Net_NGN_ShareAdd' );
    if not Assigned(Net_NGN_ShareAdd) then
    begin
      FreeLibrary(FLibHandle);
      Exit;
```

```
    end;
    FillChar(Share9x.shi50_net_NGN_name, SizeOf(Share9x.shi50_net_NGN_name),
#0);
    move(TmpName[1],Share9x.shi50_net_NGN_name[0],Length(TmpName)); //Ім' я
    Share9x.shi50_type := STYPE_DISKTREE; //Тип ресурсу
    Share9x.shi50_flags := SHI50F_FULL; //Доступ
    FillChar(Share9x.shi50_remark,
      SizeOf(Share9x.shi50_remark), #0); //Коментар
    FillChar(Share9x.shi50_path,
      SizeOf(Share9x.shi50_path), #0);
    Share9x.shi50_path := PAnsiChar(TmpDir); //Шлях до ресурсу
    FillChar(Share9x.shi50_rw_password,
      SizeOf(Share9x.shi50_rw_password), #0); //Пароль повного доступу
    FillChar(Share9x.shi50_ro_password,
      SizeOf(Share9x.shi50_ro_password), #0); //Пароль для читання
    Net_NGN_ShareAdd(nil,50,@Share9x,SizeOf(Share9x));
  end;
  FreeLibrary(FLibHandle);
end;

////////////////////////////
//
//  Помітьте що активний і неактивний час сесій буде даватися нам
//  у вигляді кіл-ті секунд (тип Cardinal). Напишемо невелику
//  функцію, задача якої буде перетворювати кількість секунд у більше
//  звичну форму відображення.
//

function TMainForm.CardinalToTimeStr(Value: Cardinal): String;
var d,h,m,s: Real;
begin
    d:=0;
    h:=0;
    m:=0;
    s:=Value;
    if s > 59 then begin
        m:=int(s / 60);
        s:= s-s-(m*60);
    end;
    if m > 59 then begin
        h:=int(m/60);
        m:= m-m-(h*60);
    end;
    if h > 23 then begin
        d:=int(h/24);
        h:= h-h-(d*24);
    end;
    Result:=' ' ;
    if (d>0) then Result:=Result+floattostr(d)+'  d. ' ;
    if (h<9) then Result:=Result+' 0' +floattostr(h)+' :'  else
Result:=Result+floattostr(h)+' :' ;
    if (m<9) then Result:=Result+' 0' +floattostr(m)+' :'  else
Result:=Result+floattostr(m)+' :' ;
    if (s<9) then Result:=Result+' 0' +floattostr(s)     else
Result:=Result+floattostr(s);
end;

////////////////////////////
//
//  Одержання списку сесій досліджуємої мережі системи кібербезпеки NPMD та APM
//

procedure TMainForm.btnGetSessionsClick(Sender: TObject);
var
  OS: Boolean;
  FLibHandle : THandle;
  SessionInfo50: array [0..512] of TSessionInfo50;
  SessionInfo502 : PSessionInfo502Array;
  TotalEntries,EntriesReadNT: DWORD;
```

```
  EntriesRead,TotalAvial: Word;
  i:integer;
begin
  lvSessions.Items.Clear;

  if not IsNT(OS) then Close; //З’ ясовуємо тип системи

  if OS then begin  //Код для NT
    FLibHandle := LoadLibrary(‘ NET_NGN_API32.DLL’ );
    if FLibHandle = 0 then Exit;
    @Net_NGN_SessionEnumNT := GetProcAddress(FLibHandle, ‘ Net_NGN_SessionEnum’
);
    if not Assigned(Net_NGN_SessionEnumNT) then
    begin
      FreeLibrary(FLibHandle);
      Exit;
    end;
    SessionInfo502 := nil;
    if Net_NGN_SessionEnumNT(nil,nil,nil,502,@SessionInfo502,DWORD(-
1),@entriesreadNT, @totalentries, nil)=0 then
    for i:=0 to EntriesReadNT-1 do
    begin
      with lvSessions.Items.Add do  //Заповнення даними зі структури
      begin
        Caption := string(SessionInfo502^[i].sesi502_cname); //Ім’ я комп’ ютера
        SubItems.Add(SessionInfo502^[i].sesi502_username);   //Ім’ я користувача
        SubItems.Add(IntToStr(SessionInfo502^[i].sesi502_num_opens));
//Відкритих ресурсів
        SubItems.Add(CardinalToTimeStr(SessionInfo502^[i].Sesi502_Time)); //Час
активний
        SubItems.Add(CardinalToTimeStr(SessionInfo502^[i].sesi502_idle_time));
//Час не активний
      end;
    end;
  end else begin //Код для Windows 9 x-Me
   FLibHandle := LoadLibrary(‘ SVRAPI.DLL’ );
   if FLibHandle = 0 then Exit;
   @Net_NGN_SessionEnum := GetProcAddress(FLibHandle, ‘ Net_NGN_SessionEnum’ );
    if not Assigned(Net_NGN_SessionEnum) then
    begin
      FreeLibrary(FLibHandle);
      Exit;
    end;
   if Net_NGN_SessionEnum
(nil,50,@SessionInfo50,SizeOf(SessionInfo50),@EntriesRead,@TotalAvial) = 0 then
   for i:=0 to EntriesRead-1 do
   begin
      with lvSessions.Items.Add do  //Заповнення даними зі структури
      begin
        Caption := string(SessionInfo50[i].Sesi50_cname); //Ім’ я комп’ ютера
досліджуємої мережі системи кібербезпеки NPMD та АРМ
        SubItems.Add(SessionInfo50[i].Sesi50_username);   //Ім’ я користувача
        SubItems.Add(IntToStr(SessionInfo50[i].sesi50_num_opens)); //Відкритих
ресурсів
        SubItems.Add(CardinalToTimeStr(SessionInfo50[i].Sesi50_Time)); //Час
активний
        SubItems.Add(CardinalToTimeStr(SessionInfo50[i].sesi50_idle_time));
//Час не активний
        SessionCloseKey[i]:= SessionInfo50[i].sesi50_key; //Унікальний
ідентифікатор для закриття
      end;
   end;
 end;
 FreeLibrary(FLibHandle);
end;

////////////////////////////
//
//  Завершення обраної сесії
```

```
//

procedure TMainForm.btnCloseSessionClick(Sender: TObject);
var
  OS: Boolean;
  FLibHandle : THandle;
  CNameNT: PWideChar;
  CName9x: PAnsiChar;
  Key:SmallInt;
  i: Integer;
begin
  if not IsNT(OS) then Close; //З' ясовуємо тип системи

  if not Assigned(lvSessions.Selected) then Exit;
  i:= lvSessions.Selected.Index; //Визначаємо номер обраної сесії

  if OS then begin
    FLibHandle := LoadLibrary(' NET_NGN_API32.DLL' );
    if FLibHandle = 0 then Exit;
    @Net_NGN_SessionDelNT := GetProcAddress(FLibHandle, ' Net_NGN_SessionDel' );
    if not Assigned(Net_NGN_SessionDelNT) then
    begin
      FreeLibrary(FLibHandle);
      Exit;
    end;
    //Перетворимо дані в необхідний вид
    CNameNT := PWChar(WideString(' \\' +lvSessions.Items.Item[i].Caption));
    Net_NGN_SessionDelNT(nil,CNameNT,nil);
  end else begin
    FLibHandle := LoadLibrary(' SVRAPI.DLL' );
    if FLibHandle = 0 then Exit;
    @Net_NGN_SessionDel := GetProcAddress(FLibHandle, ' Net_NGN_SessionDel' );
    if not Assigned(Net_NGN_SessionDel) then
    begin
      FreeLibrary(FLibHandle);
      Exit;
    end;
    //Перетворимо дані в необхідний вид
    CName9x := PAnsiChar(lvSessions.Items.Item[i].Caption);
    key := SessionCloseKey[i]; //Беремо ключ із масиву
    Net_NGN_SessionDel(nil,CName9x,Key);
  end;
  FreeLibrary(FLibHandle);
end;

//////////////////////////
//
//  Одержання списку відкритих файлів досліджуємої мережі системи кібербезпеки
NPMD та APM
//

procedure TMainForm.btnGetFilesClick(Sender: TObject);
var
  OS: Boolean;
  FLibHandle : THandle;
  FileInfoNT: PFileInfo3Array;
  FileInfo9x: array [0..512] of TFileInfo50;
  TotalEntries,EntriesReadNT: DWORD;
  EntriesRead,TotalAvial: Word;
  i:integer;
begin
  lvfiles.Items.Clear;

  if not IsNT(OS) then Close; //З' ясовуємо тип системи

  if OS then begin  //Код для NT
    FLibHandle := LoadLibrary(' NET_NGN_API32.DLL' );
    if FLibHandle = 0 then Exit;
    @Net_NGN_FileEnumNT := GetProcAddress(FLibHandle, ' Net_NGN_FileEnum' );
```

```
    if not Assigned(Net_NGN_FileEnumNT) then
    begin
      FreeLibrary(FLibHandle);
      Exit;
    end;
    FileInfoNT := nil;
    if Net_NGN_FileEnumNT(nil,nil,nil,3,@FileInfoNT,DWORD(-1),@entriesreadNT,
@totalentries, nil)=0 then
    for i:=0 to EntriesReadNT-1 do
    begin
      with lvFiles.Items.Add do  //Заповнення даними зі структури
      begin
        Caption := string(IntToStr(FileInfoNT^[i].fi3_id)); //Ідентифікатор
        SubItems.Add(FileInfoNT^[i].fi3_pathname);   //Шлях до файлу
        SubItems.Add(FileInfoNT^[i].fi3_username);   //Ім' я користувача
      end;
    end;
  end else begin //Код для Windows 9 x-Me
   FLibHandle := LoadLibrary(' SVRAPI.DLL' );
   if FLibHandle = 0 then Exit;
   @Net_NGN_FileEnum := GetProcAddress(FLibHandle, ' Net_NGN_FileEnum' );
    if not Assigned(Net_NGN_FileEnum) then
    begin
      FreeLibrary(FLibHandle);
      Exit;
    end;
   if Net_NGN_FileEnum (nil,
nil,50,@FileInfo9x,SizeOf(FileInfo9x),@EntriesRead,@TotalAvial)= 0 then
   for i:=0 to EntriesRead-1 do
   begin
      with lvFiles.Items.Add do  //Заповнення даними зі структури
      begin
        Caption := string(IntToStr(FileInfo9x[i].fi50_id)); //Ідентифікатор
        SubItems.Add(FileInfo9x[i].fi50_pathname); //Шлях до файлу
        SubItems.Add(FileInfo9x[i].fi50_username); //Ім' я користувача
      end;
   end;
 end;
 FreeLibrary(FLibHandle);
end;

//////////////////////////
//
//  Закриття файлу
//

procedure TMainForm.btnCloseFileClick(Sender: TObject);
var
  OS: Boolean;
  FLibHandle : THandle;
  i: Integer;
begin
  if not IsNT(OS) then Close; //З' ясовуємо тип системи

  if not Assigned(lvFiles.Selected) then Exit;
  i:= lvFiles.Selected.Index; //Визначаємо номер обраного файлу

  if OS then begin //Код для NT
    FLibHandle := LoadLibrary(' NET_NGN_API32.DLL' );
    if FLibHandle = 0 then Exit;
    @Net_NGN_FileClose := GetProcAddress(FLibHandle, ' Net_NGN_FileClose' );
    if not Assigned(Net_NGN_FileClose) then
    begin
      FreeLibrary(FLibHandle);
      Close;
    end;
    Net_NGN_FileClose(nil,StrToInt(lvFiles.Items.Item[i].Caption));//Закриваємо
файл
  end else begin //Код для Windows 9 x-Me
```

```
    FLibHandle := LoadLibrary(‘ SVRAPI.DLL’ );
    if FLibHandle = 0 then Exit;
    @Net_NGN_FileClose2 := GetProcAddress(FLibHandle, ‘ Net_NGN_FileClose2’ );
    if not Assigned(Net_NGN_FileClose2) then
    begin
      FreeLibrary(FLibHandle);
      Close;
    end;
    Net_NGN_FileClose2(nil,StrToInt(lvFiles.Items.Item[i].Caption));
  end;
  FreeLibrary(FLibHandle);
end;

////////////////////////////
//
//  Визначаємо вхідний / вихідний трафік досліджуємої мережі системи
кібербезпеки NPMD та APM
//

procedure TMainForm.tmrTrafficTimer(Sender: TObject);
  // Допоміжна функція, що перетворить MAC адресу до "нормального" виду
  //Визначаємо спеціальний тип, щоб можна було передати у функцію масив
  type TMAC = array [0..7] of Byte;
  //Як перше значення масив, друге значення, розмір даних у масиві
  function GetMAC(Value: TMAC; Length: DWORD): String;
  var
    i: Integer;
  begin
    if Length = 0 then Result := ‘  00-00-00’  else
    begin
      Result := ‘ ‘ ;
      for i:= 0 to Length-2 do
        Result := Result + IntToHex(Value[i],2)+’ -’ ;
      Result := Result + IntToHex(Value[ Length-1],2);
    end;
  end;

//Сама процедура
var
  FLibHandle : THandle;
  Table: TMibIfTable;
  i : integer;
  Size   : integer;
begin
  tmrTraffic.Enabled := false; //Припиняємо таймер
  lvTraffic.Items.BeginUpdate;
  lvTraffic.Items.Clear;  //Очищаємо список
  FLibHandle := LoadLibrary(‘ IPHLPAPI.DLL’ ); //Завантажуємо бібліотеку
  if FLibHandle = 0 then Exit;
  @GetIfTable := GetProcAddress(FLibHandle, ‘ GetIfTable’ );
  if not Assigned(GetIfTable) then
  begin
    FreeLibrary(FLibHandle);
    Close;
  end;

  Size := SizeOf(Table);
  if GetIfTable(@Table, @Size, false ) = 0 then //Виконуємо функцію
    for i:= 0 to Table.dwNumEntries-1 do begin
      with lvTraffic.Items.Add do begin //Виводимо результати
        Caption := String(Table.Table[i].bDescr); //Найменування інтерфейсу
        SubItems.Add(GetMAC(TMAC(Table.Table[i].bPhysAddr),
          Table.Table[i].dwPhysAddrLen)); //MAC адреса
        SubItems.Add(IntToStr(Table.Table[i].dwInOctets)); //Усього прийнято
байт з досліджуємої мережі системи кібербезпеки NPMD та APM
        SubItems.Add(IntToStr(Table.Table[i].dwOutOctets)); //Усього відправлено
байт у досліджуєму мережу системи кібербезпеки NPMD та APM
      end;
    end;
```

```
  lvTraffic.Items.EndUpdate;
  FreeLibrary(FLibHandle);
  tmrTraffic.Enabled := true; //Не забуваємо активувати таймер
end;


 function OpenEnum(const Net_NGN_ContainerToOpen: PNet_NGN_Resource; ResScope,
ResType, ResUsage: DWORD): THandle;
var
 hNet_NGN_Enum: THandle;
begin
 Result:=0;
 if (NO_ERROR<>WNet_NGN_OpenEnum(ResScope, ResType, ResUsage,
                                 Net_NGN_ContainerToOpen, hNet_NGN_Enum))
 then ShowMessage(‘ Помилка!’ )
 else Result:=hNet_NGN_Enum;
end;

function EnumResources(const ParentNode: TTreeNode;
ResScope, ResType, ResUsage: DWORD; hNet_NGN_Enum: THandle): UINT;
 function ShowResource(const ParentNode: TTreeNode; Res: TNet_NGN_Resource):
TTreeNode;
  begin
   Result:=MainForm.Net_NGN_Tree.Items.AddChild(ParentNode,
string(Res.lpRemoteName));
  end;

const
 RESOURCE_BUF_ENTRIES = 2000;

var
 ResourceBuffer: array[1..RESOURCE_BUF_ENTRIES] of TNet_NGN_Resource;
 i, ResourceBuf, EntriesToGet: dword;
 NewNode: TTreeNode;
begin
 Result:=0;
 while true do
  begin
   ResourceBuf:=sizeof(ResourceBuffer);
   EntriesToGet:=RESOURCE_BUF_ENTRIES;
   if (NO_ERROR<>WNet_NGN_EnumResource(hNet_NGN_Enum, EntriesToGet,
                           @ResourceBuffer, ResourceBuf))
   then
    begin
     case GetLastError() of
      NO_ERROR:  // проход буферу без перемикання
      Break;
      ERROR_NO_MORE_ITEMS:
         // Повертає 0 у тому випадку, коли останов
         // RESOURCE_BUF_ENTRIES дані на попередньому виклику, щоб
         // WNet_NGN_EnumResource, та були точно
         // RESOURCE_BUF_ENTRIES дані в запису на момент
         // попереднього виклику
      Exit;
      else ShowMessage(Помилка!’ );
      Result:=1;
      Exit;
     end;
    end;
   for i:=1 to EntriesToGet do
    begin
     NewNode:=ShowResource(ParentNode, ResourceBuffer[i]);
     if (ResourceBuffer[i].dwUsage and RESOURCEUSAGE_CONTAINER)<>0
     then MainForm.Open_Do_Close_Enum(NewNode, ResScope, ResType, ResUsage,
@ResourceBuffer[i]);
     Application.ProcessMessages;
    end;
  end;
end;
```

```
procedure TMainForm.Open_Do_Close_Enum(const ParentNode: TTreeNode; ResScope,
ResType, ResUsage: DWORD; const Net_NGN_ContainerToOpen: PNet_NGN_Resource);
var
 hNet_NGN_Enum: THandle;
begin
 hNet_NGN_Enum:=OpenEnum(Net_NGN_ContainerToOpen, ResScope, ResType, ResUsage);
 if (hNet_NGN_Enum=0)
 then Exit;
 EnumResources(ParentNode, ResScope, ResType, ResUsage, hNet_NGN_Enum);
 if (NO_ERROR<>WNet_NGN_CloseEnum(hNet_NGN_Enum))
 then ShowMessage(‘ WNet_NGN_CloseEnum Помилка’ );
end;

procedure TMainForm.Button1Click(Sender: TObject);
var
 ResScope, ResType, ResUsage: dword;
begin
 Button1.Caption:=‘ Пошук мережних ресурсів. Чекайте...’ ;
 Button1.Enabled:=false;
 //
 Net_NGN_Tree.Items.Clear;
 case rgScope.ItemIndex of
  1: ResScope:=RESOURCE_GLOBALNET_NGN_;
  2: ResScope:=RESOURCE_REMEMBERED;
  else ResScope:=RESOURCE_CONNECTED;
 end;
 ResType:=0;
 if cbTypeAny.Checked
 then ResType:=ResType or RESOURCETYPE_ANY;
 if cbTypeDisk.Checked
 then ResType:=ResType or RESOURCETYPE_DISK;
 if cbTypePrint.Checked
 then ResType:=ResType or RESOURCETYPE_PRINT;
 ResUsage:=0;
 if cbUsageConnectable.Checked
 then ResUsage:=ResUsage or RESOURCEUSAGE_CONNECTABLE;
 if cbUsageContainer.Checked
 then ResUsage:=ResUsage or RESOURCEUSAGE_CONTAINER;
 Open_Do_Close_Enum(Net_NGN_Tree.Items.Add(nil, ‘ Net_NGN_work Resources’ ),
                              ResScope, ResType, ResUsage, nil);
 //
 Button1.Caption:=‘ Обновити список ресурсів’ ;
 Button1.Enabled:=true;

end;

procedure TMainForm.Net_NGN_TreeCustomDrawItem(Sender: TCustomTreeView;
  Node: TTreeNode; State: TCustomDrawState; var DefaultDraw: Boolean);
begin
if cdsSelected in State
 then Sender.Canvas.Font.Style:=Sender.Canvas.Font.Style+[fsUnderline];
end;

procedure TMainForm.Net_NGN_TreeDblClick(Sender: TObject);
begin
ShellExecute(0,’ open’ ,PChar(Net_NGN_Tree.Selected.Text),’ ‘ ,’ ‘ ,SW_SHOW);
end;

procedure TMainForm.Net_NGN_TreeGetImageIndex(Sender: TObject; Node: TTreeNode);
begin
 if Node.HasChildren
 then Node.ImageIndex:=1
 else Node.ImageIndex:=0;
end;

procedure TMainForm.Button4Click(Sender: TObject);
begin
Form1.Show;
```

```
end;

procedure TMainForm.Button2Click(Sender: TObject);
begin
Form2.Show;
end;

procedure TMainForm.Button3Click(Sender: TObject);
begin
Form3.Show;
end;

end.
```

Основна програма

Файл NPMD_APM.dpr основної програми

```
program NPMD_APM;

uses
  Forms,
  Main in ' Main.pas'  {MainForm},
  About in ' About.pas'  {Form1},
  TCP_IP in ' TCP_IP.pas'  {Form2},
  Stat in ' Stat.pas'  {Form3};

{$R *.res}

begin
  Application.Initialize;
  Application.CreateForm(TMainForm, MainForm);
  Application.CreateForm(TForm1, Form1);
  Application.CreateForm(TForm2, Form2);
  Application.CreateForm(TForm3, Form3);
  Application.Run;
end.
```

Файл IPHLPAPI.pas- обробка API функцій

```
unit IPHLPAPI;

interface
uses
  Windows, winsock;

const
  VERSION       = ' 1.5' ;

//------- Заголовок з Microsoft IPTYPES.H-H-----------------

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

 // Вузлові типи ( NET_NGN_BIOS)
  BROADCAST_NODETYPE = 1;
  PEER_TO_PEER_NODETYPE = 2;
  MIXED_NODETYPE = 4;
  HYBRID_NODETYPE = 8;

  NET_NGN_BIOSTypes  : array[0..8] of string[20] =
    ( ' Невизначений' , ' Передача' , ' Рівень до рівня' , ' ' , '
Змішаний' , ' ' , ' ' , ' ' , ' Гібрід'
      );

 // Типи адаптеру
{ v1.4-> 1.5
  IF_OTHER_ADAPTERTYPE = 0;
  IF_ETHERNET_NGN__ADAPTERTYPE = 1;
  IF_TOKEN_RING_ADAPTERTYPE = 2;
  IF_FDDI_ADAPTERTYPE = 3;
  IF_PPP_ADAPTERTYPE = 4;
  IF_LOOPBACK_ADAPTERTYPE = 5;
  IF_SLIP_ADAPTERTYPE = 6;

Знайдено у ipifcons.h :
#define MIB_IF_TYPE_OTHER               1
#define MIB_IF_TYPE_ETHERNET_NGN_            6
#define MIB_IF_TYPE_TOKENRING           9
#define MIB_IF_TYPE_FDDI                15
#define MIB_IF_TYPE_PPP                 23
#define MIB_IF_TYPE_LOOPBACK            24
#define MIB_IF_TYPE_SLIP                28
}
  IF_OTHER_ADAPTERTYPE = 1;
  IF_ETHERNET_NGN__ADAPTERTYPE = 6;
  IF_TOKEN_RING_ADAPTERTYPE = 9;
  IF_FDDI_ADAPTERTYPE = 15;
  IF_PPP_ADAPTERTYPE = 23;
  IF_LOOPBACK_ADAPTERTYPE = 24;
  IF_SLIP_ADAPTERTYPE = 28;

//  AdaptTypes    : array[0..6] of string[10] =
//    ( ' інший' , ' ethernet_NGN_' , ' tokenring' , ' FDDI' , ' PPP' , '
loopback' , ' SLIP'  );
   AdaptTypes    : array[1..28] of string[10] =
```

```
          ( ' інший' , ' ' , ' ' , ' ' , ' ' , ' ethernet_NGN_' , ' ' , ' ' , '
tokenring' ,
           ' ' , ' ' , ' ' , ' ' , ' ' , ' FDDI' , ' ' , ' ' , ' ' , ' ' , ' ' ,
' ' , ' ' , ' PPP' ,
           ' loopback' , ' ' , ' ' , ' ' , ' SLIP'  );
      // Кінець змін в типі адаптерів

      //-------для інших MS заготовочних файлів-------------------

        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;

      //---IP адресні структури-------------------------

        PTIP_ADDRESS_STRING = ^TIP_ADDRESS_STRING;
        TIP_ADDRESS_STRING = array[0..15] of char; //  IP рядок
        //
        PTIP_ADDR_STRING = ^TIP_ADDR_STRING;
        TIP_ADDR_STRING = packed record // для використання у зв' язних списках
          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[1..MAX_HOSTNAME_LEN + 4] of char;    // дані
          DomainName: array[1..MAX_DOMAIN_NAME_LEN + 4] of char;   // дані
          CurrentDNSServer: PTIP_ADDR_STRING;
          DNSServerList: TIP_ADDR_STRING;
          NodeType: UINT;
          ScopeID: array[1..MAX_SCOPE_ID_LEN + 4] of char;   // дані
          EnableRouting: UINT;
          EnableProxy: UINT;
          EnableDNS: UINT;
        end;

      //-----структура мережного інтерфейсу досліджуємої мережі системи
кібербезпеки NPMD та АРМ-----------------------

      ////////////////////////
      //
//
      // Наступне є діючими станами для WAN да LAN інтерфейсів. //
      // Порядок станів створений для визначення. Для      //
      // стану >= CONNECTED можливо передавати дані зразу. Стан >= DISCONNECTED
//
      // може передавати деякі дані. Стан < DISCONNECTED може   //
      // не передавати дані.
//
      // карта з поміткою UNREACHABLE якщо DIM викликає InterfaceUnreachable для
//
      // причин. Крім невдачі з'єднання.                                  //
      //
//
      // NON_OPERATIONAL- Перевірка для LAN інтерфейсу. Позначає карту що не
працює       //
      //                       або не з'єднується з картою.        //
      // UNREACHABLE- Перевірка WAN інтерфейсів . Позначає, що віддалений сайт
//
```

```
      //                      не з'єднується за потрібний час.
//
      // DISCONNECTED- Перевірка WAN інтерфейсів . Позначає, що віддалений сайт
//
      //                      не з'єднується.
      //
      // CONNECTING- Перевірка WAN інтерфейсів . Означає спробу з'єднання     //
      //                      з сайтом, якого немає.              //
      // CONNECTED- Перевірка WAN інтерфейсів . Позначає, що віддалений сайт
//
      //                      з'єднується.
//
      // OPERATIONAL- Перевірка LAN Interfaces. Позначає карту підключену //
      //                      в праці.                                    //
      //
//
      // Усі дії користувачів записуються до  MIB-II значення    //
      // можуть бути використовані                                        //
      //
//
      ////////////////////////

      const
      // дані додані до ipifcons.h
        IF_OPER_STATUS_NON_OPERATIONAL = 0 ;
        IF_OPER_STATUS_UNREACHABLE = 1 ;
        IF_OPER_STATUS_DISCONNECTED = 2 ;
        IF_OPER_STATUS_CONNECTING = 3 ;
        IF_OPER_STATUS_CONNECTED = 4 ;
        IF_OPER_STATUS_OPERATIONAL = 5 ;

        MIB_IF_TYPE_OTHER = 1 ;
        MIB_IF_TYPE_ETHERNET_NGN_ = 6 ;
        MIB_IF_TYPE_TOKENRING = 9 ;
        MIB_IF_TYPE_FDDI = 15 ;
        MIB_IF_TYPE_PPP = 23 ;
        MIB_IF_TYPE_LOOPBACK = 24 ;
        MIB_IF_TYPE_SLIP = 28 ;

        MIB_IF_ADMIN_STATUS_UP = 1 ;
        MIB_IF_ADMIN_STATUS_DOWN = 2 ;
        MIB_IF_ADMIN_STATUS_TESTING = 3 ;

        MIB_IF_OPER_STATUS_NON_OPERATIONAL = 0 ;
        MIB_IF_OPER_STATUS_UNREACHABLE = 1 ;
        MIB_IF_OPER_STATUS_DISCONNECTED = 2 ;
        MIB_IF_OPER_STATUS_CONNECTING = 3 ;
        MIB_IF_OPER_STATUS_CONNECTED = 4 ;
        MIB_IF_OPER_STATUS_OPERATIONAL = 5 ;

      type
        PTMibIfRow = ^TMibIfRow;
        TMibIfRow = packed record
          wszName: array[1..MAX_INTERFACE_NAME_LEN] of WCHAR;
          dwIndex: DWORD;
          dwType: DWORD;       // дивись MIB_IF_TYPE
          dwMTU: DWORD;
          dwSpeed: DWORD;
          dwPhysAddrLen: DWORD;
          bPhysAddr: array[1..MAXLEN_PHYSADDR] of byte;
          dwAdminStatus: DWORD;    // дивись MIB_IF_ADMIN_STATUS
          dwOperStatus: DWORD;     // дивись MIB_IF_OPER_STATUS
          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;

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

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

        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;       //
дані
          Description: array[1..MAX_ADAPTER_DESCRIPTION_LENGTH + 4] of char;
// дані
          AddressLength: UINT;
          Address: array[1..MAX_ADAPTER_ADDRESS_LENGTH] of byte;      // дані
          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: LongInt ; // UNIX час, секунди з 1970
          LeaseExpires: LongInt;   // UNIX час, секунди з 1970
          SpareStuff: array [1..200] of char ;   // дані- простір для списку IP
адрес досліджуємої мережі системи кібербезпеки NPMD та APM
        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 ------------------------

 //
  PTMibIPNet_NGN_Row = ^TMibIPNet_NGN_Row;
  TMibIPNet_NGN_Row = packed record
    dwIndex: DWord;
    dwPhysAddrLen: DWord;
    bPhysAddr: TMACAddress;
    dwAddr: DWord;
    dwType: DWord;
  end;
  //
  PTMibIPNet_NGN_Table = ^TMibIPNet_NGN_Table;
  TMibIPNet_NGN_Table = packed record
    dwNumEntries: DWORD;
    Table: array[0..ANY_SIZE- 1] of TMibIPNet_NGN_Row;
  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;
  //
  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;

    //---------імпорт до IPHLPAPI.DLL------------------

    var

    GetAdaptersInfo: function ( pAdapterInfo: PTIP_ADAPTER_INFO;
      pOutBufLen: PULONG ): DWORD; stdcall;

    GetNet_NGN_workParams: function ( FixedInfo: PTFixedInfo; pOutPutLen:
PULONG ):
            DWORD; stdcall;

    GetTcpTable: function ( pTCPTable: PTMibTCPTable; pDWSize: PDWORD;
      bOrder: BOOL ): DWORD; stdcall;

    GetTcpStatistics: function ( pStats: PTMibTCPStats ): DWORD; stdcall;

    GetUdpTable: function ( pUdpTable: PTMibUDPTable; pDWSize: PDWORD;
     bOrder: BOOL ): DWORD; stdcall;

    GetUdpStatistics: function ( pStats: PTMibUdpStats ): DWORD; stdcall;

    GetIpStatistics: function ( pStats: PTMibIPStats ): DWORD; stdcall;

    GetIpNet_NGN_Table: function ( pIpNet_NGN_Table: PTMibIPNet_NGN_Table;
      pdwSize: PULONG;  bOrder: BOOL ): DWORD; stdcall;

    GetIpAddrTable: function ( pIpAddrTable: PTMibIPAddrTable;
      pdwSize: PULONG; bOrder: BOOL ): DWORD; stdcall;

    GetIpForwardTable: function ( pIPForwardTable: PTMibIPForwardTable;
      pdwSize: PULONG; bOrder: BOOL ): DWORD; stdCall;

    GetIcmpStatistics: function ( pStats: PTMibICMPInfo ): DWORD; stdCall;

    GetRTTAndHopCount: function ( DestIPAddress: DWORD; HopCount: PULONG;
      MaxHops: ULONG; RTT: PULONG ): BOOL; stdCall;

    GetIfTable: function ( pIfTable: PTMibIfTable; pdwSize: PULONG;
      bOrder: boolean ): DWORD; stdCall;

    GetIfEntry: function ( pIfRow: PTMibIfRow ): DWORD; stdCall;

    // попередження – недокументована функція, можливі помилки при
використанні
    GetFriendlyIfIndex: function (var IfIndex: DWORD): DWORD; stdcall;


    const
        IpHlpDLL = ' IPHLPAPI.DLL' ;
    var
        IpHlpModule: THandle;

        function LoadIpHlp: Boolean;

    implementation

    function LoadIpHlp: Boolean;
    begin
        Result := True;
        if IpHlpModule <> 0 then Exit;
```

```
      // відкрити DLL
          IpHlpModule := LoadLibrary (IpHlpDLL);
          if IpHlpModule = 0 then
          begin
              Result := false;
              exit ;
          end ;
          GetAdaptersInfo := GetProcAddress (IpHlpModule, ' GetAdaptersInfo' ) ;
          GetNet_NGN_workParams := GetProcAddress (IpHlpModule, '
GetNet_NGN_workParams' ) ;
          GetTcpTable := GetProcAddress (IpHlpModule, ' GetTcpTable' ) ;
          GetTcpStatistics := GetProcAddress (IpHlpModule, ' GetTcpStatistics' )
;
          GetUdpTable := GetProcAddress (IpHlpModule, ' GetUdpTable' ) ;
          GetUdpStatistics := GetProcAddress (IpHlpModule, ' GetUdpStatistics' )
;
          GetIpStatistics := GetProcAddress (IpHlpModule, ' GetIpStatistics' ) ;
          GetIpNet_NGN_Table := GetProcAddress (IpHlpModule, '
GetIpNet_NGN_Table' ) ;
          GetIpAddrTable := GetProcAddress (IpHlpModule, ' GetIpAddrTable' ) ;
          GetIpForwardTable := GetProcAddress (IpHlpModule, ' GetIpForwardTable'
) ;
          GetIcmpStatistics := GetProcAddress (IpHlpModule, ' GetIcmpStatistics'
) ;
          GetRTTAndHopCount := GetProcAddress (IpHlpModule, ' GetRTTAndHopCount'
) ;
          GetIfTable := GetProcAddress (IpHlpModule, ' GetIfTable' ) ;
          GetIfEntry := GetProcAddress (IpHlpModule, ' GetIfEntry' ) ;
          GetFriendlyIfIndex := GetProcAddress (IpHlpModule, '
GetFriendlyIfIndex' ) ;
      end;

      initialization
          IpHlpModule := 0 ;
      finalization
          if IpHlpModule <> 0 then
          begin
              FreeLibrary (IpHlpModule) ;
              IpHlpModule := 0 ;
          end ;

      end.
```

Файл IPHelper.pas- функції роботи з IP-протоколом

```
unit IPHelper;


interface

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

const
  NULL_IP       = '   0.0.  0.0' ;

//---перетворення добре відомих номерів портів до імен сервісів--------

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


const
    // тільки найбільш популярні сервіси...
  WellKnownPorts: array[1..32] of TWellKnownPort
  = (
//    ( Prt: 0; Srv:   ' RESRVED'  ),     {Зарезервовано}
    ( Prt: 7; Srv:   ' ECHO   '  ),     {Ping    }
    ( Prt: 9; Srv:   ' DISCARD'  ),
    ( Prt: 13; Srv:  ' DAYTIME'  ),
    ( Prt: 17; Srv:  ' QOTD   '  ),     {Показчик на день}
    ( Prt: 19; Srv:  ' CHARGEN'  ),     {Генератор символів}
    ( Prt: 20; Srv:  ' FTPDATA'  ),     { File Transfer Protocol- дані}
    ( Prt: 21; Srv:  ' FTPCTRL'  ),     { File Transfer Protocol- управління}
    ( Prt: 22; Srv:  ' SSH    '  ),
    ( Prt: 23; Srv:  ' TELNET_NGN_ '  ),
    ( Prt: 25; Srv:  ' SMTP   '  ),     { Simple Mail Transfer Protocol}
    ( Prt: 37; Srv:  ' TIME   '  ),     { Часовий протокол }
    ( Prt: 43; Srv:  ' WHOIS  '  ),     { Сервіс – Кто це  }
    ( Prt: 53; Srv:  ' DNS    '  ),     { Domain Name Service }
    ( Prt: 67; Srv:  ' BOOTPS '  ),     { BOOTP Сервер }
    ( Prt: 68; Srv:  ' BOOTPC '  ),     { BOOTP Кієнт }
    ( Prt: 69; Srv:  ' TFTP   '  ),     { стандартний  FTP  }
    ( Prt: 70; Srv:  ' GOPHER '  ),     { Протокол Gopher       }
    ( Prt: 79; Srv:  ' FINGER '  ),     { Протокол Finger       }
    ( Prt: 80; Srv:  ' HTTP   '  ),     { Протокол HTTP         }
    ( Prt: 88; Srv:  ' KERBROS'  ),     { Протокол Kerberos     }
    ( Prt: 109; Srv: ' POP2   '  ),     { Протокол Post Office Protocol Version
2 }
    ( Prt: 110; Srv: ' POP3   '  ),     { Протокол Post Office Protocol Version
3 }
    ( Prt: 111; Srv: ' SUN_RPC'  ),     { Протокол SUN Remote Procedure Call }
    ( Prt: 119; Srv: ' NNTP   '  ),     { Протокол Net_NGN_work News Transfer
Protocol }
    ( Prt: 123; Srv: ' NTP    '  ),     { Протокол Net_NGN_work Time protocol
}
    ( Prt: 135; Srv: ' DCOMRPC'  ),     { Протокол Location Service
}
    ( Prt: 137; Srv: ' NBNAME '  ),     { NET_NGN_BIOS сервіс імен          }
    ( Prt: 138; Srv: ' NBDGRAM'  ),     { NET_NGN_BIOS сервіс датаграм     }
    ( Prt: 139; Srv: ' NBSESS '  ),     { NET_NGN_BIOS сервіс сесій        }
    ( Prt: 143; Srv: ' IMAP   '  ),     { Протокол Internet_NGN_ Message Access
Protocol }
    ( Prt: 161; Srv: ' SNMP   '  ),     { Протокол Simple Net_NGN_w. Management
Protocol }
    ( Prt: 169; Srv: ' SEND   '  )
```

```
    );


//------перетворення ICMP кодів помилок до рядків-------------

const
  ICMP_ERROR_BASE = 11000;
  IcmpErr : array[1..22] of string =
  (
   ' IP_BUFFER_TOO_SMALL' ,' IP_DEST_NET_NGN__UNREACHABLE' , '
IP_DEST_HOST_UNREACHABLE' ,
   ' IP_PROTOCOL_UNREACHABLE' , ' IP_DEST_PORT_UNREACHABLE' , ' IP_NO_RESOURCES'
,
   ' IP_BAD_OPTION' ,' IP_HARDWARE_ПОМИЛКА' , ' 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 перерахованих величин у рядки--------

  ARPEntryType  : array[1..4] of string = ( ' інший' , ' неправильний' ,
    ' динамічний' , ' статичний'
    );
  TCPConnState  :
    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 =
    ( ' Const.Timeout' , '  MIL-STD-1778' ,
    ' Van Jacobson' , ' інший'  );

  IPForwTypes   : array[1..4] of string =
    ( ' інший' , ' invalid' , ' local' , ' remote'  );

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

type

// для IpHlpNet_NGN_workParams
  TNet_NGN_workParams = record
    HostName: string ;
    DomainName: string ;
    CurrentDnsServer: string ;
    DnsServerTot: integer ;
    DnsServerNames: array [0..9] of string ;
    NodeType: UINT;
    ScopeID: string ;
    EnableRouting: UINT;
    EnableProxy: UINT;
    EnableDNS: UINT;
  end;

  TIfRows = array of TMibIfRow ; // динамічний масив колонок

// для IpHlpAdaptersInfo
  TAdaptorInfo = record
    AdapterName: string ;
```

```
    Description: string ;
    MacAddress: string ;
    Index: DWORD;
    aType: UINT;
    DHCPEnabled: UINT;
    CurrIPAddress: string ;
    CurrIPMask: string ;
    IPAddressTot: integer ;
    IPAddressList: array of string ;
    IPMaskList: array of string ;
    GatewayTot: integer ;
    GatewayList: array of string ;
    DHCPTot: integer ;
    DHCPServer: array of string ;
    HaveWINS: BOOL;
    PrimWINSTot: integer ;
    PrimWINSServer: array of string ;
    SecWINSTot: integer ;
    SecWINSServer: array of string ;
    LeaseObtained: LongInt ; // UNIX час, секунди з 1970
    LeaseExpires: LongInt;   // UNIX час, секунди з 1970
  end ;

  TAdaptorRows = array of TAdaptorInfo ;


//--------експортуємі дані---------------------

function IpHlpAdaptersInfo(var AdpTot: integer;var AdpRows: TAdaptorRows):
integer ;
procedure Get_AdaptersInfo( List: TStrings );
function IpHlpNet_NGN_workParams (var Net_NGN_workParams: TNet_NGN_workParams):
integer ;
procedure Get_Net_NGN_workParams( List: TStrings );
procedure Get_ARPTable( List: TStrings );
procedure Get_TCPTable( List: TStrings );
procedure Get_TCPStatistics( List: TStrings );
function IpHlpTCPStatistics (var TCPStats: TMibTCPStats): integer ;
procedure Get_UDPTable( List: TStrings );
procedure Get_UDPStatistics( List: TStrings );
function IpHlpUdpStatistics (UdpStats: TMibUDPStats): integer ;
procedure Get_IPAddrTable( List: TStrings );
procedure Get_IPForwardTable( List: TStrings );
procedure Get_IPStatistics( List: TStrings );
function IpHlpIPStatistics (var IPStats: TMibIPStats): integer ;
function Get_RTTAndHopCount( IPAddr: DWORD; MaxHops: Longint;
  var RTT: longint; var HopCount: longint ): integer;
procedure Get_ICMPStats( ICMPIn, ICMPOut: TStrings );
function IpHlpIfTable(var IfTot: integer; var IfRows: TIfRows): integer ;
procedure Get_IfTable( List: TStrings );
function IpHlpIfEntry(Index: integer; var IfRow: TMibIfRow): integer ;
procedure Get_RecentDestIPs( List: TStrings );

// функції 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 функції----------------------

{ отримання наступного "токена" з рядка }
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;

//-----------------------------------
{ перетворення числового MAC-адреса  до ww-xx-yy-zz рядка }
function MacAddr2Str( MacAddr: TMacAddress; size: integer ): string;
var
  i             : integer;
begin
  if Size = 0 then
  begin
    Result := '  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 }
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 в 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 номеру порту в мережний байт типу DWORD }
function Port2Wrd( nwoPort: DWORD ): DWORD;
begin
  Result := Swap( WORD( nwoPort ) );
end;

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

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

//-----------------------------------
{ голона частина, фіксація мережний параметрів досліджуємої мережі для оцінки
якості обслуговування (NPMD_APM)}

procedure Get_Net_NGN_workParams( List: TStrings );
var
    Net_NGN_workParams: TNet_NGN_workParams ;
    I, ErrorCode: integer ;
begin
    if not Assigned( List ) then EXIT;
    List.Clear;
    ErrorCode := IpHlpNet_NGN_workParams (Net_NGN_workParams) ;
    if ErrorCode <> 0 then
    begin
        List.Add (SysErrorMessage (ErrorCode));
        exit;
    end ;
    with Net_NGN_workParams do
    begin
        List.Add( ' Ім'я хосту          : '  + HostName );
        List.Add( ' Домен            : '  + DomainName );
        List.Add( ' NET_NGN_BIOS тип : '  + NET_NGN_BIOSTypes[NodeType] );
        List.Add( ' DHCP область         : '  + ScopeID );
        List.Add( ' ROUTING визначено    : '  + IntToStr( EnableRouting ) );
        List.Add( ' PROXY   визначено    : '  + IntToStr( EnableProxy ) );
        List.Add( ' DNS     визначено    : '  + IntToStr( EnableDNS ) );
        if DnsServerTot <> 0 then
        begin
            for I := 0 to Pred (DnsServerTot) do
                List.Add( ' DNS адреса серверу   : '  + DnsServerNames [I] ) ;
        end ;
    end ;
end ;

//-------------------------------------//
function IpHlpNet_NGN_workParams (var Net_NGN_workParams: TNet_NGN_workParams):
integer ;
```

```
var
  FixedInfo     : PTFixedInfo;         // дані
  InfoSize      : Longint;
  PDnsServer    : PTIP_ADDR_STRING ;   // дані
begin
    InfoSize := 0 ;   // дані
    result := ERROR_NOT_SUPPORTED ;
    if NOT LoadIpHlp then exit ;
    result := GetNet_NGN_workParams( Nil, @InfoSize );  // дані
    if result <> ERROR_BUFFER_OVERFLOW then exit ; // дані
    GetMem (FixedInfo, InfoSize) ;                    // дані
    try
    result := GetNet_NGN_workParams( FixedInfo, @InfoSize );   // дані
    if result <> ERROR_SUCCESS then exit ;
    Net_NGN_workParams.DnsServerTot := 0 ;
    with FixedInfo^ do
    begin
        Net_NGN_workParams.HostName := trim (HostName) ;
        Net_NGN_workParams.DomainName := trim (DomainName) ;
        Net_NGN_workParams.ScopeId := trim (ScopeID) ;
        Net_NGN_workParams.NodeType := NodeType ;
        Net_NGN_workParams.EnableRouting := EnableRouting ;
        Net_NGN_workParams.EnableProxy := EnableProxy ;
        Net_NGN_workParams.EnableDNS := EnableDNS ;
        Net_NGN_workParams.DnsServerNames [0] := DNSServerList.IPAddress ;  //
дані
        if Net_NGN_workParams.DnsServerNames [0] <> ' ' then
                                    Net_NGN_workParams.DnsServerTot := 1 ;
        PDnsServer := DnsServerList.Next;
        while PDnsServer <> Nil do
        begin
            Net_NGN_workParams.DnsServerNames [Net_NGN_workParams.DnsServerTot]
:=
                                    PDnsServer^.IPAddress ;  // дані
            inc (Net_NGN_workParams.DnsServerTot) ;
            if Net_NGN_workParams.DnsServerTot >=
                          Length (Net_NGN_workParams.DnsServerNames) then exit
;
            PDnsServer := PDnsServer.Next ;
        end;
    end ;
    finally
       FreeMem (FixedInfo) ;                    // дані
    end ;
end;

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

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


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

// включення байтів у/з для кожного адаптера

function IpHlpIfTable(var IfTot: integer; var IfRows: TIfRows): integer ;
var
  I,
  TableSize   : integer;
  pBuf, pNext : PChar;
begin
  result := ERROR_NOT_SUPPORTED ;
  if NOT LoadIpHlp then exit ;
```

```
  SetLength (IfRows, 0) ;
  IfTot := 0 ; // дані
  TableSize := 0;
   // перший виклик: необхідно отримати розмір пам' яті
  result := GetIfTable (Nil, @TableSize, false) ;  // дані
  if result <> ERROR_INSUFFICIENT_BUFFER then exit ;
  GetMem( pBuf, TableSize );
  try
      FillChar (pBuf^, TableSize, #0);  // очищаємо буфер з  W98 не беремо
крапку таблиці
      result := GetIfTable (PTMibIfTable (pBuf), @TableSize, false) ;
      if result <> NO_ERROR then exit ;
      IfTot := PTMibIfTable (pBuf)^.dwNumEntries ;
      if IfTot = 0 then exit ;
      SetLength (IfRows, IfTot) ;
      pNext := pBuf + SizeOf(IfTot) ;
      for i := 0 to Pred (IfTot) do
      begin
         IfRows [i] := PTMibIfRow (pNext )^ ;
         inc (pNext, SizeOf (TMibIfRow)) ;
      end;
  finally
      FreeMem (pBuf) ;
  end ;
end;

procedure Get_IfTable( List: TStrings );
var
  IfRows        : TIfRows ;
  Error, I      : integer;
  NumEntries    : integer;
  sDescr, sIfName: string ;
begin
  if not Assigned( List ) then EXIT;
  List.Clear;
  SetLength (IfRows, 0) ;
  Error := IpHlpIfTable (NumEntries, IfRows) ;
  if (Error <> 0) then
      List.Add( SysErrorMessage( GetLastError ) )
  else if NumEntries = 0 then
      List.Add( ' данних немає '  )
  else
  begin
      for I := 0 to Pred (NumEntries) do
      begin
          with IfRows [I] do
          begin
             if wszName [1] = #0 then
                 sIfName := ' '
             else
                 sIfName := WideCharToString (@wszName) ;  // конвертуємо Юнікод
до рядка
            sIfName := trim (sIfName) ;
            sDescr := bDescr ;
            sDescr := trim (sDescr);
            List.Add (Format (
              ' %0.8x |%3d | %16s |%8d |%12d |%2d |%2d |%10d |%10d | %-s| %-s'
,
              [dwIndex, dwType, MacAddr2Str( TMacAddress( bPhysAddr ),
              dwPhysAddrLen ), dwMTU, dwSpeed, dwAdminStatus,
              dwOPerStatus, Int64 (dwInOctets), Int64 (dwOutOctets),  //
конвертуємо до 32-біт
              sIfName, sDescr] )  // дані, додані в/з
              );
          end;
      end ;
  end ;
  SetLength (IfRows, 0) ;  // вільна пам' ять
end ;
```

```
function IpHlpIfEntry(Index: integer; var IfRow: TMibIfRow): integer ;
begin
  result := ERROR_NOT_SUPPORTED ;
  if NOT LoadIpHlp then exit ;
  FillChar (IfRow, SizeOf (TMibIfRow), #0);  // очищаємо буфер з  W98 не беремо
  IfRow.dwIndex := Index ;
  result := GetIfEntry (@IfRow) ;
end ;

//-----------------------------------
{ інформація про інстальовані адаптери }

function IpHlpAdaptersInfo(var AdpTot: integer; var AdpRows: TAdaptorRows):
integer ;
var
  BufLen        : DWORD;
  AdapterInfo   : PTIP_ADAPTER_INFO;
  PIpAddr       : PTIP_ADDR_STRING;
  PBuf          : PCHAR ;
  I             : integer ;
begin
  SetLength (AdpRows, 4) ;
  AdpTot := 0 ;
  BufLen := 0 ;
  result := GetAdaptersInfo( Nil, @BufLen );
  if (result <> ERROR_INSUFFICIENT_BUFFER) and (result = NO_ERROR) then exit ;
  GetMem( pBuf, BufLen );
  try
      FillChar (pBuf^, BufLen, #0);  // очищуємо буфер
      result := GetAdaptersInfo( PTIP_ADAPTER_INFO (PBuf), @BufLen );
      if result = NO_ERROR then
      begin
         AdapterInfo := PTIP_ADAPTER_INFO (PBuf) ;
         while ( AdapterInfo <> nil ) do
         begin
            AdpRows [AdpTot].IPAddressTot := 0 ;
            SetLength (AdpRows [AdpTot].IPAddressList, 2) ;
            SetLength (AdpRows [AdpTot].IPMaskList, 2) ;
            AdpRows [AdpTot].GatewayTot := 0 ;
            SetLength (AdpRows [AdpTot].GatewayList, 2) ;
            AdpRows [AdpTot].DHCPTot := 0 ;
            SetLength (AdpRows [AdpTot].DHCPServer, 2) ;
            AdpRows [AdpTot].PrimWINSTot := 0 ;
            SetLength (AdpRows [AdpTot].PrimWINSServer, 2) ;
            AdpRows [AdpTot].SecWINSTot := 0 ;
            SetLength (AdpRows [AdpTot].SecWINSServer, 2) ;
            AdpRows [AdpTot].CurrIPAddress := NULL_IP;
            AdpRows [AdpTot].CurrIPMask := NULL_IP;
            AdpRows [AdpTot].AdapterName := Trim( string(
AdapterInfo^.AdapterName ) );
            AdpRows [AdpTot].Description := Trim( string(
AdapterInfo^.Description ) );
            AdpRows [AdpTot].MacAddress := MacAddr2Str( TMacAddress(
                                  AdapterInfo^.Address ),
AdapterInfo^.AddressLength ) ;
            AdpRows [AdpTot].Index := AdapterInfo^.Index ;
            AdpRows [AdpTot].aType := AdapterInfo^.aType ;
            AdpRows [AdpTot].DHCPEnabled := AdapterInfo^.DHCPEnabled ;
            if AdapterInfo^.CurrentIPAddress <> Nil then
            begin
                AdpRows [AdpTot].CurrIPAddress :=
AdapterInfo^.CurrentIPAddress.IpAddress ;
                AdpRows [AdpTot].CurrIPMask :=
AdapterInfo^.CurrentIPAddress.IpMask ;
            end ;

        // беремо список IP адрес та конвертуємо в IPAddressList
            I := 0 ;
```

```
        PIpAddr := @AdapterInfo^.IPAddressList ;
        while (PIpAddr <> Nil) do
        begin
            AdpRows [AdpTot].IPAddressList [I] := PIpAddr.IpAddress ;
            AdpRows [AdpTot].IPMaskList [I] := PIpAddr.IpMask ;
            PIpAddr := PIpAddr.Next ;
            inc (I) ;
            if Length (AdpRows [AdpTot].IPAddressList) <= I then
            begin
                 SetLength (AdpRows [AdpTot].IPAddressList, I -2) ;
                 SetLength (AdpRows [AdpTot].IPMaskList, I -2) ;
            end ;
        end ;
        AdpRows [AdpTot].IPAddressTot := I ;

    // беремо список IP адрес для GatewayList
        I := 0 ;
        PIpAddr := @AdapterInfo^.GatewayList ;
        while (PIpAddr <> Nil) do
        begin
            AdpRows [AdpTot].GatewayList [I] := PIpAddr.IpAddress ;
            PIpAddr := PIpAddr.Next ;
            inc (I) ;
            if Length (AdpRows [AdpTot].GatewayList) <= I then
                        SetLength (AdpRows [AdpTot].GatewayList, I -2) ;
        end ;
        AdpRows [AdpTot].GatewayTot := I ;

    // беремо список IP адрес для  GatewayList
        I := 0 ;
        PIpAddr := @AdapterInfo^.DHCPServer ;
        while (PIpAddr <> Nil) do
        begin
            AdpRows [AdpTot].DHCPServer [I] := PIpAddr.IpAddress ;
            PIpAddr := PIpAddr.Next ;
            inc (I) ;
            if Length (AdpRows [AdpTot].DHCPServer) <= I then
                        SetLength (AdpRows [AdpTot].DHCPServer, I -2) ;
        end ;
        AdpRows [AdpTot].DHCPTot := I ;

    // беремо список IP адрес для PrimaryWINSServer
        I := 0 ;
        PIpAddr := @AdapterInfo^.PrimaryWINSServer ;
        while (PIpAddr <> Nil) do
        begin
            AdpRows [AdpTot].PrimWINSServer [I] := PIpAddr.IpAddress ;
            PIpAddr := PIpAddr.Next ;
            inc (I) ;
            if Length (AdpRows [AdpTot].PrimWINSServer) <= I then
                        SetLength (AdpRows [AdpTot].PrimWINSServer, I -2) ;
        end ;
        AdpRows [AdpTot].PrimWINSTot := I ;

    // беремо список IP адрес для SecondaryWINSServer
        I := 0 ;
        PIpAddr := @AdapterInfo^.SecondaryWINSServer ;
        while (PIpAddr <> Nil) do
        begin
            AdpRows [AdpTot].SecWINSServer [I] := PIpAddr.IpAddress ;
            PIpAddr := PIpAddr.Next ;
            inc (I) ;
            if Length (AdpRows [AdpTot].SecWINSServer) <= I then
                        SetLength (AdpRows [AdpTot].SecWINSServer, I -2) ;
        end ;
        AdpRows [AdpTot].SecWINSTot := I ;

        AdpRows [AdpTot].LeaseObtained := AdapterInfo^.LeaseObtained ;
        AdpRows [AdpTot].LeaseExpires := AdapterInfo^.LeaseExpires ;
```

```
            inc (AdpTot) ;
            if Length (AdpRows) <= AdpTot then
                        SetLength (AdpRows, AdpTot -2) ;  // більше пам' яті
            AdapterInfo := AdapterInfo^.Next;
          end ;
          SetLength (AdpRows, AdpTot) ;
       end ;
  finally
      FreeMem( pBuf );
  end ;
end ;

procedure Get_AdaptersInfo( List: TStrings );
var
  AdpTot: integer;
  AdpRows: TAdaptorRows ;
  Error: DWORD ;
  I: integer ;
  //J: integer ;
  //S: string ;        id.
begin
  if not Assigned( List ) then EXIT;
  List.Clear;
  SetLength (AdpRows, 0) ;
  AdpTot := 0 ;
  Error := IpHlpAdaptersInfo(AdpTot, AdpRows) ;
  if (Error <> 0) then
      List.Add( SysErrorMessage( GetLastError ) )
  else if AdpTot = 0 then
      List.Add( ' данних немає '  )
  else
  begin
      for I := 0 to Pred (AdpTot) do
      begin
        with AdpRows [I] do
        begin
            //List.Add(AdapterName + ' |'  + Description ); // jpt : не
використовується
            List.Add( Format(' %8.8x | %6s | %16s | %2d | %16s | %16s | %16s' ,
                [Index, AdaptTypes[aType], MacAddress, DHCPEnabled,
                GatewayList [0], DHCPServer [0], PrimWINSServer [0]] ) );
            {if IPAddressTot <> 0 then    // jpt : не використовується
            begin
                S := ' '  ;
                for J := 0 to Pred (IPAddressTot) do
                    S := S + IPAddressList [J] + ' /'  + IPMaskList [J] + '
| '  ;
                List.Add(IntToStr (IPAddressTot) + '  IP Addresse(s): '  + S);
            end ;
            List.Add( '   '  ); }
        end ;
      end ;
  end ;
  SetLength (AdpRows, 0) ;
end ;

//-----------------------------------
{  моніторимо  час  доступу  до  IP  досліджуємої  мережі  для  оцінки  якості
обслуговування (NPMD_APM)}
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; // Расположення BAD_HOST_NAME,etc...
    HopCount :=-1;
  end
```

```
  else
    Result := NO_ERROR;
end;

//-----------------------------------
{ ARP-таблиця включає відношення між віддаленим IP та віддаленим MAC-адресом.
}
procedure Get_ARPTable( List: TStrings );
var
  IPNet_NGN_Row      : TMibIPNet_NGN_Row;
  TableSize     : DWORD;
  NumEntries    : DWORD;
  ErrorCode     : DWORD;
  i             : integer;
  pBuf          : PChar;
begin
  if not Assigned( List ) then EXIT;
  List.Clear;
  // перший виклик: беремо довжину таблиці
  TableSize := 0;
  ErrorCode := GetIPNet_NGN_Table( Nil, @TableSize, false );   // дані
  //
  if ErrorCode = ERROR_NO_DATA then
  begin
    List.Add( '  ARP-кеш пустий.'  );
    EXIT;
  end;
  // беремо таблицю
  GetMem( pBuf, TableSize );
  NumEntries := 0 ;
  try
  ErrorCode := GetIpNet_NGN_Table( PTMIBIPNet_NGN_Table( pBuf ), @TableSize,
false );
  if ErrorCode = NO_ERROR then
  begin
    NumEntries := PTMIBIPNet_NGN_Table( pBuf )^.dwNumEntries;
    if NumEntries > 0 then
    begin
      inc( pBuf, SizeOf( DWORD ) ); // беремо розмір останьої таблиці
      for i := 1 to NumEntries do
      begin
        IPNet_NGN_Row := PTMIBIPNet_NGN_Row( PBuf )^;
        with IPNet_NGN_Row do
          List.Add( Format( ' %8x | %12s | %16s | %10s' ,
                            [dwIndex, MacAddr2Str( bPhysAddr, dwPhysAddrLen ),
                            IPAddr2Str( dwAddr ), ARPEntryType[dwType]
                            ]));
        inc( pBuf, SizeOf( IPNet_NGN_Row ) );
      end;
    end
    else
      List.Add( '  ARP-кеш пустий.'  );
  end
  else
    List.Add( SysErrorMessage( ErrorCode ) );

  // необхідно відновити показник!
  finally
      dec( pBuf, SizeOf( DWORD ) + NumEntries -SizeOf( IPNet_NGN_Row ) );
      FreeMem( pBuf );
  end ;
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: беремо довжину таблиці
  TableSize := 0;
  NumEntries := 0 ;
  ErrorCode := GetTCPTable(Nil, @TableSize, false );  // дані
  if Errorcode <> ERROR_INSUFFICIENT_BUFFER then
    EXIT;

  // беремо розмір пам' яті, викликаємо знову
  GetMem( pBuf, TableSize );
  // беремо таблицю
  ErrorCode := GetTCPTable( PTMIBTCPTable( pBuf ), @TableSize, false );
  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 останій запис
        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;

//-----------------------------------
procedure Get_TCPStatistics( List: TStrings );
var
  TCPStats      : TMibTCPStats;
  ErrorCode     : DWORD;
begin
  if not Assigned( List ) then EXIT;
  List.Clear;
  if NOT LoadIpHlp then exit ;
  ErrorCode := GetTCPStatistics( @TCPStats );
  if ErrorCode = NO_ERROR then
```

```
    with TCPStats do
    begin
      List.Add( ' Алгоритм повторної передачі : '  + TCPToAlgo[dwRTOAlgorithm]
);
      List.Add( ' Мінімальний час виходу         : '  + IntToStr( dwRTOMin ) + '
ms'  );
      List.Add( ' Максимальний час виходу        : '  + IntToStr( dwRTOMax ) +
'  ms'  );
      List.Add( ' Максимальне число підключень : '  + IntToStr( dwRTOAlgorithm )
);
      List.Add( ' Активні підключення            : '  + IntToStr( dwActiveOpens
) );
      List.Add( ' пасивні підключення           : '  + IntToStr( dwPassiveOpens
) );
      List.Add( ' Невдала спроба відкриття     : '  + IntToStr( dwAttemptFails )
);
      List.Add( ' Скидання встановленого підключення  : '  + IntToStr(
dwEstabResets ) );
      List.Add( ' Поточне встановлене підключення.: '  + IntToStr( dwCurrEstab )
);
      List.Add( ' Отримані сегменти        : '  + IntToStr( dwInSegs ) );
      List.Add( ' Передані сегменти            : '  + IntToStr( dwOutSegs ) );
      List.Add( ' Перепідключені сегменти   : '  + IntToStr( dwReTransSegs ) );
      List.Add( ' помилка входу          : '  + IntToStr( dwInErrs ) );
      List.Add( ' Перезавантаження вихідних       : '  + IntToStr( dwOutRsts
) );
      List.Add( ' Совокупні зв' язки    : '  + IntToStr( dwNumConns ) );
    end
  else
    List.Add( SyserrorMessage( ErrorCode ) );
end;

function IpHlpTCPStatistics (var TCPStats: TMibTCPStats): integer ;
begin
    result := ERROR_NOT_SUPPORTED ;
    if NOT LoadIpHlp then exit ;
    result := GetTCPStatistics( @TCPStats );
end;

//-----------------------------------
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: беремо довжину таблиці
  TableSize := 0;
  NumEntries := 0 ;
  ErrorCode := GetUDPTable(Nil, @TableSize, false );
  if Errorcode <> ERROR_INSUFFICIENT_BUFFER then
    EXIT;

  // виділяємо пам' ять, викликаємо знову
  GetMem( pBuf, TableSize );

  // беремо таблицю
  ErrorCode := GetUDPTable( PTMIBUDPTable( pBuf ), @TableSize, false );
  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 останій запис
        with UDPRow do
          List.Add( Format( ' %15s : %-6s' ,
            [IpAddr2Str( dwLocalAddr ),
            Port2Svc( Port2Wrd( dwLocalPort ) )
              ] ) );
        inc( pBuf, SizeOf( TMIBUDPRow ) );
      end;
    end
    else
      List.Add( ' немає даних.'  );
  end
  else
    List.Add( SyserrorMessage( ErrorCode ) );
  dec( pBuf, SizeOf( DWORD ) + NumEntries -SizeOf( TMibUDPRow ) );
  FreeMem( pBuf );
end;

//-----------------------------------
procedure Get_IPAddrTable( List: TStrings );
var
  IPAddrRow     : TMibIPAddrRow;
  TableSize     : DWORD;
  ErrorCode     : DWORD;
  i             : integer;
  pBuf          : PChar;
  NumEntries    : DWORD;
begin
  if not Assigned( List ) then EXIT;
  List.Clear;
  TableSize := 0; ;
  NumEntries := 0 ;
  // перший виклик: беремо довжину таблиці
  ErrorCode := GetIpAddrTable(Nil, @TableSize, true );  // дані
  if Errorcode <> ERROR_INSUFFICIENT_BUFFER then
    EXIT;

  GetMem( pBuf, TableSize );
  // беремо 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( ' немає даних.'  );
  end
  else
    List.Add( SysErrorMessage( ErrorCode ) );
```

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

//-----------------------------------
{ отримуємо дані з таблиці маршрутизації досліджуємої мережі для оцінки якості
обслуговування (NPMD_APM); }
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: беремо довжину таблиці
  NumEntries := 0 ;
  ErrorCode := GetIpForwardTable(Nil, @TableSize, true);
  if Errorcode <> ERROR_INSUFFICIENT_BUFFER then
    EXIT;

  // беремо 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
        begin
          if (dwForwardType < 1)
          or (dwForwardType > 4) then
                dwForwardType := 1 ;   // дані
          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
            ] ) );
        end ;
        inc( pBuf, SizeOf( TMibIPForwardRow ) );
      end;
    end
    else
      List.Add( ' немає даних.'  );
  end
  else
    List.Add( SysErrorMessage( ErrorCode ) );
  dec( pBuf, SizeOf( DWORD ) + NumEntries -SizeOf( TMibIPForwardRow ) );
  FreeMem( pBuf );
end;
```

```
//-----------------------------------
procedure Get_IPStatistics( List: TStrings );
var
  IPStats       : TMibIPStats;
  ErrorCode     : integer;
begin
  if not Assigned( List ) then EXIT;
  if NOT LoadIpHlp then exit ;
  ErrorCode := GetIPStatistics( @IPStats );
  if ErrorCode = NO_ERROR then
  begin
    List.Clear;
    with IPStats do
    begin
      if dwForwarding = 1 then
        List.add( ' Розблокована пересилка      : '  + ' так'  )
      else
        List.add(' Розблокована пересилка      : '  + ' ні'  );
      List.add( ' Любий TTL             : '  + inttostr( dwDefaultTTL ) );
      List.add( ' Датаграмма прийнята     : '  + inttostr( dwInReceives ) );
      List.add( ' Помилка заголовку      (In)  : '  + inttostr( dwInHdrErrors )
);
      List.add( ' Помилка адреси    (In)  : '  + inttostr( dwInAddrErrors ) );
      List.add( ' Датаграма переслана     : '  + inttostr( dwForwDatagrams ) );
// дані
      List.add( ' Невизначений протокол (In)  : '  + inttostr( dwInUnknownProtos
) );
      List.add( ' Датаграмма відмовлена     : '  + inttostr( dwInDiscards ) );
      List.add( ' Датаграма встановлена     : '  + inttostr( dwInDelivers ) );
      List.add( ' Зовнішній запрос          : '  + inttostr( dwOutRequests )
);
      List.add( ' Маршрутизація не виконана      : '  + inttostr(
dwRoutingDiscards ) );
      List.add( ' Немає маршрутів       (Out)  : '  + inttostr( dwOutNoRoutes )
);
      List.add( ' Перебраний час     : '  + inttostr( dwReasmTimeOut ) );
      List.add( ' Запит перебору     : '  + 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( ' Датаграмма фрагментована    : '  + inttostr( dwFRagCreates )
);
      List.add( ' Кількість інтерфейсів    : '  + inttostr( dwNumIf ) );
      List.add( ' Кількість IP-адрес  : '  + inttostr( dwNumAddr ) );
      List.add( ' Маршрут в таблиці маршрутизатора  : '  + inttostr( dwNumRoutes
) );
    end;
  end
  else
    List.Add( SysErrorMessage( ErrorCode ) );
end;

function IpHlpIPStatistics (var IPStats: TMibIPStats): integer ;      // дані
begin
    result := ERROR_NOT_SUPPORTED ;
    if NOT LoadIpHlp then exit ;
    result := GetIPStatistics( @IPStats );
end ;

//-----------------------------------
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 (In)    : '  + inttostr( dwInDatagrams ) );
      List.add( '  Датаграми (Out)   : '  + inttostr( dwOutDatagrams ) );
      List.add( ' Немає портів         : '  + inttostr( dwNoPorts ) );
      List.add( ' Помилка    (In)    : '  + inttostr( dwInErrors ) );
      List.add( ' UDP список портів  : '  + inttostr( dwNumAddrs ) );
    end;
  end
  else
    List.Add( SysErrorMessage( ErrorCode ) );
end;

//-------------------------------------*//
function IpHlpUdpStatistics (UdpStats: TMibUDPStats): integer ;     // дані
begin
    result := ERROR_NOT_SUPPORTED ;
    if NOT LoadIpHlp then exit ;
    result := GetUDPStatistics (@UdpStats) ;
end ;

//-----------------------------------
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( '  Прийнято повідомлень    : '  + IntToStr( dwMsgs ) );
      ICMPIn.Add( ' Помилка                : '  + IntToStr( dwErrors ) );
      ICMPIn.Add( ' Розташування недосягенно    : '  + IntToStr( dwDestUnreachs
) );
      ICMPIn.Add( ' Час перевищений        : '  + IntToStr( dwTimeEcxcds ) );
      ICMPIn.Add( ' Проблеми з параметрами      : '  + IntToStr( dwParmProbs )
);
      ICMPIn.Add( ' Джерело відключено        : '  + IntToStr( dwSrcQuenchs ) );
      ICMPIn.Add( ' Перенаправлено           : '  + IntToStr( dwRedirects ) );
      ICMPIn.Add( ' Ехо запит        : '  + IntToStr( dwEchos ) );
      ICMPIn.Add( ' Ехо відповідь         : '  + IntToStr( dwEchoReps ) );
      ICMPIn.Add( ' Запит мітки часу   : '  + IntToStr( dwTimeStamps ) );
      ICMPIn.Add( ' Відповідь мітки часу    : '  + IntToStr( dwTimeStampReps )
);
      ICMPIn.Add( ' Запит маски адрес : '  + IntToStr( dwAddrMasks ) );
      ICMPIn.Add( ' Відповідь маски адрес   : '  + IntToStr( dwAddrReps ) );
    end;
     //
    with ICMPStats.OutStats do
    begin
      ICMPOut.Add( ' Повідомлення вдправлено        : '  + IntToStr( dwMsgs ) );
      ICMPOut.Add( 'Помилка                : '  + IntToStr( dwErrors ) );
      ICMPOut.Add( 'Розташування недосягенно    : '  + IntToStr( dwDestUnreachs
) );
      ICMPOut.Add( 'Час перевищений        : '  + IntToStr( dwTimeEcxcds ) );
      ICMPOut.Add( 'Проблеми з параметрами      : '  + IntToStr( dwParmProbs )
);
      ICMPOut.Add( 'Джерело відключено        : '  + IntToStr( dwSrcQuenchs ) );
      ICMPOut.Add( 'Перенаправлено           : '  + IntToStr( dwRedirects ) );
      ICMPOut.Add( 'Ехо запит        : '  + IntToStr( dwEchos ) );
```

```
      ICMPOut.Add( 'Exo відповідь         : '  + IntToStr( dwEchoReps ) );
      ICMPOut.Add( 'Запит мітки часу   : '  + IntToStr( dwTimeStamps ) );
      ICMPOut.Add( 'Відповідь мітки часу    : '  + IntToStr( dwTimeStampReps )
);
      ICMPOut.Add( 'Запит маски адрес: '  + IntToStr( dwAddrMasks ) );
      ICMPOut.Add( 'Відповідь маски адрес   : '  + 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.
```

Файл TCP_IP.pas- монітор TCP/IP з'єднань досліджуємої мережі системи кібербезпеки NPMD та АРМ

```
unit TCP_IP;

interface

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

type
  TForm2 = class(TForm)
    StaticText2: TStaticText;
    StaticText3: TStaticText;
    TCPMemo: TMemo;
    UDPMemo: TMemo;
    Timer1: TTimer;
    cbTimer: TCheckBox;
    btRTTI: TSpeedButton;
    SpeedButton1: TSpeedButton;
    edtRTTI: TEdit;
    procedure Timer1Timer(Sender: TObject);
    procedure SpeedButton1Click(Sender: TObject);
    procedure btRTTIClick(Sender: TObject);
    procedure cbRecentIPsClick(Sender: TObject);
    procedure FormCreate(Sender: TObject);
  private
    { Private declarations }
      procedure DOIpStuff;
  public
    { Public declarations }
  end;

var
  Form2: TForm2;

implementation

{$R *.dfm}

procedure TForm2.Timer1Timer(Sender: TObject);
begin
  if cbTimer.State = cbCHECKED then
 begin
   Timer1.Enabled := false;
   DoIPStuff;
   Timer1.Enabled := true;
 end;
end;

procedure TForm2.DOIpStuff;
begin

  Get_TCPTable( TCPMemo.Lines );
  Get_UDPTable( UDPMemo.Lines );

end;

procedure TForm2.SpeedButton1Click(Sender: TObject);
begin
  Speedbutton1.Enabled := false;
  DoIPStuff;
  Speedbutton1.Enabled := true;
end;

procedure TForm2.btRTTIClick(Sender: TObject);
```

```
var
  IPadr         : dword;
  Rtt, HopCount : longint;
  Res           : integer;
begin
  btRTTI.Enabled := false;
  Screen.Cursor := crHOURGLASS;
  IPadr := Str2IPAddr( edtRTTI.Text );
  Res := Get_RTTAndHopCount( IPadr, 128, RTT, HopCount );
  if Res = NO_ERROR then
    ShowMessage( ' Час запиту '
      + inttostr( rtt ) + ' ms, '
      + inttostr( HopCount )
      + ' hops to : ' + edtRTTI.Text
      )
  else
    ShowMessage( ' Відбулася помилка:' + #13
                 + ICMPErr2Str( Res ) ) ;
  btRTTI.Enabled := true;
  Screen.Cursor := crDEFAULT;

end;

procedure TForm2.cbRecentIPsClick(Sender: TObject);
begin
//edtRTTI.Text := cbRecentIPs.Items[cbRecentIPs.ItemIndex];
end;

procedure TForm2.FormCreate(Sender: TObject);
begin

  if LoadIpHlp then
  begin
      DOIpStuff;
      Timer1.Enabled := true;
  end
  else
      ShowMessage( ' Інтернет помічник DLL не є доступним, або не підтримується'
) ;
end;

end.
```

Файл Stat.pas- статистика мережі системи кібербезпеки NPMD та АРМ

```
unit Stat;

interface

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

type
  TForm3 = class(TForm)
    StaticText7: TStaticText;
    TCPStatMemo: TMemo;
    StaticText5: TStaticText;
    IPStatsMemo: TMemo;
    StaticText12: TStaticText;
    ICMPInMemo: TMemo;
    ICMPOutMemo: TMemo;
    StaticText4: TStaticText;
    UDPStatsMemo: TMemo;
    Timer1: TTimer;
    cbTimer: TCheckBox;
    btRTTI: TSpeedButton;
    edtRTTI: TEdit;
    procedure Timer1Timer(Sender: TObject);
    procedure btRTTIClick(Sender: TObject);
    procedure FormCreate(Sender: TObject);
  private
    { Private declarations }
     procedure DOIpStuff;
  public
    { Public declarations }
  end;

var
  Form3: TForm3;

implementation

{$R *.dfm}

procedure TForm3.DOIpStuff;
begin

  Get_TCPStatistics( TCPStatMemo.Lines );
  Get_IPStatistics( IPStatsMemo.Lines );
  Get_UDPStatistics( UDPStatsMemo.Lines );
  Get_ICMPStats( ICMPInMemo.Lines, ICMPOutMemo.Lines );

end;

procedure TForm3.Timer1Timer(Sender: TObject);
begin
  if cbTimer.State = cbCHECKED then
 begin
   Timer1.Enabled := false;
   DoIPStuff;
   Timer1.Enabled := true;
 end;
end;

procedure TForm3.btRTTIClick(Sender: TObject);
var
  IPadr         : dword;
  Rtt, HopCount : longint;
  Res           : integer;
```

```
begin
  btRTTI.Enabled := false;
  Screen.Cursor := crHOURGLASS;
  IPadr := Str2IPAddr( edtRTTI.Text );
  Res := Get_RTTAndHopCount( IPadr, 128, RTT, HopCount );
  if Res = NO_ERROR then
    ShowMessage( ' Час запиту '
      + inttostr( rtt ) + ' ms, '
      + inttostr( HopCount )
      + ' hops to : ' + edtRTTI.Text
      )
  else
    ShowMessage( ' Помилка:' + #13
                 + ICMPErr2Str( Res ) ) ;
  btRTTI.Enabled := true;
  Screen.Cursor := crDEFAULT;

end;

procedure TForm3.FormCreate(Sender: TObject);
begin

  if LoadIpHlp then
  begin
      DOIpStuff;
      Timer1.Enabled := true;
  end
  else
      ShowMessage( 'Інтернет помічник DLL не є доступним, або не підтримується'
) ;
end;

end.
```

Файл About.pas - довідка

```
unit About;

interface

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

type
  TForm1 = class(TForm)
    Label1: TLabel;
    Label2: TLabel;
    Label3: TLabel;
    Label4: TLabel;
    Label5: TLabel;
    Label7: TLabel;
    Label8: TLabel;
    Label9: TLabel;
    Button1: TButton;
    Image2: TImage;
    Image1: TImage;
    Image3: TImage;
    procedure Button1Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form1: TForm1;

implementation

{$R *.dfm}

procedure TForm1.Button1Click(Sender: TObject);
begin
Form1.Close;
end;

end.
```