Центральноукраїнський національний технічний університет
Центр заочної та дистанційної освіти
Кафедра кібербезпеки та програмного забезпечення

”Допущено до захисту”
Завідувач кафедри кібербезпеки
та програмного забезпечення
д.т.н., професор
_____ Олексій СМІРНОВ
“____” _____ 2023 р.

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА
за першим (бакалаврським) рівнем вищої освіти
на тему
“Програмне забезпечення системи кібербезпеки для виявлення аномалій телекомунікаційного трафіку на основі спектрально-часового аналізу”

Виконав здобувач вищої освіти
IV курсу, групи КБ-21СКЗ
ОПП «Кібербезпека»
спеціальності 125 «Кібербезпека»
_____ Галушка Д.А.
«____» _____ 2023 р.

Керівник проекту
доктор технічних наук, професор
_____ Смірнов О.А.
«____» _____ 2023 р.
Рецензент _____
_____

м. Кропивницький

Центральноукраїнський національний технічний університет

Центр *Заочної та дистанційної освіти*
Кафедра *Кібербезпеки та програмного забезпечення*
Освітній ступінь *бакалавр*
Галузь знань ._12 _____*"*Інформаційні технології*"*_____
Спеціальність *125 "*Кібербезпека*"*
Освітньо-професійна (освітньо-наукова) програма _*"*Кібербезпека*"*_

ЗАТВЕРДЖУЮ
Завідувач кафедри
д.т.н., проф.
_____ Олексій СМІРНОВ
« 17 » січня 2023 року

ЗАВДАННЯ НА ВИПУСКНУ КВАЛІФІКАЦІЙНУ РОБОТУ ЗА ПЕРШИМ (БАКАЛАВРСЬКИМ) РІВНЕМ ВИЩОЇ ОСВІТИ ЗДОБУВАЧА ВИЩОЇ ОСВІТИ

Галущі Дмитру Андрійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи *Програмне забезпечення системи кібербезпеки для виявлення аномалій телекомунікаційного трафіку на основі спектрально-часового аналізу*

2. Керівник роботи *Смірнов Олексій Анатолійович, докт. техн. наук, професор*
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу № 17-02 від 5.01.2023 року

3. Строк подання студентом роботи до захисту *23.05.2023 р.*

4. Мета та завдання випускної кваліфікаційної роботи*: Метою роботи є розробка програмного забезпечення системи кібербезпеки для виявлення аномалій телекомунікаційного трафіку на основі спектрально-часового аналізу*

5. 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 кібербезпеки в промислову експлуатацію.
6. Висновки

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

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

7. Дата видачі завдання « 17 » січня 2023 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти	Строк виконання етапів випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти	Примітка
1.	Аналіз існуючих систем	10.03.2023 р.	
2.	Постановка задачі, оформлення ТЗ	15.03.2023 р.	
3.	Розробка моделі компонента	20.03.2023 р.	
4.	Розробка структур даних	25.03.2023 р.	
5.	Розробка алгоритмів зв’язку та відображення	30.03.2023 р.	
6.	Програмування алгоритмів	10.04.2023 р.	
7.	Оформлення ПЗ	17.04.2023 р.	
8.	Попередній захист роботи	23.05.2023 р.	

Дата видачі завдання
« 17 » січня 2023 р.

Підпис керівника
_____ Смірнов О.А.
(прізвище та ініціали)

Завдання прийнято до виконання
« 17 » січня 2023 р.

Підпис здобувача
_____ Галушка Д.А.
(прізвище та ініціали)

АНОТАЦІЯ

Галушка Д.А. Програмне забезпечення системи кібербезпеки для виявлення аномалій телекомунікаційного трафіку на основі спектрально-часового аналізу. 125 Кібербезпека. Центральноукраїнський національний технічний університет. Кропивницький. 2023.

В даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи кібербезпеки для виявлення аномалій телекомунікаційного трафіку на основі спектрально-часового аналізу.

Метою розробки є програмне забезпечення системи кібербезпеки для виявлення аномалій телекомунікаційного трафіку на основі спектрально-часового аналізу.

Результат роботи – програмна реалізація системи кібербезпеки для виявлення аномалій телекомунікаційного трафіку на основі спектрально-часового аналізу.

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 10/11.

Програму розроблено в середовищі Delphi 10.

Ключові слова: кібербезпека, аномалії, телекомунікаційний трафік

ABSTRACT

Halushka D.A. Cybersecurity system software for detecting telecommunication traffic anomalies based on spectrum-temporal analysis. 125 Cyber security. Central Ukrainian National Technical University. Kropyvnytskyi. 2023.

In this graduation thesis for the first (bachelor) level of higher education, software is developed, which is intended for the cyber security system to detect anomalies in telecommunication traffic based on spectral-temporal analysis.

The purpose of the development is the software of the cyber security system for the detection of telecommunication traffic anomalies based on spectral-temporal analysis.

The result of the work is the software implementation of the cyber security system for the detection of telecommunication traffic anomalies based on spectral-temporal analysis.

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.

A convenient user interface has been developed. Instructions for working with software tools are provided.

The program can be used on PCs of IBM PC architecture with Windows 10/11 OS.

The program was developed in the Delphi 10 environment.

Keywords: cyber security, anomalies, telecommunication traffic

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ

ЕОМ	–	електрона обчислювальна машина
КВ	–	коефіцієнт варіації
КЗ	–	канал зв'язку
НСД	–	несанкціонований доступ
ПС	–	програмна середа
СВВ	–	система виявлення вторгнень
СеМО	–	експонентна мережа масового обслуговування
СМО	–	система масового обслуговування
СПД	–	система передачі даних

ВСТУП

Актуальність теми. Активне використання на сучасний час розподілених комп'ютерних систем та мереж приводить до необхідності приділяти увагу питанням безпеки. Особливе місце в реалізації політики безпеки організації займають системи виявлення вторгнень (подій з безпекою) (СВВ), які можуть як виконувати функцію зворотного зв'язка, контролюючи ефективність компонентів системи безпеки, тобто бути, як доповненням до існуючого комплексу засобів захисту, так і являти собою самостійний продукт. Впровадження багатьох СВВ, як і комплексних систем безпеки, стримує ряд факторів, таких як одноразові капіталовкладення, необхідність компетентної установки, налаштування, підтримки й т.д. У таких компаніях, як правило, функція спостереження за роботою мережі покладається на адміністратора. У такому випадку результат залежить від людського фактора, що включає досвід, інтуїцію, відповідальність, працездатність і т.п. Слід зазначити, що практично в кожній компанії, що має в розпорядженні розподілену мережу, установлені засоби збору статистичних даних про завантаження інтерфейсів мережного встаткування. Таким чином, закономірним кроком до автоматизації процесу виявлення позаштатних ситуацій, є впровадження доступного й, можна сказати, універсального засобу, що аналізує інтенсивності потоків даних у пошуку незвичайних і підозрілих подій або тенденцій, яке можна віднести до підкласу СВВ. При цьому може використовуватися як сигнатурний метод, так і метод описової статистики.

Математично обґрунтованими видами аналізу часових рядів є дослідження сигналу на основі часових, спектральних і спектрально-часових алгоритмів, які інтенсивно розвиваються останнє з невеликим десятиліття.

Аналіз у часовій області ґрунтується на методах математичної статистики і його можливості досить великі. Але слід зазначити, що досліджуваному телекомунікаційному сигналу властиво «виражене коливальне поводження»

через його особливості його формування. Хоча в методах часового аналізу існують підходи для опису такого роду сигналів, найбільш підходящими для дослідження коливального процесу є методи спектрального й спектрально-часового аналізу. Частотні подання є більше інформативними й дозволяють розширити можливості існуючих систем виявлення аномалій, але вимагають більших розмірностей для подання результатів і мають більшу обчислювальну складність алгоритмів, що стримує їхнє застосування й розвиток у прикладних завданнях. Отже, є актуальною розробка методу виявлення аномалій в інтенсивностях потоків даних на основі алгоритмів аналізу частотних складових, оптимізованих по обчислювальному навантаженню.

Мета й завдання дослідження. Метою роботи є програмне забезпечення системи кібербезпеки для виявлення аномалій телекомунікаційного трафіку на основі спектрально-часового аналізу.

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

– Огляд існуючих систем для виявлення аномалій телекомунікаційного трафіку на основі спектрально-часового аналізу.

– Дослідження системи кібербезпеки для виявлення аномалій телекомунікаційного трафіку на основі спектрально-часового аналізу.

– Програмна реалізація системи кібербезпеки для виявлення аномалій телекомунікаційного трафіку на основі спектрально-часового аналізу.

Практична цінність отриманих результатів полягає в тому, що розроблені алгоритми дозволяють успішно вирішувати задачі для виявлення аномалій телекомунікаційного трафіку на основі спектрально-часового аналізу.

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи кібербезпеки для виявлення аномалій телекомунікаційного трафіку на основі спектрально-часового аналізу, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти.

1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ

1.1 Призначення системи

Система призначена для виявлення аномалій телекомунікаційного трафіку на основі спектрально-часового аналізу. У наш час кібертероризм усе більше знаходить реальні риси. Ефективно протистояти віртуальному ворогові можна, тільки знаючи його в особу. Тому пропонується класифікація мережних аномалій, що відбиває сучасний стан проблеми.

Класифікація мережних аномалій

Відомі мережні аномалії (МА) настільки різноманітні, що єдиній класифікації вони не піддаються. Так, існує розподіл на активні й пасивні, зовнішні й внутрішні, навмисні й ненавмисні аномалії й т.д. Однак дані підходи не відбивають всіх характеристик досліджуваного явища і є обмеженими. Тому автором пропонується класифікація МА з погляду об'єкта впливу – інформаційної системи (ІС), що включає програмно-апаратний комплекс і мережну інфраструктуру.

Відповідно до обраного підходу можна поділити МА на дві основні групи:

– програмно-апаратні відхилення;

– проблеми безпеки.

До програмно-апаратних відхилень відносяться:

– апаратні несправності;

– помилки конфігурування;

– помилки програмного забезпечення;

– проблеми продуктивності встаткування.

Порушення мережної безпеки містять у собі наступні аномалії:

– сканування, атаки з метою відмови від обслуговування;

– вірусна активність;

– поширення програмних "хробаків";

– експлуатація уразливостей;

– аналізатори трафіку (сніфери);

– мережні модифікатори.

Найбільший економічний збиток операторам зв'язку наносять атаки з метою перевантаження мереж або сервісів і мережна вірусна активність.

Програмно-апаратні відхилення

Помилки програмного забезпечення компонентів ІС можуть викликати переклад у позаштатний режим з наступним припиненням надання сервісів.

Помилки конфігурування переводять функціональні можливості компонентів ІС у невідповідність штатним проектним параметрам, що порушує загальну працездатність.

Порушення продуктивності спричиняють вихід параметрів ІС за межі розрахункових значень, що супроводжується порушенням забезпечення надання сервісів.

Апаратні несправності можуть викликати як повний вихід з ладу окремих компонентів ІС, так і деградуючий вплив окремої підсистеми на весь комплекс.

Порушення безпеки

Мережне сканування (network scan) виробляється з метою аналізу топології мережі й виявлення доступних для атаки сервісів. У процесі сканування виробляється спроба з'єднання з мережними сервісами методом звертання за визначеним портом. У випадку відкритого сканування сканер виконує тристоронню процедуру квітирування, а у випадку закритого (stealth) – не завершує з'єднання. Тому що при скануванні окремого хосту відбувається перебір сервісів (портів), те дана аномалія характеризується спробами звертання з однієї IP адреси сканера на визначену IP адреса по безлічі портів. Однак, найчастіше скануванню піддаються цілі підмережі, що виражається в наявності в атакованій мережі безлічі пакетів з одного IP адреси сканера по безлічі IP адрес досліджуваної підмережі, іноді навіть методом послідовного перебору. Найбільш відомими мережними сканерами є: nmap, ISS, satan, strobe, xscan і інші.

Аналізатори трафіку або сніффери призначені для перехоплення й аналізу мережного трафіку. У найпростішому випадку для цього виробляється переклад мережного адаптера апаратного комплексу в режим, що прослуховує, і потоки даних у сегменті, до якого він підключений, стають доступні для подальшого вивчення. Тому що багато прикладних програм використовують протоколи, що передають інформацію у відкритому, незашифрованому виді, робота сніфферів різко знижує рівень безпеки. Відзначимо, що виражених аномалій у роботі мережі сніффери не викликають. Найбільш відомими сніфферами є: tcpdump, ethereal, sniffit, Microsoft network monitor, netxray, lan explorer.

У комп'ютерній безпеці термін уразливість (vulnerability) використовується для позначення слабозахищеного від несанкціонованого впливу компонента ІС. Уразливість може бути результатом помилок проектування, програмування або конфігурування. Уразливість може існувати тільки теоретично або мати експлуатуючу програмну реалізацію – експлойт. У мережному аспекті уразливостям можуть бути піддані інформаційні ресурси, такі як операційні системи й ПЗ сервісів.

Вірусна мережна активність є результатом спроб поширення комп'ютерних вірусів і хробаків, використовуючи мережні ресурси. Найчастіше комп'ютерний вірус експлуатує яку-небудь єдину уразливість у мережній прикладній службі, тому вірусний трафік характеризується наявністю безлічі звертань з однієї зараженої IP адреси до багатьох IP адрес по визначеному порту, що відповідає потенційно уразливому сервісу.

Мережні модифікатори роблять перекручування переданих по мережі даних з метою порушення установлених з'єднань або одержання несанкціонованого доступу до інформаційних ресурсів. До даного класу аномалій відносяться спуфінг (spoofing), урізання (in-the-middle) і інші. Технологія спуфінга дозволяє зловмисникові генерувати мережні пакети з підробленою адресою відправника, що належить закритій мережі, видаючи себе за санкціонованого користувача. Порушення типу урізання виражаються в

модифікації мережних потоків даних між кінцевими учасниками з'єднання або підміні одного з мережних сервісів.

Атаки типу "відмова в обслуговуванні" (Denial of Service, DoS) приведуть до перевантаження й неприступності інформаційних (сервери, сервіси) або мережних ресурсів (канали зв'язку, комутатори, маршрутизатори).

Основні типи DoS атак:

– Атаки, спрямовані на перевантаження інформаційних ресурсів серверів (ОС і додатків). Приклад: mailbomb.

– Атаки, що використовують помилки в реалізації стека протоколів TCP/IP в ОС. Для цього використовується генерація спеціально сконструйованої серії пакетів, при обробці яких відбувається збій у роботі ОС. Приклади: teardrop, land.

– Блокування каналів зв'язку й маршрутизаторів здійснюється за допомогою потужного потоку пакетів (flood, затоплення), повністю задіючі обчислювальні потужності маршрутизаторів або смугу пропущення каналу зв'язку. У підсумку легітимний трафік ігноруються й користувачі одержують відмову в доступі.

Існують наступні різновиди DoS атак з погляду мережних характеристик:

– Tcp flood – потік tcp пакетів.

– Tcp syn flood – потік tcp пакетів із прапором установки з'єднання.

– Udp flood – потік udp пакетів.

– Icmp unicast flood – потік icmp пакетів.

– Icmp broadcast flood – пакети з підробленою адресою джерела й широкомовною адресою призначення викликають потік відповідей на дану адресу джерела Приклади: smurf.

– Ip packet fragmentation – потік фрагментованих пакетів.

– Distributed DoS (DDoS) – розподілена атака DoS, що використовує безліч мережних джерел Приклади: Tribe Flood Network (TFN), Stacheldracht, Trinoo.

1.2 Область застосування

Область застосування системи, яка розробляється є системи виявлення вторгнень у комп'ютерну мережу. Виконаємо огляд і класифікацію систем виявлення вторгнень (СВВ) і визначимо їхнє місце в завданнях мережної безпеки.

СВВ визначається як програмний або апаратний засіб, що збирає інформацію з різних точок комп'ютерної системи, що захищається (обчислювальної мережі) і який аналізує цю інформацію для виявлення, як спроб порушення, так і реалізованих порушень захисту (вторгнень). СВВ забезпечують додатковий рівень захисту комп'ютерних систем, але можуть бути й самостійним інструментом. Звичайно архітектура СВВ включає:

– сенсорну підсистему, призначену для збору інформації (подій, пов'язаних з безпекою системи, яка захищається);

– підсистему аналізу (виявлення), призначену для виявлення атак і підозрілих дій на основі даних сенсорів;

– підсистему подання даних (користувальницький інтерфейс, консоль керування), що дозволяє конфігурувати СВВ, спостерігати за станом системи, яка захищається, і СВВ, переглядати виявлені підсистемою аналізу інциденти.

Іноді в архітектуру СВВ включають сховище, що забезпечує накопичення первинних подій і результатів аналізу.

Серед СВВ виділені системи виявлення аномалій, основною функцією яких є виявлення незвичайних і підозрілих подій або тенденцій у динаміку спостережуваного процесу.

Запропонуємо наступну класифікацію СВВ.

1. За оброблюваною інформацією:

– мережні;

– вузлові;

– гібридні.

2. За методами:

– виявлення аномалій і пошук зловживань;

– контрольоване й неконтрольоване навчання;
– моделювання правил;
– моделювання станів;
– описова статистика;
– нейронні мережі;
– експертні системи;
– синтаксичний аналіз.

3. За відповідною реакцією:
– пасивні;
– активні.

4. За типом поширення:
– дослідницькі;
– вільно розповсюджувані;
– комерційні;
– для державних установ.

Розглянемо можливості побудові систем виявлення аномалій на основі дослідження інтенсивностей потоків даних у мережі. Автоматичний пошук елементарних збурювань може вироблятися на основі сигнатурного методу й методу описової статистики.

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи кібербезпеки для виявлення аномалій телекомунікаційного трафіку на основі спектрально-часового аналізу, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти.

2 ПЕРЕГЛЯД АНАЛОГІЧНИХ ІСНУЮЧИХ СИСТЕМ

2.1 Огляд існуючих систем, технологій, архітектур, програмних рішень за профілем теми випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти

Для виявлення аномалій телекомунікаційного трафіку на основі спектрально-часового аналізу пропонується використовувати сніффери. Тому проведемо огляд сніфферів. Сніффери – це програми, які перехоплюють весь мережний трафік. Сніффери корисні для діагностики мережі (для адміністраторів) і для перехоплення паролів (зрозуміло для кого). Наприклад, якщо одержати доступ до однієї мережної машини й установити там сніффер, то незабаром всі паролі від їх підмережі будуть відомі. Сніффери ставлять мережну карту в режим, що прослуховує (PROMISC). Тобто вони одержують всі пакети. У локальній мережі можна перехоплювати всі пакети, що відправляються, із всіх машин (якщо ви не розділені всякими хабами), тому що там практикується широкомовлення. Сніффери можуть перехоплювати всі пакети (що дуже незручно, жахливо швидко переповняється лог файл, зате для більше детального аналізу мережі саме воно) або тільки перші байти від усяких ftp, telnet, pop3 і т.д. Сніфферів зараз багато. Безліч сніфферів є як під Unix, так і під Windows (навіть під DOS є). Сніффери можуть підтримувати тільки визначену операційну систему (наприклад, linux_sniffer.c, що підтримує Linux), або декілька (наприклад, Sniffit, працює з BSD, Linux, Solaris). Сніффери так розмножилися через те, що паролі передаються по мережі відкритим текстом. Таких служб багато. Це telnet, ftp, pop3, www і т.д. Цими службами користується багато народу. Після буму сніфферів почали з'являтися різні алгоритми шифрування цих протоколів. З'явилася SSH (альтернатива telnet, що підтримує шифрування), SSL (Secure Socket Layer – розробка Netscape, здатна зашифрувати www сеанс). З'явилися Kerberous,

VPN(Virtual Private Network). З'явилися якісь AntiSniff'и, ifstatus'и й т.д. Але це в корені не змінило положення. Служби, які використовують передачу пароля plain text'ом, використовуються щосили, тому сніффувати будуть ще довго.

Windows-реалізації сніфферів

CommView

Розроблювач: TamoSoft, Inc.

Досить просунутий сніффер виробництва TamoSoft Inc., скачати можна на сайті. Можна встановити свої правила на сніффінг (наприклад, ігнорувати ICMP, а TCP сніффувати, також крім інтернет-протоколів є підтримка ethernet-протоколів, таких як ARP, SNMP, NOVELL і т.д.). Можна, наприклад, сніффувати тільки вхідні пакети, а інші ігнорувати, можна вказати лог-файл для всіх пакетів з лімітів розміру в мегабайтах. Має дві утиліти – Packet Generator і NIC Vendor Indentifier. Можна подивитися всі подробиці посланих/отриманих пакетів (наприклад, в TCP-пакеті можна переглянути Source Port, Destination Port, Data length, Checksum, Sequence, Window, Ack, Flags, Urgent). Радує ще те, що вона автоматично встановлює capture-драйвер. У загальному утиліта дуже корисна для сніффінгу, рекомендую всім.

SpyNet

Розроблювач: packetstorm.securify.com.

Досить відомий сніффер виробництва Laurentiu Nicula 2000. Звичайні функції – перехоплення/декодинг пакетів. Хоча декодинг розвинений (можна, наприклад, по пакетах відтворювати сторінки, на яких побував користувач).

Analyzer

Розроблювач: neworder.box.sk.

Analyzer вимагає установку спеціального драйвера, вкладеного в пакет (packet.inf, packet.sys). Можна подивитися всю інформацію про вашу мережну карту. Також Analyzer підтримує роботу з командним рядком. Він прекрасно працює з локальною мережею. Має кілька утиліт: ConvDump, GnuPlot, FlowsDet, Analisys Engine.

Iris Network Traffic Analyzer

Розроблювач: eEye Digital Security.

IRIS продукт відомої фірми eEye скачати можна на сайті. Сніффер не дешевий, але дуже функціональний. Представляє великі можливості по фільтрації. Мене в ньому сильно порадували три функції:

1.Protocol Distribution

2.Top hosts

3.Size Distribution

Також є Packet Decoder, скачати який можна на сайті. Він підтримує розвинену систему логів. А доступні можливості фільтрації перевершують всі сніффери огляду. Це Hardware Filter, що може ловити або всі пакети (Promiscious), або з різними обмеженнями (наприклад, захоплювати тільки multicast пакети або broadcast пакети, або тільки Mac-Фрейми). Можна фільтрувати по певним MAC/IP адресах, по портах, по пакетах, що містить певні символи. Загалом, непоганий сніффер. Вимагає 50comupd.dll.

WinDUMP

Аналог TCPdump for Unix. Цей сніффер діє через командний рядок і представляє мінімальні можливості по конфігурації й ще вимагає бібліотеку WinPcap.

SniffitNT

Теж вимагає WinPcap. Робота тільки як командним рядком, так і в інтерактивному режимі. Зі складними опціями. Мені не дуже.

ButtSniff

Звичайний пакетний сніффер, створений найвідомішою групою CDC (Cult of the Dead Cow). Фішка в тім, що його можна використовувати, як плагін до BO (дуже корисно). Робота з командного рядка.

Shadow IM Sniffer

Розроблювач: SAFETY-LAB.

Опис: Shadow IM Sniffer – програма перехоплення повідомлень IM клієнтів у межах ЛОМ. Підтримка різних IM клієнтів і різних форматів повідомлень – робить сніффер унікальним продуктом. Всі перехоплені повідомлення

зберігаються в бази SQL (MSSQL, MySQL). Можливість роботи сніффера як у звичайному режимі так і в режимі сервісу.

Існують ще безліч снiфферів, таких як Natas, NetXRay, CooperSniffer, LanExplorer, Net Analyzer і т.д.

Unix'ові сніффери

Всі сніффери даного огляду можна знайти на packetstorm.securify.com.

linsniffer

Це простий сніффер для перехоплення логінів/паролів. Стандартна компіляція (gcc -o linsniffer linsniffer.c). Логи пише в tcp.log.

linux_sniffer

Linux_sniffer потрібно тоді, коли ви хочете детально вивчити мережа. Стандартна компіляція. Видає додаткові дані, типу isn, ack, syn, echo_request (ping) і т.д.

Sniffit

Sniffit – просунута модель сніффера написана Brecht Claerhout. Install (потрібна libcap):

#./configure

#make

Тепер запускаємо сніффер:

#./sniffit

usage: ./sniffit [-xdabvn] [-P proto] [-A char] [-p port] [(– r|-R) recordfile]

[-l sniflen] [-L logparam] [-F snifdevice] [-M plugin]

[-D tty] (-t<Target IP> | -s<Source IP>) | (– i|-I) | -c<config file>]

Plugins Available:

0 ---i Dummy Plugin

1 ---i DNS Plugin

Як бачите, сніффіт підтримує безліч опцій. Можна використовувати сніффер в інтерактивному режимі. Сніффіт хоч і досить корисна програма, але в

Sniffit великі проблеми із захистом. Для Sniffit'a уже вийшли ремоутний рут і дос для Linux. Не кожний сніффер собі таке дозволяє.

HUNT

Він дуже простий в використанні, підтримує багато функцій і на даний момент не має проблем з безпекою. Плюс не особливо вимогливий до бібліотек (як, наприклад, linsniffer і Linux_sniffer). Він може в реальному часі перехоплювати поточні з'єднання й під чисту "дампить" з віддаленого термінала. Рекомендую всім для посиленого використання.

Install:

#make

Run:

#hunt -i [interface]

READSMB

Сніффер READSMB вирізаний з LophtCrack і портирован під Unix (як не дивно). Readsmb перехоплює SMB пакети.

TCPDUMP

tcpdump – досить відомий аналізатор пакетів. Вимагає бібліотеку Libpcap.

Install:

#./configure

#make

Тепер запускаємо її:

#tcpdump

tcpdump: listening on ppp0

Всі твої коннекти виводить на термінал. От приклад виводу на пінг ftp.technotronic.com:

02:03:08.918959 195.170.212.151.1039 > 195.170.212.77.domain: 60946+ A? ftp.technotronic.com. (38)

02:03:09.456780 195.170.212.77.domain > 195.170.212.151.1039: 60946* 1/3/3 (165)

02:03:09.459421 195.170.212.151 > 209.100.46.7: icmp: echo request

02:03:09.996780 209.100.46.7 > 195.170.212.151: icmp: echo reply

02:03:10.456864 195.170.212.151 > 209.100.46.7: icmp: echo request

02:03:10.906779 209.100.46.7 > 195.170.212.151: icmp: echo reply

02:03:11.456846 195.170.212.151 > 209.100.46.7: icmp: echo request

02:03:11.966786 209.100.46.7 > 195.170.212.151: icmp: echo reply

Загалом, сніффер корисний для налагодження мереж, знаходження несправностей і т.д.

Dsniff

Dsniff вимагає libpcap, ibnet, libnids і OpenSSH. Записує тільки уведені команди, що дуже зручно. От приклад балки коннекта на unix-shells.com:

02/18/01 03:58:04 tcp my.ip.1501 -> handi 253-158-170.arcor-ip.net.23

(telnet)

stalsen

asdqwe123

ls

pwd

who

last

exit

От dsniff перехопив логін з паролем (stalsen/asdqwe123).

Install:

#./configure

#make

#make install

Захист від сніфферів

Самий вірний спосіб захисту від сніфферів – використовувати шифрування (SSH, Kerberous, VPN, S/Key, S/MIME, SHTTP, SSL і т.д.). Ну, а якщо не

полювання відмовлятися від plain text служб і встановлення додаткових пакетів? Тоді настав час використовувати антисніфферські пакети...

AntiSniff for Windows

Цей продукт випустила відома група Lopht. Це був перший продукт у своєму роді. AntiSniff, як сказано в описі: "AntiSniff is a Graphical User Interface (GUI) driven tool for detecting promiscuous Network Interface Cards (NICs) on your local network segment". Загалом, ловить в promisc режимі. Підтримує величезну кількість тестів (DNS test, ARP test, Ping Test, ICMP Time Delta Test, Echo Test, PingDrop test). Можна сканувати як одну машину, так і сітку. Тут є підтримка логів. AntiSniff працює під Windows. Але незабаром з'явився сніффер за назвою AntiAntiSniffer, написаний Майком Перрі (Mike Perry) (знайти його можна за адресою www.void.ru/news/9908/snoof.txt).Він заснований на LinSniffer (розглянутий далі).

Unix sniffer detect

Сніффер можна виявити командою:

#ifconfig -a

lo Link encap:Local Loopback

inet addr:127.0.0.1 Mask:255.0.0.0

UP LOOPBACK RUNNING MTU:3924 Metric:1

RX packets:2373 errors:0 dropped:0 overruns:0 frame:0

TX packets:2373 errors:0 dropped:0 overruns:0 carrier:0

collisions:0 txqueuelen:0

ppp0 Link encap: Point-to-Point Protocol

inet addr:195.170.y.x P-t-P:195.170.y.x Mask:255.255.255.255

UP POINTOPOINT PROMISC RUNNING NOARP MULTICAST MTU:1500 Metric:1

RX packets:3281 errors:74 dropped:0 overruns:0 frame:74

TX packets:3398 errors:0 dropped:0 overruns:0 carrier:0

collisions:0 txqueuelen:10

Як бачите, інтерфейс ppp0 знаходиться в PROMISC mode. Або оператор завантажив сніффер для перевірки мережі, або вас уже взломали. Але пам’ятайте, що ifconfig можна спокійно підмінити, тому використовуйте tripwire для виявлення змін і всілякі програми для перевірки на сніффи.

AntiSniff for Unix

Працює на BSD, Solaris і Linux. Підтримує ping/icmp time test, arp test, echo test, dns test, etherping test, загалом, аналог AntiSniff'a для Win, тільки для Unix.

Install:

#make linux-all

Sentinel

Теж корисна програма для вилову сніфферів. Підтримує безліч тестів, проста у використанні.

Install : #make

#./sentinel

./sentinel [method] [-t <target ip>] [options]

Methods:

[-a ARP test]

[-d DNS test]

[-i ICMP Ping Latency test]

[-e ICMP Etherping test]

Options:

[-f < non-existant host>]

[-v Show version and exit]

[-n <number of packets/seconds>]

[-I <device>]

2.2 Обґрунтування вибору засобів для побудови системи кібербезпеки 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 випускну кваліфікаційну роботу за першим (бакалаврським) рівнем вищої освіти, реалізації підлягає програмне забезпечення, яке призначено для системи кібербезпеки для виявлення аномалій телекомунікаційного трафіку на основі спектрально-часового аналізу.

В процесі розробки випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти 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 кібербезпеки контролю роботи технологічного обладнання на виробництві в автоматизованому режимі. Розробити функціональну та структурну схеми системи;

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

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

д) розробити рекомендації по організаційних та методичних заходах, які забезпечать впровадження системи кібербезпеки 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

Структура методу динамічного виявлення аномалій в інтенсивностях потоків телекомунікаційних даних

Математично обґрунтованими видами аналізу часових рядів є дослідження сигналу в часовій, спектральній і спектрально-часовій областях.

Слід зазначити, що досліджуваному телекомунікаційному сигналу властиво «коливальне поводження» через його формування на основі однозначно непередбачених дій користувачів або програм. Для досліджень такого роду залежностей у теорії цифрової обробки сигналів звичайно використовуються більше інформативні при розкритті частотної картини спектральні й спектрально-часові алгоритми.

Покажемо перспективи застосування методів виділення частотних складових для завдання виявлення аномалій в інтенсивностях потоків даних:

– виконання еталонного порівняння;
– обчислення частотного подання переданомального часового інтервалу;
– визначення динаміка зміни частот;
– побудова частотної моделі сигналу;

і ін.

Багато підходів є аналогами традиційному дослідженню сигналу в часовій області, використовують його принципи як для самої обробки, так і для підготовки даних до аналізу, але дозволяють інакше глянути на об'єкт дослідження.

Розглянемо основи спектрального й спектрально-часового аналізів, включених як базові алгоритми в метод виявлення аномалій телекомунікаційних даних. Робиться вивід про те, що особливості застосування зазначених методів у

конкретному додатку вимагають додаткового дослідження. У тому числі досить бажаним є зниження обчислювальної складності алгоритмів, тому що в порівнянні із традиційним аналізом у часовій області вони є більше трудомісткими через використання подань сигналу більшої розмірності, що негативно позначається на привабливості їхнього використання.

Опишемо структуру методу динамічного виявлення аномалій в інтенсивностях потоків телекомунікаційних даних, що включає наступні основні етапи:

1. Збір даних.

2. «Ручна» класифікація частини даних на предмет присутності нормального (типового) або аномального трафіку.

3. Попередній Фур'є-аналіз типового трафіку.

4. Виконання вейвлет-розкладання типового трафіку по обраних функціях для оцінки його спектрального-часового подання.

5. Обробка вейвлет-подання аномального трафіку.

а) визначення моментів різкої зміни характеру поводження рядів телекомунікаційних даних по рівневих і частотних характеристиках;

б) формування баз шаблонів вейвлет-образів:

– бази вейвлет-образів для характеру розвитку трафіку на тлі критичних ситуацій;

– бази вейвлет-образів для характеру розвитку трафіку до критичних моментів (до початку зареєстрованих проблем у роботі мережі);

– збереження відповідностей між даними баз;

в) визначення частотних показників для провокуючий розвиток аномалії подій, у тому числі аналіз енергетичних спектрів послідовностей до критичних моментів (спектра могутньості E_F і скалограми E_W) на предмет перерозподілу енергії між частотами;

6. Верифікація прогнозу характеристик трафіку з реальними значеннями (на основі попередньо зібраних даних поза реальним режимом часу).

7. При надходженні нових даних:

а) зіставлення їх вейвлет-подань із шаблонами сформованих баз даних за допомогою 2-D кореляції й формування рішення про наявність передкритичної або критичної ситуації.

б) виявлення зв'язків між аномаліями, що відбулися в різних точках спостереження.

У результаті виконання етапів з першого по шостий включно маємо:

– алгоритм розпізнавання позаштатної ситуації;

– початкові бази вейвлет-шаблонів для позаштатних ситуацій і попередніх за часом інтервалів;

– модель прогнозування характеру розвитку трафіку і як наслідок апріорне визначення можливих позаштатних ситуацій і їхня класифікація.

Опишемо застосування спектральних і спектрально-часових алгоритмів, що є основними інструментами в розв'язуваному завданні динамічного виявлення аномалій у часових рядах інтенсивностей телекомунікаційних даних:

– аналіз внесків частот визначає формування моделей типового трафіку й окремих ділянок з особливостями;

– картини вейвлет-коефіцієнтів аномалій і попередніх їм ділянок сигналу формують бази даних шаблонів для характеру розвитку трафіку на тлі критичних ситуацій і до критичних моментів;

– побудовані на основі обчислених коефіцієнтів енергетичні спектри й інші залежності дозволяють відслідковувати розподіл (або перерозподіл) енергії між частотами й визначати сховану природу окремих ділянок сигналу.

Розкриємо етапи нагромадження інформації, аналізу готових наборів, формування баз даних шаблонів, верифікації, виявлення аномалій у потоках даних, аналізу взаємозв'язків аномалій у різних потоках.

Дослідження спектрального й спектрально-часового перетворень

Детально проаналізуємо алгоритми дискретного перетворення Фур'є (ДПФ), дискретного перетворення Хартлі (ДПХ), швидкого перетворення Фур'є (БПФ) і швидкого перетворення Хартлі (БПХ).

Обґрунтуємо можливість заміни широко використовуваного ДПФ на менш відоме, але більше ефективне ДПХ.

У результаті отримані математичні моделі оптимізації короткочасного аналізу (за участю часового «вікна» сканування) для прямих ПФ і ПХ зсувної послідовності:

$$F_k^l = \left(F_k^{l-1} - x_{l-1} + x_{N+l-1}\right)W^{-k} = \left(F_k^{l-1} - x_{l-1} + x_{N+l-1}\right)e^{j\frac{2\pi}{N}k},\ k = \overline{0, N-1},$$

$$H_{k\cos}^l = \left(H_{k\cos}^{l-1} - x_{l-1}\right)\times\left(\cos\frac{2\pi}{N}k - \sin\frac{2\pi}{N}k\right) + x_{N+l-1}\cos\frac{2\pi}{N}k,\ k = \overline{0, N-1},$$

$$H_{k\sin}^l = 0,\ k = 0,$$

$$H_{k\sin}^l = H_{k\sin}^{l-1}\times\left(\cos\frac{2\pi}{N}k + \sin\frac{2\pi}{N}k\right) - x_{N+l-1}\sin\frac{2\pi}{N}k,\ k = \overline{1, N-1},$$

де F_i – значення i-ого відліку спектра Фур'є, обчисленого на l-ому етапі, H_i – значення i-ого відліку синусної або косинусної компоненти спектра Хартлі, обчисленого на l-ому етапі, x_t – значення i-ого відліку сигналу, N – розмір «вікна».

Для перетворення Хартлі обґрунтована необхідність роздільного обчислення косинусних і синусних складових з наступним їхнім підсумовуванням.

Перехід від ДПХ до ДПФ варто виконувати за формулою:

$$F_k = H_k\times\left(\frac{1}{2} - j\frac{1}{2}\right) + H_{N-k}\times\left(\frac{1}{2} + j\frac{1}{2}\right) = \frac{1}{2}\times\left(H_k + H_{N-k}\right) + j\frac{1}{2}\times\left(H_{N-k} - H_k\right),$$
$$k = \overline{0, N-1}$$

Запропоновані алгоритми обчислення значень відрахунків спектрів на основі попередніх значень надалі будуть називатися оптимізованими.

Отримані практичні результати підтвердили ефективність запропонованих методів перетворення й у своєму загальному виді збіглися з теоретичними, незважаючи на присутні відхилення, внесені специфікою програмної реалізації. Відповідно до отриманих значень середніх часів виконання короткочасного спектрального аналізу в межах одного «вікна» можна зробити наступний вивід: для скорочення часу виконання короткочасного Фур'є-аналізу треба перше «вікно» обчислювати за допомогою БПХ із наступним перетворенням обчислених значень у результат ПФ, а наступні «вікна» на основі запропонованого методу для ПФ.

Переваги запропонованого методу:

– висока швидкість обчислень;

– лінійна залежність швидкості обчислень від розміру «вікна».

Недоліки запропонованого методу:

– наявність швидко зростаючої погрішності обчислень ПХ після другої ітерації алгоритму перерахунку спектра.

Для усунення недоліку запропонованого методу необхідно:

– застосування розширених форматів подання даних;

– при досягненні критичного значення погрішності зробити перерахування спектра по стандартному алгоритмі.

Далі частина роздягнула присвятимо спектрально-часовим вейвлет-перетворенням (ВП). Розглянуто принципи різних варіацій дискретизованого перетворення, вплив базису на результат вейвлет-розкладання й дані рекомендації з можливого використання вейвлетов.

Докладно розглянуті:

1) дискретне перетворення зі зміною масштабів і зрушень за значенням, рівному часової локалізації вейвлету;

2) дискретний діадний кратномасштабний аналіз;

3) дискретизоване розкладання.

Виконано аналіз виникаючих крайових ефектів спектрально-часових картин ВП на прикладах розгляду кутів впливу елементів аналізованої послідовності на вейвлет-коефіцієнти й кутів вірогідності (правдоподібності) у сітках перетворень. Усередині кута правдоподібності, як відомо, крайові ефекти проявлятися не можуть, оскільки кожна особливість, у тому числі й виникаюча на краях аналізованого сигналу, робить лише локальний вплив на вейвлет-спектрограмму. Скоректовано вид кутів впливу й правдоподібності для зсувної послідовності.

З позиції інформативності по кількості уточнюючих коефіцієнтів кращим є випадок 3), потім 2), потім 1).

З погляду збільшення області достовірних значень кращим є випадок 1), потім 3), потім 2).

У порядку зростання обчислювальної складності перетворення розташовуються в порядку 1), 2), 3).

Для завдання розпізнавання аномалії перші два критерії є вирішальними, причому по жодному з них обране перетворення не повинне бути гіршим.

Таким є дискретизоване перетворення, хоча його надмірність очевидна й підтверджується третім критерієм – обчислювальною складністю.

Можна припустити, що оптимальним було б перетворення, «находящееся» між дискретним діадним і дискретизованим, причому саме перше повинне бути взяте за основу, тому що має добре оптимізовані по організації обчислень і використовуваної пам'яті швидкі алгоритми статичного обчислення сітки коефіцієнтів, і розширено для наділення перевагами дискретизованого розкладання. Результат може бути аналогічний використанню фреймів, тому що перетворення на основі фреймів має велика подібність із дискретизованим розкладанням з тією різницею, що зміна масштабу відбувається по ступенях «двійки».

Розроблено схему виконання дискретизованого за часом і діадного по масштабах розкладання на основі швидкого алгоритму діадного

кратномасштабного аналізу. Відзначимо, що в цей час існують алгоритми, оптимизуючі стандартну схему швидкого діадного розкладання, наприклад на основі нестандартного матричного множення, тобто пропонований у роботі метод розрахунку вейвлет-коефіцієнтів може бути додатково оптимізований по обчислювальних витратах.

Виконано оптимізацію обчислень сіток дискретизованого розкладання за принципом фреймів і покладених у його основу алгоритмів діадного кратномасштабного аналізу при зрушенні послідовності.

Загальна формула для розрахунку кількості *num* коефіцієнтів, що обчислюються, діадного розкладання при зрушенні послідовності:

$$num_{\Delta x=\overline{0,1}} = l-1,$$

$$num_{\Delta x=\overline{2^t,2^{t+1}-1}} = l+t-1-\sum_{s=1}^{t} 2^{\log_2 l-s} = l+t-1-2^{\log_2 l}\cdot\sum_{s=1}^{t} 2^{-s},$$

де l – довжина аналізованої послідовності, $t = 1, \log_2(l-1)$, t – ціле, значення $\log_2(l-1)$ визначає кількість рівнів вейвлет-розкладання.

При зрушенні послідовності на:

$$\Delta x \geq \frac{l}{2},$$

кількість вейвлет-коефіцієнтів, значення яких необхідно обчислити дорівнює:

$$num_{\Delta x\geq\frac{l}{2}} = \log_2 l,$$

а кількість коефіцієнтів, що беруть участь в оптимізації розрахунків:

$$koef\_opt_{\Delta x\geq\frac{l}{2}} = \sum_{level=1}^{\log_2 l-1}\left(2^{level}-1\right)\cdot 2^{\log_2 l-level-1} = l-1-\log_2 l,$$

де *level* – рівень вейвлет-розкладання.

Виграш при подібній оптимізації розрахунків на прикладі 16-елементного вектора для аналізу продемонстрований на рисунку 3.1. Нижній горизонтальний і убутний графіки відповідають кількості вейвлет-коефіцієнтів, які обчислюються при зрушенні часової послідовності на Ах у випадку стандартного підходу з повним обчисленням вейвлет-образа на кожному кроці й у випадку оптимізованого алгоритму, відповідно. Верхній горизонтальний і убутний

графіки аналогічні попередньому, але враховують ще необхідність обчислення на кожному кроці відліку апроксимації. Пунктирні лінії вказують тренд для числа обчислень при оптимізованому підході.

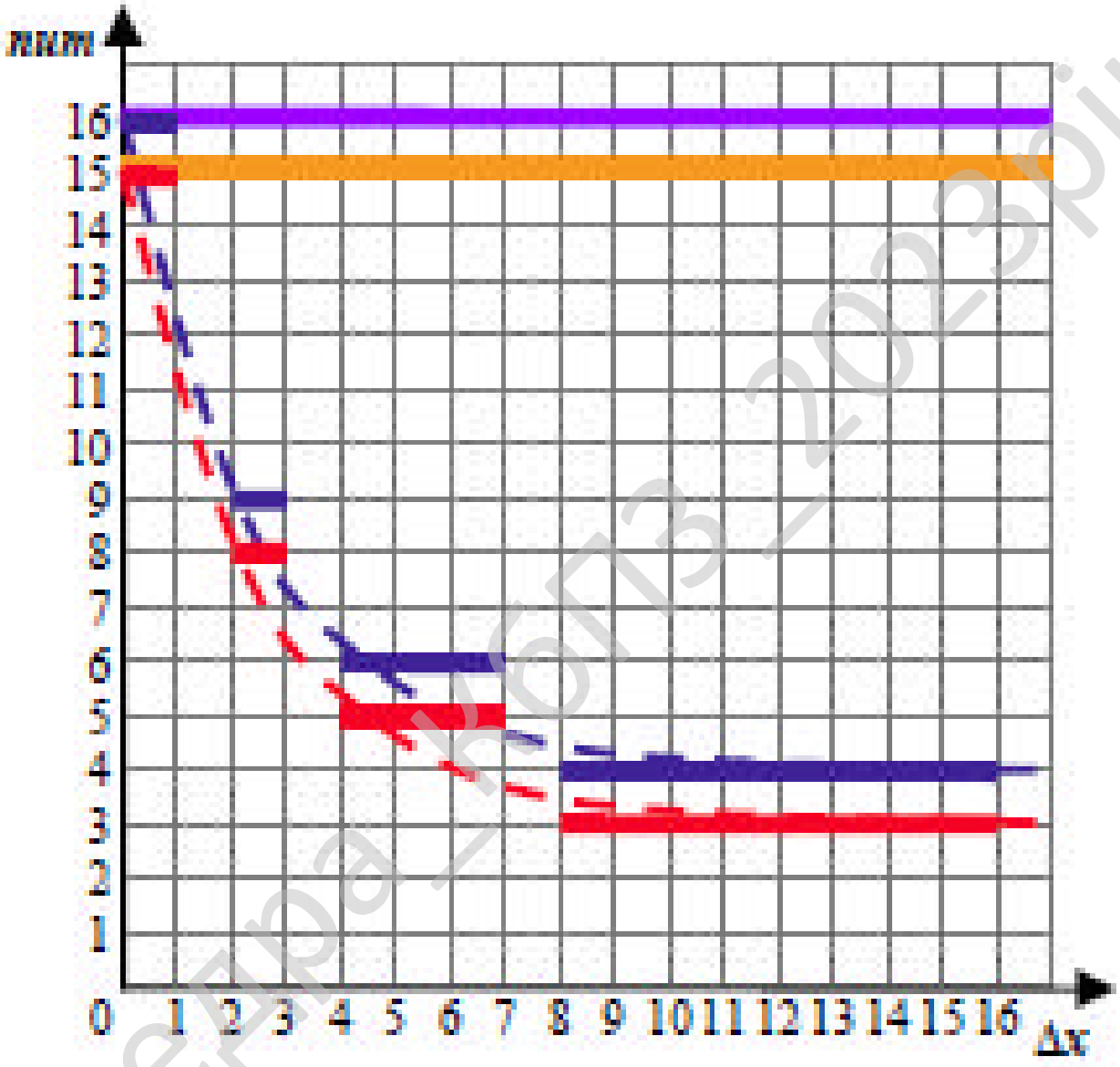

Рисунок 3.1 – Ілюстрація результатів дослідження

При виконанні дискретизованого ВП зі зміною масштабів по ступенях «двійки» обґрунтований вибір позицій для перерахування коефіцієнтів з обліком відсутності або наявності розгляду кута правдоподібності, а також його виду.

Кількість вейвлет-коефіцієнтів, значення яких необхідно обчислити при зрушенні послідовності:

$$пит = \log_2 l.$$

Розроблено підхід до формування довірчої області на картині вейвлет-коефіцієнтів. Завдання було вирішено послідовним виконанням наступних етапів:

– вибір алгоритму пророкування значень майбутніх елементів часового ряду;

– розрахунок довірчого інтервалу для передвіщених значень;

– виконання верифікації;

– установлення зв'язку довірчого інтервалу із границею кута правдоподібності.

– розширення кута правдоподібності за допомогою введення довірчої області в картину вейвлет-коефіцієнтів.

Запропоновано підхід для мінімізації розкиду значень вейвлет-коефіцієнтів у розрахованої довірчої вейвлет-області на основі введення нестандартної внутрішньої границі довірчої часової області.

На впевненість влучення вейвлет-коефіцієнтів у розрахований діапазон (вейвлет-упевненість) впливають значення відрахунків вейвлету, тому що їх можна інтерпретувати в рамках справжнього етапу як вагові коефіцієнти, відповідно до яких приймаються в розгляд відрахунки часового ряду, у тому числі неіснуючі. Таким чином, упевненість із урахуванням застосування процедури децимації до елементів послідовності відповідно до досліджуваного масштабу розраховується в такий спосіб:

$$p_{w_{j,k}} = \frac{1}{2M-1}\sum_{m=0}^{2M-1} p_{w_{j+1,k+\frac{m\cdot l}{2^{j+1}}}} \cdot |g_m|,\ j = \overline{0,\log_2 l - 1},\ k = \overline{0,l-1},$$

де j – рівень розкладання, що змінюється по ступенях 2, l – довжина аналізованого часового ряду); до – параметр, що визначає зрушення вейвлету; $p_{wj,к}$ – вейвлет-упевненість для позиції (j,к), з якої щире значення вейвлет-коефіцієнта потрапить у розрахований довірчий діапазон $\Delta w = [w^{min}, w^{max}]$;

(2M – 1) – довжина області визначення вейвлету (M визначає число нульових моментів), g – значення вейвлет-функції, що збігаються з коефіцієнтами фільтра синтезу верхніх частот без застосування процедури нормування.

Якщо:

$$k + \frac{m \cdot l}{2^{j+1}} > l - 1$$

,

то замість:

$$p_{w_{j+1,s+\frac{k \cdot l}{2^{j+1}}}}$$

,

у формулі буде використовуватися впевненість для першого впливаючого елемента:

$$w_{j+1,k+\frac{m \cdot l}{2^{j+1}}}$$

,

на:

$$x\left(t_{0+\left(k+\frac{m \cdot l}{2^{j+1}}-(l-1)\right)}\right) = x\left(t_{k+\frac{m \cdot l}{2^{j+1}}-l+1}\right)$$

,

з неіснуючої частини часового ряду, де t_0 – момент останнього виконаного виміру.

Діапазони змін вейвлет-коефіцієнтів розраховуються на основі границь довірчих інтервалів $\Delta x = [xmi^n, x^{max}]$ для передвіщених відрахунків часового ряду усередині внутрішньої границі довірчої часової області.

Наведено приклад формування довірчої області коефіцієнтів розкладання визначенням осередків нульового, першого й другого рівнів, значення яких потраплять у певний діапазон із упевненістю $p_{wave\_mln} > 0{,}8$. Їхнє виділення показане на рисунку 3.2 темним заливанням.

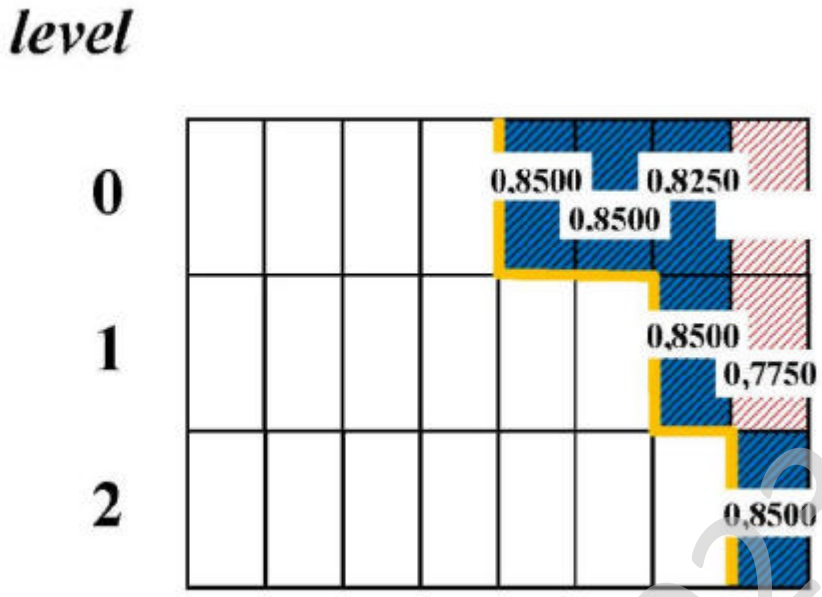

Рисунок 3.2 – Формування довірчої області коефіцієнтів розкладання

Для використання в аналізі довірчої вейвлет-області найбільш підходящим є формування декількох її подань, заповнених, наприклад, значеннями:

– верхньої границі;

– нижньої границі;

– розрахованими на основі передвіщених значень апроксимованої часової послідовності;

– випадково розподіленими між верхньою й нижньою границями.

Комбінація таких подань довірчої вейвлет-області дозволить найбільше вірно судити про поводження вейвлет-коефіцієнтів за межами кута правдоподібності.

Для одержання картини вейвлет-коефіцієнтів, придатної для виконання операцій розпізнавання, необхідна певна довжина вихідної послідовності. Мала довжина послідовності спричиняється пропорційну кількість вейвлет-коефіцієнтів, що робить картину розкладання розбитої на відповідну кількість прямокутних областей. Елементи такої картини занадто грубі й непридатні для подальшого розпізнавання.

До вихідної послідовності пропонується застосувати алгоритм, що збільшує число елементів ряду. Проаналізовано результати використання спектральних і часових алгоритмів інтерполяції.

3.2 Розробка структурної схеми

Структурна схема розробленої системи зображена на рисунку 3.3. На ній показано структуру системи, яка складається з наступних блоків:

1. Система моніторингу мережі, яка включає в себе Базу даних результатів моніторингу мережі.

2. Система виявлення аномалій, яка включає в себе:

– Модуль взаємодії з базою даних моніторингу.

– Модуль частотного аналізу.

– Модуль обробки результатів.

– Інтерфейс адміністратора.

– Базу даних виборок.

– Базу даних шаблонів нормального поводження трафіку.

Наведено результати виконання окремих етапів. Інтерес представляє ієрархічна організація баз шаблонів. Порівняння відбувається за принципом поступового наближення зі знаходження статистичної залежності чорно-білих зображень на основі кореляційного методу. При цьому розмірність еталона й досліджуваного подання по вертикалі збігаються. Рух еталона уздовж вейвлет-образу сигналу відбувається в горизонтальному напрямку у випадку стаціонарного аналізу. При динамічному виявленні особливостей еталон і сигнал зіставляються правими границями.

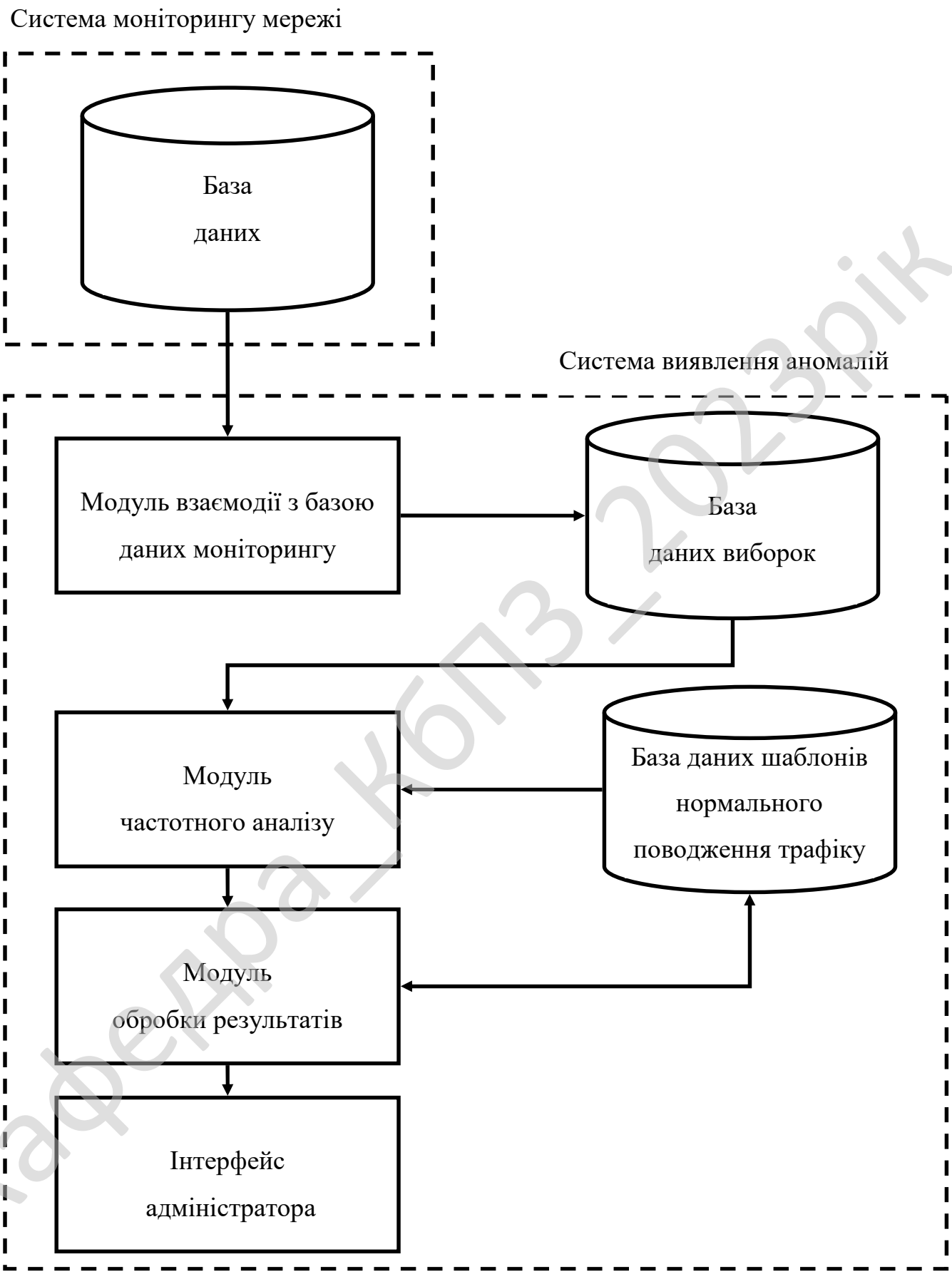

Рисунок 3.3 – Структурна схема системи

Поява аномалій у різних точках спостереження протягом невеликого проміжку часу говорить про можливий взаємозв'язок подій. Об'єкт, що має більш ранній за часом момент виникнення аномалії може бути джерелом нестандартного поводження ряду даних. Досліджуючи виникнення аномалій на різних пристроях, можливим є побудова дерева аномалій, що веде від джерела через посередників різних рівнів до приймачів і навпаки: від приймача до джерела.

3.3 Розробка функціональної схеми

Функціональна схема розробленої системи зображена на рисунку 3.4. Так, як функціональна схема є більш подобним описом функціональних можливостей структурної схеми, то вона буде представляти собою, більш детальний варіант структурної схеми.

З рисунку видно, що розроблена система складається з наступних частин:

– Блок визначення виду атаки.

– Блок моніторингу мережі.

– Блок аналізу мережної статистики.

– Блок визначення топології мережі.

– Блок виявлення аномального поводження трафіку.

– Блок зберігання результатів.

Блок визначення виду атаки

Блок визначення виду атаки:

– Атака ARP-spoofing на таблицю mac-адрес комутаторів.

– Широкомовний шторм.

– Додатки, що роблять інтенсивне широкомовне розсилання, наприклад: широкомовні чати й мережні ігри.

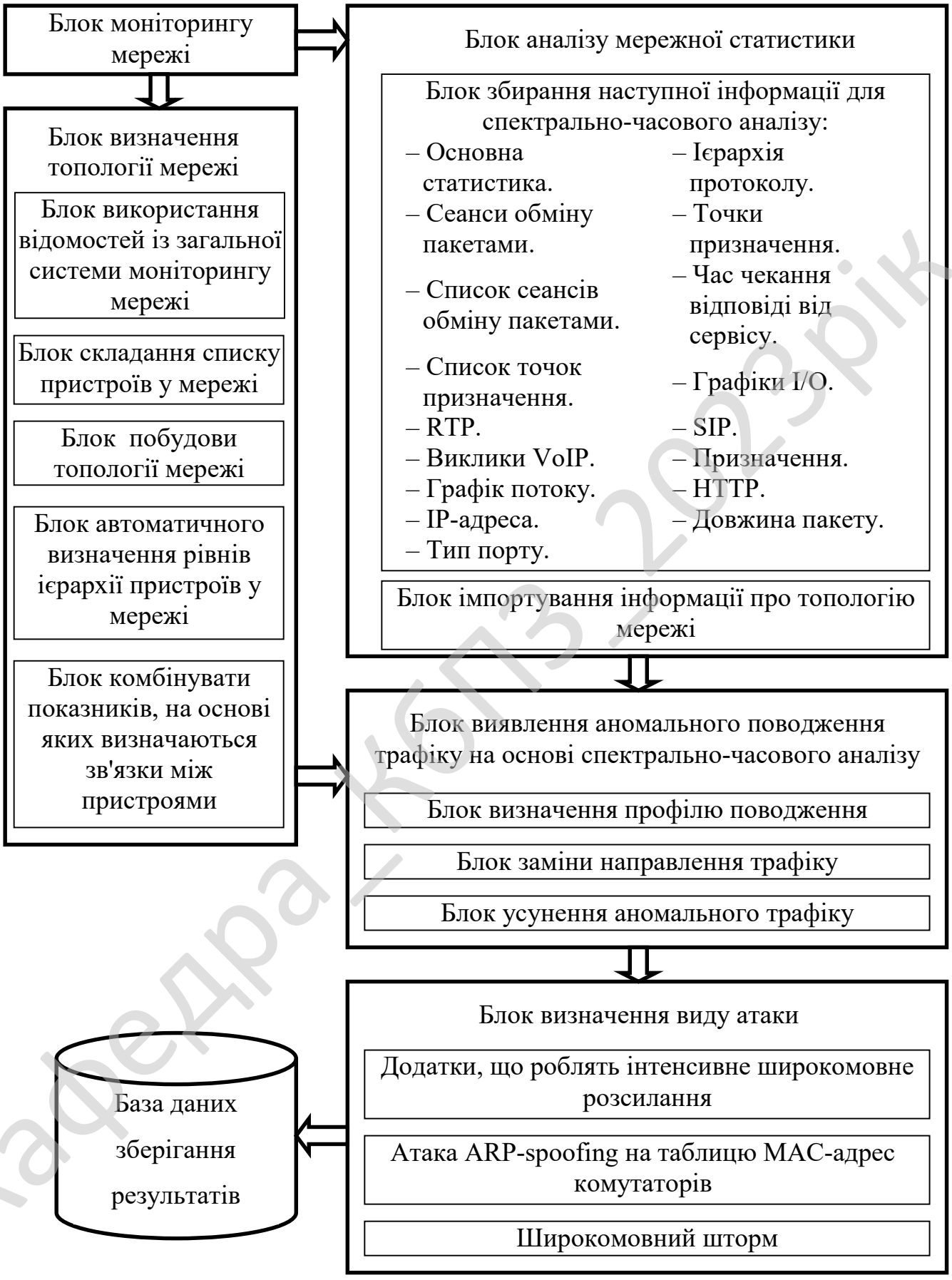

Рисунок 3.4 – Функціональна схема системи

ARP-spoofing

ARP-spoofing – техніка атаки в Ethernet мережах, що дозволяє перехоплювати трафік між хостами. Заснована на використанні протоколу ARP.

При використанні в розподіленій обчислювальній системи (РВМ) алгоритмів віддаленого пошуку існує можливість здійснення в такій мережі типової віддаленої атаки «помилковий об'єкт РВМ». Аналіз безпеки протоколу ARP показує, що, перехопивши на атакуючому хості усередині даного сегмента мережі широкомовний ARP-запит, можна послати помилкову ARP-відповідь, у якій оголосити себе шуканим хостом (наприклад, маршрутизатором), і надалі активно контролювати мережний трафік дезінформованного хосту, впливаючи на нього за схемою «помилковий об'єкт РВМ».

Протокол ARP призначений для перетворення IP-адрес в MAC-адреси. Найчастіше мова йде перетворенні в адреси Ethernet, але ARP використовується й у мережах інших технологій: Token Ring, FDDI і інших.

Алгоритм роботи ARP

Протокол може використовуватися в наступних випадках:

1. Хост А хоче передати IP-пакет вузлу В, що перебуває з ним в одній мережі.

2. Хост А хоче передати IP-пакет вузлу В, що перебуває з ним у різних мережах, і користується для цього послугами маршрутизатора R.

У кожному із цих випадку вузлом А буде використовуватися протокол ARP, тільки в першому випадку для визначення MAC-адреси вузла В, а в другому – для визначення MAC-адреси маршрутизатора R. В останньому випадку пакет буде переданий маршрутизатору для подальшої ретрансляції.

Далі для простоти розглядається перший випадок, коли інформацією обмінюються вузли, що перебувають безпосередньо в одній мережі. (Випадок коли пакет адресований вузлу, який знаходиться за маршрутизатором, відрізняється тільки тим, що в пакетах переданих після того як ARP-перетворення завершене, використовується IP-адреса одержувача, але MAC-адреса маршрутизатора, а не одержувача.)

Проблеми ARP

Протокол ARP є абсолютно незахищеним. Він не має ніякого способу перевірки дійсності пакетів: як запитів, так і відповідей. Ситуація стає ще більш складною, коли може використовуватися мимовільний ARP (gratuitous ARP).

Мимовільний ARP – таке поводження ARP, коли ARP-відповідь надсилається, коли в цьому (з погляду одержувача) немає особою необхідності. Мимовільна ARP-відповідь це пакет-відповідь ARP, присланий без запиту. Він застосовується для визначення конфліктів IP-адрес у мережі: як тільки станція одержує адресу по DHCP або адреса привласнюється вручну, розсилається ARP-відповідь gratuitous ARP.

Мимовільний ARP може бути корисний у наступних випадках:

– Відновлення ARP-таблиць, зокрема, у кластерних системах.

– Інформування комутаторів.

– Повідомлення про включення мережного інтерфейсу.

Незважаючи на ефективність мимовільного ARP, він є особливо небезпечним, оскільки з його допомогою можна запевнити віддалений вузол у тому, що MAC-адреса якої-небудь системи, що перебуває з нею в одній мережі, змінилася й указати, яка адреса використовується тепер.

До виконання ARP-spoofing'a в ARP-таблиці вузлів A і B існують записи з IP– і MAC-адресами один одного. Обмін інформацією виробляється безпосередньо між вузлами A і B.

У ході виконання ARP-spoofing'a комп'ютер **C**, що виконує атаку, відправляє ARP-відповіді (без одержання запитів):

– вузлу A: з IP-адресою вузла B і MAC-адресою вузла C;

– вузлу B: з IP-адресою вузла A і MAC-адресою вузла C.

У силу того що комп'ютери підтримують мимовільний ARP (gratuitous ARP), вони модифікують власні ARP-таблиці й поміщають туди записи, де замість справжніх MAC-адрес комп'ютерів A і B коштує MAC-адреса комп'ютера C.

Після того як атака виконана, коли комп'ютер А хоче передати пакет комп'ютеру В, він знаходить в ARP-таблиці запис (він відповідає комп'ютеру С) і визначає з її MAC-адресу одержувача. Відправлений по цьому MAC-адресу пакет приходить комп'ютеру С замість одержувача. Комп'ютер С потім ретранслює пакет тому, кому він дійсно адресований – тобто комп'ютеру В.

Широкомовний шторм

Широкомовний шторм – лавина (сплеск) широкомовних акетів (на другому рівні моделі OSI – кадрів). Розмноження некоректно сформованих широкомовних повідомлень у кожному вузлі приводить до експонентного росту їхнього числа й паралізує роботу мережі. Звичайно такі пакети використовуються мережними сервісами для оповіщення станцій про свою присутність. Вважається нормальним, якщо широкомовні пакети становлять не більше 10% від загального числа пакетів у мережі.

Також досить часто до шторму приводять кільця в мережі при некоректному налаштуванні протоколу Spanning Tree, оскільки в заголовку пакетів Ethernet немає інформації про час життя кадру, як, наприклад, у пакетів IP. Крім цього широкомовний шторм застосовується (навмисно) зломщиками.

Відповідно до галузевого стандарту де-факто число широкомовних і багатоадресних кадрів у мережі не повинне перевищувати 8-10% від загального числа кадрів.

Широкомовний кадр – це кадр, адресований всім станціям у домені мережі. Багатоадресний кадр – це кадр, адресований групі станцій у домені мережі. Оскільки широкомовний кадр адресований всім станціям, то, одержавши його, станції повинні перервати свою роботу й обробити такий кадр. Це сповільнює роботу всієї мережі.

Якщо відношення числа широкомовних кадрів до загального числа кадрів більше 10%, то такий ефект називається "широкомовним штормом".

Широкомовний шторм може бути наслідком дефектів устаткування або неправильного налаштування параметрів активного встаткування. Найчастіше це явище спостерігається в розподілених мережах NetWare, побудованих на основі комутаторів, або коли дані між сегментами або доменами мережі можуть

передаватися більш ніж по одному потенційному шляху. Якщо один з комутаторів такої мережі не підтримує протокол Spanning Tree (звичайно IEEE 802.1d) або останній неправильно настроєний або збоїть, то в мережі починається некерована циркуляція широкомовних кадрів.

Виявлення "широкомовного шторму" є не настільки тривіальним завданням, як це може здатися на перший погляд. Для його виявлення недостатньо взяти загальне число широкомовних кадрів і поділити його на загальне число кадрів, що пройшли по мережі.

Для цього ви повинні визначити: яку частку становлять широкомовні кадри в кожний інтервал часу (наприклад, за одну мінуту) і яка при цьому утилізація каналу зв'язку. Якщо, наприклад, за одну мінуту по мережі пройшло 4 кадри, а 2 з них були широкомовними, то це ще не виходить, що ви спостерігаєте "широкомовний шторм".

Захист від широкомовних штормів (broadcast storm)

Одна з характерних несправностей мережного програмного забезпечення – мимовільна генерація з високою інтенсивністю широкомовних пакетів. Широкомовним штормом вважається ситуація, у якій відсоток широкомовних пакетів перевищує 20% від загальної кількості пакетів у мережі. Звичайний комутатор або міст сліпо передає такі пакети на всі свої порти, як того вимагає його логіка роботи, засмічуючи, таким чином, мережу. Боротьба із широкомовним штормом у мережі, з'єднаної комутаторами, жадає від адміністратора відключення портів, що генерують широкомовні пакети. Маршрутизатор не поширює такі ушкоджені пакети, оскільки в коло його завдань не входить копіювання широкомовних пакетів в усі поєднувані їм мережі. Тому маршрутизатор є прекрасним засобом боротьби із широкомовним штормом, щоправда, якщо мережа розділена на достатню кількість підмереж.

Блок аналізу мережної статистики

Блок збирання наступної інформації:

– Основна статистика (Summary).

– Ієрархія протоколу (Protocol Hierachy).

– Сеанси обміну пакетами (Conversations).

– Точки призначення (Endpoints).

– Графіки I/O (IO Graphs).

– Список сеансів обміну пакетами (Conversation List).

– Список точок призначення (Endpoint List).

– Час чекання відповіді від сервісу (Service Response Time).

– RTP.

– SIP.

– Виклики VoIP (VoIP Calls).

– Призначення (Destination).

– Графік потоку (Flow Graph).

– HTTP.

– IP-адреса (IP address).

– Довжина пакету (Packet Length).

– Тип порту (Port Type).

Розпишемо їх більш детально.

1. Основна статистика. Доступні такі елементи основної статистики, як:

– Властивості захоплених файлів.

– Час захвату.

– Інформація про фільтр захвату.

– Інформація про фільтр відображення.

2. Ієрархія протоколу. Статистика ієрархії протоколу допомагає аналізувати пакети, розбиваючи відображені дані, які належать чинному рівню OSI.

3. Сеанси обміну пакетами. Якщо ви використовуєте протокол TCP/IP або програму, яка працює із цим протоколом, ви маєте побачити чотири активних вкладок для обміну пакетами за допомогою Ethernet, IP, TCP та UDP. «Діалог» між комп'ютерами відображає трафік між двома активними хостами. Номер, зазначений на вкладці після назви протоколу, означає кількість «діалогів» між

хостами. Номер, зазначений на вкладці після назви протоколу, означає кількість «діалогів» між хостами, наприклад, «Ethernet:6».

4. Точки призначення. Точки призначення забезпечують статистику даними про відправку та прийом пакетів. Номер, зазначений на вкладці після назви протоколу, вказує на кількість точок призначення. Наприклад, «Ethernet:6».

5. Графіки I/O. Основний графік може бути отриманий за допомогою команди «IO graphs» (Графіки I/O). Ще декілька графіків можуть бути додані у тому ж вікні на основі фільтрів відображення.

6. Час чекання відповіді від сервісу. 13 протоколів доступні для глибокого аналізу.

7. RTP. RTP (Real-time Transport Protocol, протокол передачі у реальному часі, RFC 3550) – це протокол для передачі звука та відео через IP-мережу. Він працює у початку протоколу дейтаграм користувача (User Datagram Protocol, UDP). Він часто використовується у сукупності з протоколами SIP або H.233, забезпечуючи виконання сигнальних завдань.

8. SIP. SIP (Session Initiation Protocol, протокол встановлення сесії, RFC 3261) – це сигнальний протокол, який оголошує відео– або VoIP-сесії. Він працює разом із протоколом RTP, який використовується для передачі мультимедійних даних.

9. Виклики VoIP.

VoIP (Voice over IP, голосовий зв’язок за допомогою Інтернету) взагалі використовує два типи протоколів:

– сигнальні протоколи, такі, як SIP або H.323

– переносні протоколи, наприклад, RTP

10 Призначення. Відображення усіх IP-адрес призначення мережевих пакетів.

11. Графік потоків. Графіки потоків забезпечує послідовний аналіз TCP-з'єднань. Перші три строки містять оголошення TCP– з'єднання з послідовностями «SYN», «SYN ACK» та «ACK».

12 HTTP. HTTP (Hypertext Transfer Protocol, протокол передачі гіпертексту) – це протокол типу «клієнт-сервер», який використовується для передачі HTML-файлів. HTTP-клієнт (у більшості випадків це web-браузер) відсилає HTTP-запит до web-серверу із полем «URL», який допомагає знайти потрібний файл. Web-сервер відповідає HTTP-пакетом та забезпечує клієнт необхідною web-сторінкою.

Меню «HTTP» містить три підменю:

– «Load Distribution» (Розподіл пакетів).

– «Packet Counter» (Лічильник пакетів).

– «Requests» (Запити).

14 IP-адреса. Відображення IP-адреси джерела або призначення мережевих пакетів.

15. Довжина пакету.

16. Тип порту. Відображення статистики портів TCP або UDP.

Блок визначення топології мережі

Блок визначення топології мережі:

– Блок використання відомостей із загальної системи моніторингу мережі, а не опитування пристрою додатково.

– Блок складання списку пристроїв у мережі, автоматично, ґрунтуючись на дані системи моніторингу.

– Блок побудови топології мережі, за станом на задану дату й відстеження змін у топології протягом часу.

– Блок автоматичного визначення рівнів ієрархії пристроїв у мережі, з виділенням периферійних, проміжних і центральних вузлів;

– Блок побудови топології мережі, незалежно від використовуваної системи моніторингу й програмно-апаратних платформ;

– Блок комбінувати показників, на основі яких визначаються зв'язки між пристроями, і при їхньому обчисленні виконувати перевірку на значимість із використанням статистичних критеріїв.

Блок виявлення аномального поводження трафіку

Блок виявлення аномального поводження трафіку:

– Блок визначення профілю поводження нормального трафіку.

– Блок заміни направлення трафіку.

– Блок усунення аномального трафіку.

При первісному розгортанні рішення по DDoS адміністратор створює профіль поводження нормального трафіку. Цей процес іменується навчанням. Компанія використовує додатки звичайним образом протягом 24 годин протягом одного тижня, і трафік додатка проходить через Детектор аномалій трафіку. У період навчання Детектор аномалій трафіку збирає базову інформацію для розуміння нормальної роботи мережі, куди входять:

– Інтенсивність пакетів для кожного типу пакетів, обмірювана як кількість пакетів у секунду (pps).

– Співвідношення пакетів, наприклад, співвідношення пакетів SYN і пакетів FIN.

– Кількість одночасних TCP-з'єднань, відкритих одним джерелом.

Базова інформація збирається по кожній цільовій адресі хост-пК, цільовій підмережі, вихідній адресі хост-пК і вихідній підмережі.

Після закінчення періоду навчання Детектор аномалій трафіку переводиться в режим моніторингу, а Блок усунення аномального трафіку – у резервний режим готовності. Доти, поки немає атаки, що активно розвивається, вхідний трафік з мережі Інтернет проходить через комутатор без якого-небудь втручання з боку Блоку усунення аномального трафіку. Копія вхідного трафіку посилає для аналізу на Детектор аномалій трафіку через зовнішній аналізатор протоколів (SPAN) або віртуальні списки ACL. Якщо Детектор аномалій трафіку

виявляє аномальне в порівнянні з базовою інформацією поводження трафіку, починається процес усунення:

– Детектор аномалій трафіку направляє в Блок усунення аномального трафіку команду почати процес зміни напрямку.

– Блок усунення аномального трафіку відхиляє (“захоплює”) трафік, адресований на атакуєму IP-адресу, переадресуючи його на самого себе.

– Блок усунення аномального трафіку піддає трафік багатоступінчастому аналізу й застосовує контрзаходи для відділення благонадійних джерел від джерел атаки. Цей процес іменується очищенням або вичищенням.

– Блок усунення аномального трафіку скидає трафік атаки й пересилає благонадійний трафік назад на нормальний маршрут проходження трафіку до мети. Цей процес іменується ін'єкцією.

Детектор аномалій трафіку

Детектор аномалій трафіку – це пасивний пристрій моніторингу, що постійно виявляє ознаки, що вказують на присутність атаки DDoS, спрямованої проти захищеного місця призначення, також іменованого зоною. Це може бути сервер, інтерфейс міжмережного екрана або інтерфейс маршрутизатора. Детектор аномалій трафіку аналізує копії всього вхідного трафіку, адресуємого в захищені зони, через SPAN або відгалуження пасивної мережі. Цей аналіз включає зіставлення поточного поводження трафіку з базовими граничними параметрами, які також іменуються зональною політикою, для виявлення аномального поводження трафіку. Якщо аномальне поводження виявлене й виглядає як можлива атака, Детектор аномалій трафіку через позаполосну управлінську мережу Ethernet посилає в Блок усунення аномального трафіку сигнал про початок аналізу й усунення атаки.

Блок усунення аномального трафіку

Блок усунення аномального трафіку – це автономний пристрій аналізу й фільтрації трафіку. Починаючи прийом трафіку, адресованого в конкретну зону, що, очевидно, піддається атаці, Блок усунення аномального трафіку проводить

точний аналіз цього трафіку. Якщо результати аналізу підтверджують, що трафік злочинний, Блок усунення аномального трафіку застосовує контрзаходи, наприклад, механізми анти-спуфінга й фільтрацію різного рівня (таблиця 3.1). Кінцевий результат полягає в тому, що трафік зі злочинних джерел скидається, а трафік із благонадійних джерел пересилається в передбачений пункт призначення.

Атаки DDoS – виявлення й усунення

Перерахуємо типи атак DDoS, які може виявляти й усувати Блок усунення аномального трафіку.

1. Атаки із заповненням смуги пропущення.

Лавинні атаки зі спуфінгом або без спуфінга:

– Прапор TCP (SYN, SYN-ACK, ACK, FIN).

– Протокол керування повідомленнями в Інтернет (ICMP).

– Протокол користувальницьких датаграмм (UDP).

Приклади: лавинна атака SYN, smurf, LAND і UDP – лавинні атаки.

Атаки зомбі-комп'ютерів/мереж зомбі-комп'ютерів, у яких кожний вихідний зомбі-пк або мережа відкриває множинні TCP-з'єднання й, у деяких випадках, видає багаторазові запити HTTP.

Атаки DNS, наприклад, лавинна атака із запитами DNS.

2. Атаки з дефіцитом ресурсів:

– Атаки пакетного розміру, характерна риса яких – фрагментированні або великі пакети. Приклади: teardrop і ping-of-death.

– Атаки зомбі-комп'ютерів/мереж зомбі-комп'ютерів з низькою інтенсивністю схожі на атаки із заповненням смуги пропущення за тим виключенням, що кожне джерело атаки посилає множинні запити з невеликим обсягом в одиницю часу.

– Атаки DNS з рекурсивним переглядом DNS.

Можливі варіанти зміни напрямку трафіку

Фахівці з IT можуть використовувати описані нижче варіанти зміни напрямку трафіку з його пересиланням з мережі, розташованого вище лежачого оператора зв'язку, на Блок усунення аномального трафіку. Цей процес також іменується “захватом” трафіку:

– Повідомлення прикордонного шлюзового протоколу (Border Gateway Protocol, BGP) із Блок усунення аномального трафіку на маршрутизатори, розташовані у вище лежачого оператора зв'язку, з інформацією про те, що трафік, адресований на захищену адресу призначення, буде переспрямований на Блок усунення аномального трафіку.

– Використання зовнішніх механізмів зміни напрямку трафіку, наприклад, маршрутизаторів віддаленого відновлення BGP.

– Повідомлення про ін'єкцію очищеного трафіку на маршруті (Route Health Injection, RHI) від Блок усунення аномального трафіку для процесу маршрутизації в Catalyst серії 6500 або в систему нагляду серії 7600. Ці повідомлення поміщають статичний маршрут у глобальну таблицю маршрутизації, у якій модуль Блок усунення аномального трафіку позначений як наступний вузол.

Можливі варіанти ін'єкції трафіку

Ін'єкція трафіку – це процес, застосовуваний у Блок усунення аномального трафіку для пересилання очищеного благонадійного трафіку в точку призначення, що піддається атаці. Рішення підтримує різні варіанти ін'єкції трафіку. У варіанті 2-ого рівня топології, очищений трафік пересилається із Блок усунення аномального трафіку на статично-конфігуруєму наступну адресу заходу. Ця адреса перебуває на маршрутизаторі, розташованому нижче й з'єднаним з тої ж VLAN або підмережею, що й інтерфейс/VLAN ін'єкції трафіку. Ін'єкцію трафіку на 2-му рівні найпростіше конфігурувати, оскільки тут не потрібно вносити які-небудь істотні зміни в конфігурацію маршрутизатора, розташованого нижче.

Варіанти ін'єкції трафіку 3-го рівня:

– Маршрутизація й пересилання по VPN (VPN Routing and Forwarding, VRF).

– Маршрутизація на основі політики (Policy-Based Routing, PBR).

– Транкинг VLAN (VLAN Trunking).

– Інкапсуляція по загальній маршрутизації (GRE) або інкапсуляція IP у тунелі IP (IPIP).

3.4 Розробка діаграми процесів

Діаграма процесів розробленої системи зображена на рисунку 3.3. Після початку роботи ми потрапляємо до інтерфейсу розробленого ПЗ, а саме до головного вікна. Далі можемо провести перегляд мережної статистики через моніторинг трафіку мережі та використовуючи аналізатор трафіку мережі.

Рисунок 3.3 – Діаграма взаємодії процесів

Крім цього через обробник помилок та вікно мережних дій ми можемо провести сканування топології мережі та провести побудову топології мережі.

Далі з вікна мережних дій також можна провести виявлення аномалій телекомунікаційного трафіку на основі дослідження інтенсивностей потоків даних у мережі з формуванням звіту який буде збережено в БД. Визначити вид атаки та блокувати і усувати аномальне поводження трафіку.

Застосовується методи динамічного виявлення аномалій в інтенсивностях потоків телекомунікаційних даних. Математично обґрунтованими видами аналізу часових рядів є дослідження сигналу в часовій, спектральній і спектрально-часовій областях. Слід зазначити, що досліджуваному телекомунікаційному сигналу властиво «коливальне поводження» через його формування на основі однозначно непередбачених дій користувачів або програм. Для досліджень такого роду залежностей у теорії цифрової обробки сигналів звичайно використовуються більше інформативні при розкритті частотної картини спектральні й спектрально-часові алгоритми.

Таким чином, розглянувши опис системи, структурну, функціональну схеми системи, та діаграму взаємодії процесів перейдемо до опису 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

Розглянемо алгоритм роботи основної програми. Його блок-схема зображена на рисунку 4.1. З рисунку видно, що блок схема складається з декількох основних блоків, а саме: блоки ініціалізації ПЗ; блоки завантаження ресурсів системи та перевірки на цілісність; роботи ПЗ; завершення роботи ПЗ. Також на блок-схемі є два посилання на підпрограми: сканування трафіку на аномалії; корегування мережного трафіку. Розглянемо всі блок схеми покроково з посиланнями.

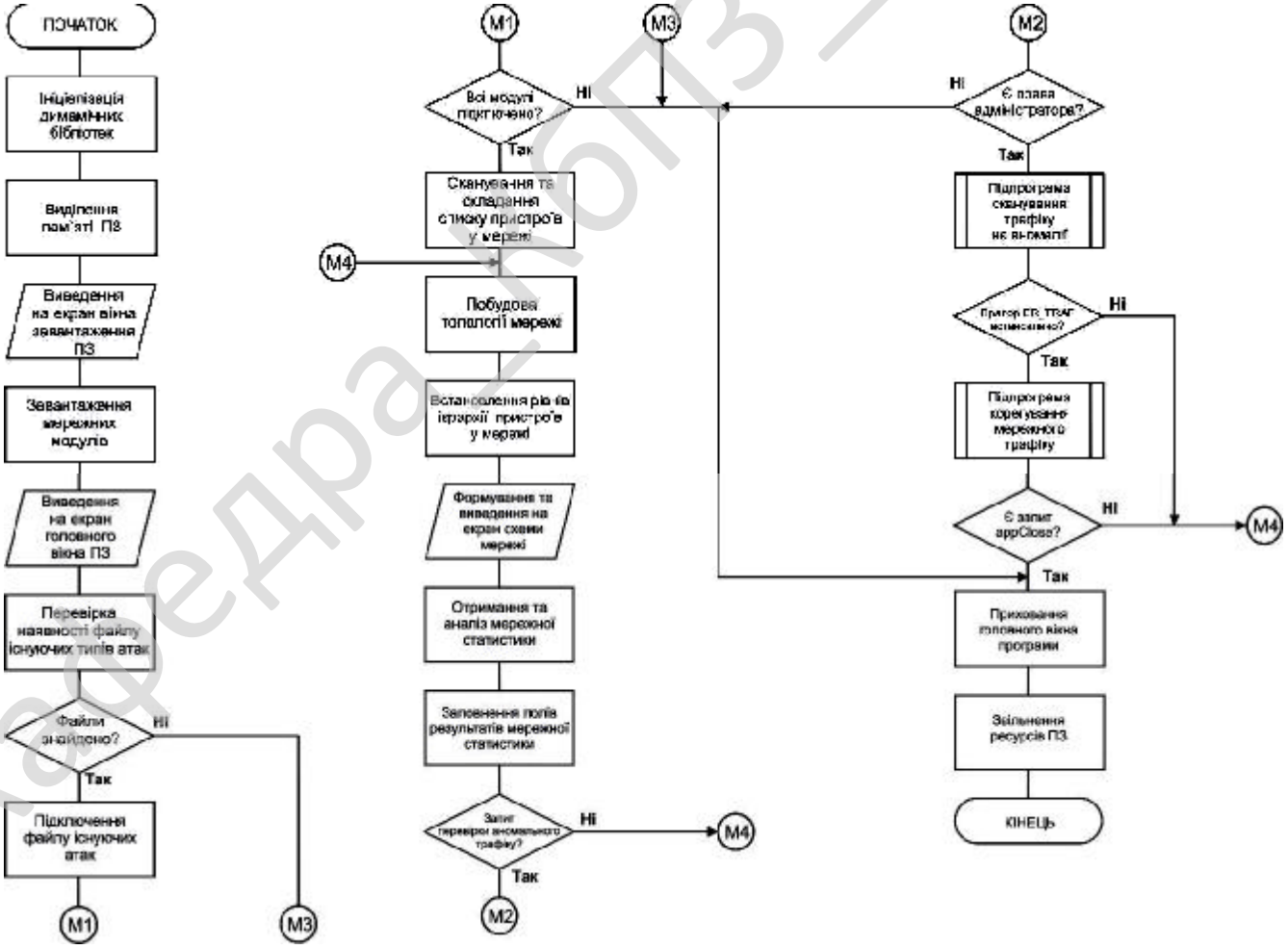

Рисунок 4.1 – Блок-схема основної програми

Після початку роботи програми проходить ініціалізація динамічних бібліотек, виділення пам`яті ПЗ та виведення на екран вікна завантаження ПЗ.

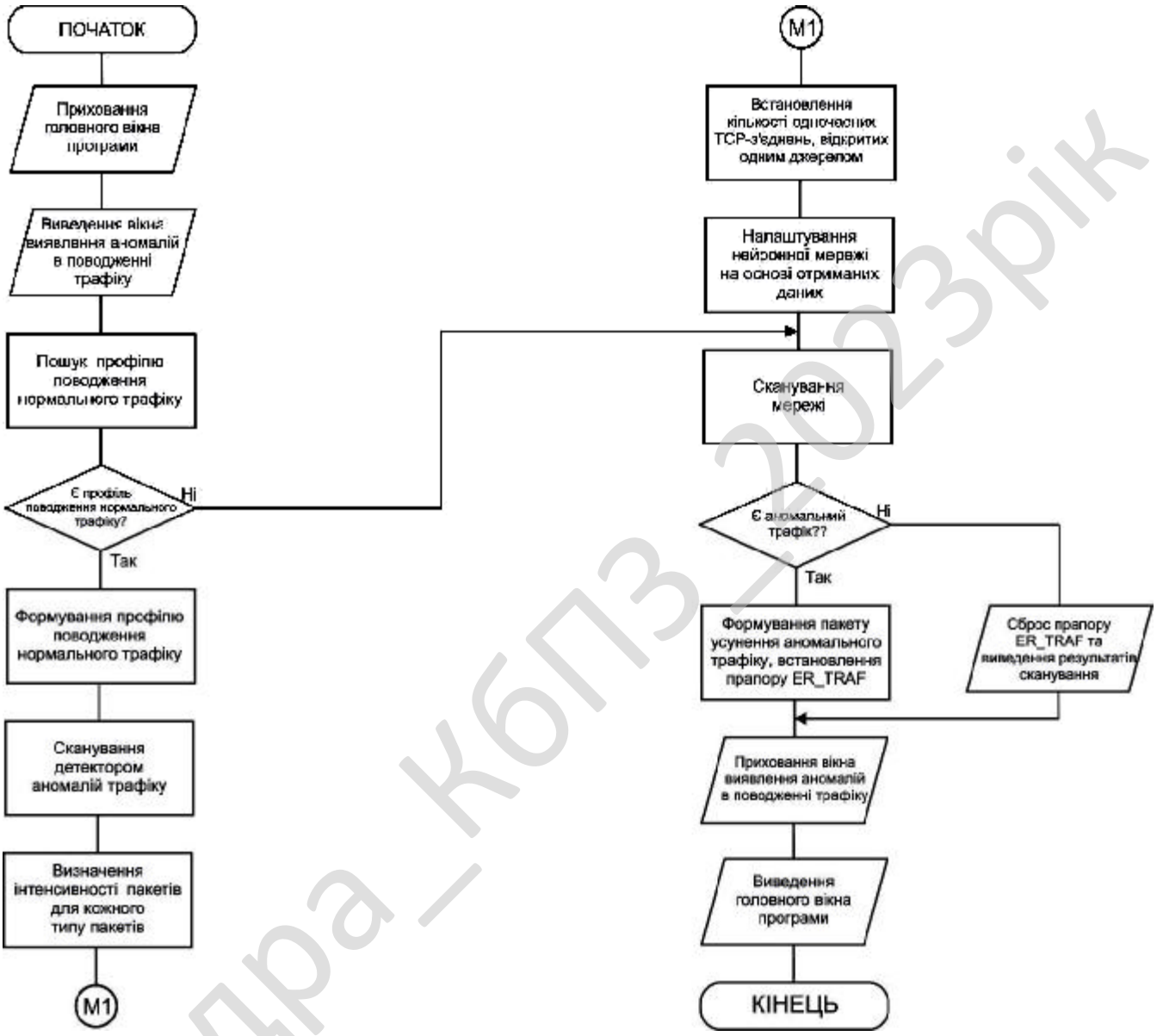

Рисунок 4.2 – Блок-схема роботи підпрограми сканування трафіку на аномалії

Після чого проходить завантаження мережних модулів, виведення на екран головного вікна ПЗ та перевірка наявності файлу існуючих типів атак. Файл знаходиться у папці з виконуючим файлом.

Якщо цей файл знайдено проходить підключення файлу існуючих атак, якщо ні ПЗ припиняє свою роботу. Далі проходе сканування та складання списку

пристроїв у мережі, побудова топології мережі та встановлення рівнів ієрархії пристроїв у мережі відносно отриманих даних.

Після цих дій проходить формування та виведення на екран схеми мережі, отримання та аналіз мережної статистики з заповненням полів результатів мережної статистики. Після запиту на перевірки аномального трафіку з перевіркою на наявність прав адміністратора виконується підпрограма сканування трафіку на аномалії яка зображена на рисунку 4.2.

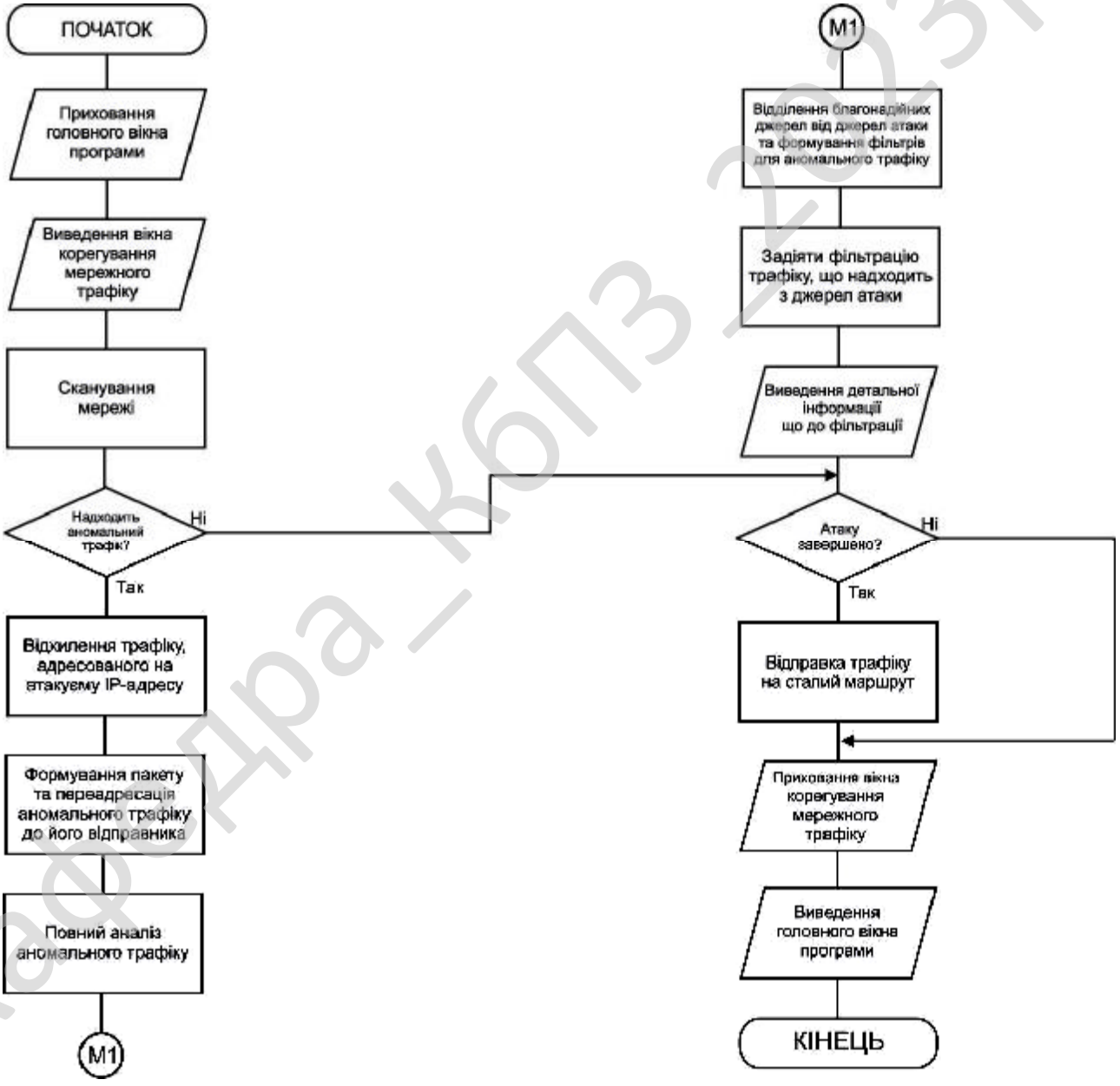

Рисунок 4.3 – Блок-схема роботи підпрограми корегування мережного трафіку

Далі якщо з підпрограми сканування трафіку на аномалії повернуто флаг ER_TRAF, яки вказує на наявність такого трафіка виконується запит з підпрограмою корегування мережного трафіку (рис.4.3). Далі проходе перевірка на отримання сигналу завершення роботи ПЗ (appClose), при отриманні якого проходить приховання головного вікна програми та звільнення ресурсів ПЗ

Опис розроблених функцій.

Розглянемо розроблені процедури та функції у бакалаврській роботі. Звичайно в XP використовуються функції Netapibufferallocate і Netapibufferfree – за допомогою їх виділяється й звільняється пам'ять під масив структур.

Я реалізував у бакалаврській роботі – оголошення структури Tshareinfo2Array = array [0..512] of Tshareinfo2; Тут пам'ять уже виділена.

Я волію користуватися саме таким оголошенням структури. І у вихідному коді розробленої бакалаврській роботі так і реалізовано.

Розглянемо розроблену функцію NetMRsharedel яка дозволить закрити обраний загальний ресурс.

```
Var NetMRsharedel:function (pszserver, psznetname:Pchar;
                            usreserved:Word):DWORD;
```

Параметри:

– ServerData повинен містити ім'я віддаленого комп'ютера, якщо закриваємо свої ресурси то даному параметру потрібно присвоїти NIL;

– Netname покажчик на рядок утримуючу ім'я ресурсу, що закривається.

Обидві функції не використовують ніяких структур. У якості імені передається не шлях до ресурсу, а саме ім'я ресурсу яке я визначив за допомогою коду (розглянутого вище). У випадку успішного виконання функцій, їх результат буде дорівнює нулю.

```
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 OS(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
//Якщо не знайдений завершення роботи  ПЗ
  if not lbxshares.Selected[i] then Exit;
  Sharename := lbxshares.Items.Strings[i];
  if OS then begin //Код для XP
    Flibhandle := Loadlibrary('MY_MR1.DLL');
    if Flibhandle = 0 then Exit;
    @NetMRsharedelnt := Getprocaddress(Flibhandle,'NetMRsharedel');
    if not Assigned(NetMRsharedelnt) 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
    NetMRsharedelnt(nil,Nament,0);  //Видаляємо ресурс
    Freemem(Nament); //Звільняємо пам'ять
  end else begin
    Flibhandle := Loadlibrary('MY_MR2.DLL');
    if Flibhandle = 0 then Exit;
    @NetMRsharedel := Getprocaddress(Flibhandle,'NetMRsharedel');
    if not Assigned(NetMRsharedel) then //Перевірка
    begin
      Freelibrary(Flibhandle);
      Exit;
    end;
    Fillchar(Name9x, Sizeof(Name9x), #0); //Очищаємо масив
    move(Sharename[1],Name9x[0],Length(Sharename));
//Заповнюємо масив
    NetMRsharedel(nil,@Name9x,0); //Видаляємо ресурс
  end;
  Freelibrary(Flibhandle);
end;
```

Розглянемо розроблену функцію NetMRshareadd. Відкриття локального ресурсу. Оголошення функції для Windows XP:

```
var
NetMRshareadd: function (ServerData: Pwidechar; level: DWORD; buf:Pointer;
parm_err: LPDWORD): DWORD;
```

Параметри:

– ServerData повинен містити ім'я віддаленого комп'ютера на якім повинна виконається функція, якщо відкриваємо локальний ресурс то даному параметру потрібно привласнити NIL;

– Lvl повинен містити ідентифікатор структури;

– buf повинен містити покажчик на структуру;

– parm_err містить покажчик помилки.

У функції для XP також використовується покажчик а не адреса покажчика. Розглянемо вихідний код виклику функції:

```
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 := '';
  Browseinfo.ulflags := BIF_RETURNONLYFSDIRS;
  lpitemid := Shbrowseforfolder(Browseinfo);
  if Assigned(lpitemid) then begin
    Shgetpathfromidlist(lpitemid, Temppath);
    Globalfreeptr(lpitemid);
  end else Result := '';
  Result := String(Temppath);
end;
```

Для виконання функції необхідно додати в розділ Uses модулі Shelldipapi і Shldipobj. Ось сам код відкриття ресурсу:

```
procedure Tmainform.btnaddsharesclick(Sender: Tobject);
const
  STYPE_DISKTREE = 0;  ACCESS_ALL = 258;  DDF_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 майбутнього ресурсу
//Визначаємо мережне ім'я
  Tmpname := Inputbox('Share name','Enter name','Test');
  if Tmpdir = '' then Exit;
  if not OS(OS) then Close; // З'ясовуємо тип системи
  if OS then begin //Код для XP
    Flibhandle := Loadlibrary('MY_MR1.DLL');
    if Flibhandle = 0 then Exit;
@NetMRshareaddnt:=Getprocaddress(Flibhandle,'NetMRshareadd');
    if not Assigned(NetMRshareaddnt) then
    begin
      Freelibrary(Flibhandle);
      Exit;
    end;
    Tmplength := Sizeof(Widechar)*256; //Визначаємо необхідний розмір
    Getmem(Tmpnament, Tmplength); //Конвертуємо в Pwchar
    Stringtowidechar(Tmpname, Tmpnament, Tmplength);
    Sharent.SS2_netname := Tmpnament; //Ім'я
    Sharent.SS2_type := STYPE_DISKTREE; //Тип ресурсу
    Sharent.SS2_remark := ''; //Коментар
    Sharent.SS2_permissions := ACCESS_READ; //Доступ
    //Кількість максимальних підключень
    Sharent.SS2_max_uses := DWORD( - 1);
    //Кількість поточних підключень
    Sharent.SS2_current_uses := 0;
    Getmem(Tmpdirnt, Tmplength);
    Stringtowidechar(Tmpdir, Tmpdirnt, Tmplength);
    Sharent.SS2_path := Tmpdirnt; //Шлях до ресурсу
    Sharent.SS2_passwd := nil; //Пароль
    NetMRshareaddnt(nil,2,@Sharent,nil); //Додаємо ресурс
    Freemem (Tmpnament); //звільняємо пам'ять
    Freemem (Tmpdirnt);
  end else begin //Код для 9x
    Flibhandle := Loadlibrary('MY_MR2.DLL');
    if Flibhandle = 0 then Exit;
    @NetMRshareadd := Getprocaddress(Flibhandle,'NetMRshareadd');
    if not Assigned(NetMRshareadd) then
    begin
```

```
      Freelibrary(Flibhandle);
      Close;
    end;
    Fillchar(Share9x.DD_netname, Sizeof(Share9x.DD_netname), #0);
    move(Tmpname[1],Share9x.DD_netname[0],Length(Tmpname)); //Ім'я
    Share9x.DD_type := STYPE_DISKTREE; //Тип ресурсу
    Share9x.DD_flags := DDF_RDONLY; //Доступ
    Fillchar(Share9x.DD_remark,
      Sizeof(Share9x.DD_remark), #0); //Коментар
    Fillchar(Share9x.DD_path,
      Sizeof(Share9x.DD_path), #0);
    Share9x.DD_path := Pansichar(Tmpdir); //Шлях до ресурсу
    Fillchar(Share9x.DD_rw_password,
      Sizeof(Share9x.DD_rw_password), #0);
//Пароль повного доступу
    Fillchar(Share9x.DD_ro_password,
    Sizeof(Share9x.DD_ro_password), #0); //Пароль для читання
    NetMRshareadd(nil,50,@Share9x,Sizeof(Share9x));
  end;
  Freelibrary(Flibhandle);
end;
```

Змінні Tmpnament і Tmpdirnt звільняються тільки після виконання функції. Це критично, а якщо ні, то структура передана функції буде із двома "незаповненими" полями.

Даний код відкриває новий ресурс на повний доступ.

Одержання списку поточних сесій. Для визначення користувачів підключених до комп'ютера була розроблена функція MyMR_Netsession. Оголошення функції для Windows XP:

```
MyMR_Netsession: function(ServerData, Uncclientname, Username: Pwchar;
                          Level: DWORD; bufptr: Pointer; prefmaxlen:DWORD;
                          entriesread, totalentries,
                          resume_handle: LPDWORD):DWORD;
```

Параметри:

– ServerData повинен містити ім'я віддаленого комп'ютера на якому повинна виконається функція, якщо дивимося в себе то даному параметрові потрібно привласнити NIL.

– Uncclientname містить покажчик на рядок утримуючу ім'я сесії.

– Username – містить покажчик на рядок утримуючу ім'я користувача.

– Lvl – повинен містити ідентифікатор структури.

– Bufptr – повинен містити адреса покажчика на масив структур.

– Prefmaxlen – повинен містити максимальну довжину повернутих даних у байтах.

– Entriesread повинен містити покажчик на змінну в яку запишеться кількість загальних ресурсів доступних на даний момент.

Для чіткого представлення часу роботи в бакалаврській роботі була розроблена функція, завдання якої перетворювати кількість секунд у більш 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 - (m*60);
    end;
    if m > 59 then begin
        h:=int(m/60);
        m:=m - (h*60);
    end;
    if h > 23 then begin
        d:=int(h/24);
        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;
```

Завершення сесій. Для завершення відкритих сесій розроблена функція MyMR_Netsessiondel. Оголошення функції для Windows XP:

```
Var MyMR_Netsessiondel: function(ServerData, Uncclientname, Username:Pwchar):DWORD;
```

Параметри:

ServerData – повинен містити ім'я віддаленого комп'ютера на якому повинна виконається функція;

uncclientname – повинен містити ім'я клієнта чия сесія завершується;

username – повинен містити ім'я користувача чия сесія завершується, якщо параметр NIL, завершаться всі сесії указані в параметрі uncclientname. Розглянемо вихідний код виклику функції:

```
procedure Tmainform.btnclosesessionclick(Sender: Tobject);
var
OS: Boolean; Flibhandle : Thandle; Cnament: Pwidechar; Cname9x: Pansichar;
  Key:Smallint; i: Integer;
begin
  if not OS(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('MY_MR1.DLL');
    if Flibhandle = 0 then Exit;
@MyMR_Netsessiondelnt := Getprocaddress(Flibhandle, 'MyMR_Netsessiondel');
    if not Assigned(MyMR_Netsessiondelnt) then
    begin Freelibrary(Flibhandle); Exit;
    end;
    //Перетворимо дані в необхідний вигляд
    Cnament := Pwchar(Widestring('\\'+lvsessions.Items.Item[i].Caption));
    MyMR_Netsessiondelnt(nil,Cnament,nil);
  end else begin
    Flibhandle := Loadlibrary('MY_MR2.DLL');
    if Flibhandle = 0 then Exit;
@MyMR_Netsessiondel:=Getprocaddress(Flibhandle, 'MyMR_Netsessiondel');
    if not Assigned(MyMR_Netsessiondel) then
    begin Freelibrary(Flibhandle); Exit;   end;
    //Перетворимо дані в необхідний вигляд
    Cname9x := Pansichar(lvsessions.Items.Item[i].Caption);
    key := Sessionclosekey[i]; //Беремо ключ із масиву
    MyMR_Netsessiondel(nil,Cname9x,Key);
  end;
  Freelibrary(Flibhandle);
end;
```

Розглянемо процедуру що надає інформацію про локальні ресурси й можливості їх контролю.

NetMRData – за допомогою цієї функції отримаємо дані про всі спільні ресурси. Оголошення функції для Windows XP:

```
NetMRData :function (ServerData:Pwchar; Level:DWORD; Bufptr:Pointer; Prefmaxlen:DWORD; Entriesread, Totalentries, resume_handle:LPDWORD): DWORD;
```

Параметри:

– ServerData повинен містити ім'я віддаленого комп'ютера, на якому повинна виконаться функція, якщо виконуємо у себе, то даному параметрові можна привласнити NIL.

– Lvl повинен містити ідентифікатор структури.

– Bufptr повинен містити адресу покажчика на масив структур.

– Prefmaxlen повинен містити максимальну довжину поверненних даних у байтах, якщо не ставити обмеження, то даному параметрові потрібно привласнити DWORD(– 1) .

– Entriesread повинен містити покажчик на змінну, у яку запишеться кількість спільних ресурсів доступних на даний момент.

Результати виконання будуть збережені в масиві структур переданих функції при її виклику. Існує 6 типів структур переданих функції NetMRData:

– SHARE_INFO_0 – тільки Windows XP SP0.

– SHARE_INFO_1 – тільки Windows XP SP1.

– SHARE_INFO_1 – тільки Windows 9x – Me.

– SHARE_INFO_2 – тільки Windows XP SP2.

– SHARE_INFO_50 – тільки Windows 9x – Me .

– SHARE_INFO_502 – тільки Windows XP SP3.

Структура SHARE_INFO_50, оголошення структури:

```
type
 TSHAREINFO50 = packed record
  DD_netname : array [0..12] of Char;
  DD_type:Byte;
  DD_flags:Word;
  DD_remark: Pchar;
```

```
  DD_path:Pchar;
  DD_rw_password:array [0..8] of Char;
DD_ro_password: array [0..8] of Char;
end;
```

Розроблені поля:

– DD_netname містить рядок з мережним ім'ям ресурсу;

– DD_type визначає тип ресурсу;

– DD_flags містить інформацію про права доступу до ресурсу;

– DD_remark покажчик на рядок необов'язкових коментарів;

– DD_path містить локальне розташування ресурсу;

– DD_rw_password містить пароль на запис – читання;

– DD_ro_password містить пароль на читання.

Розглянемо розроблену функцію, яка визначає тип системи.

```
function Tmainform.OS(var Value: Boolean): Boolean;
var Ver: Tosversioninfo;
    Bres: Boolean;
begin
  Ver.dwosversioninfosize := Sizeof(Tosversioninfo);
  Bres := Getversionex(Ver);
  if not Bres then
  begin
    Result := False;
    Exit;
  end else
    Result := True;
  case Ver.dwplatformid of
    VER_PLATFORM_WIN32_XP: Value := True;
//Windows XP  - підходить
    VER_PLATFORM_WIN32_WINDOWS : Value := False;
//Windows 9x - Me - не підходить
    VER_PLATFORM_WIN32s : Result := False;
//Windows 3.x  -  не підходить
  end;
end;
```

Якщо дана функція повернула результат True, виходить, визначення версії системи пройшло успішно, і тип системи буде зберігатися в змінній Value, а якщо

ні, то визначення типу системи, у цьому випадку прийде завершити виконання програми.

Дана функція буде практично найголовнішою, тому що вона буде вказувати у бакалаврській роботі яку частину коду виконувати.

Визначаємо, тип системи, завантажуємо необхідну бібліотеку, одержуємо адреси функцій і виконуємо функцію.

```
procedure Tmainform.btngetsharesclick(Sender: Tobject);
var
  i:Integer;
  Flibhandle : Thandle;
  Sharent : Pshareinfo2Array;
  entriesread,totalentries:DWORD;
  Share : array [0..512] of Tshareinfo50;
  pcentriesread,pctotalavail:Word;
  OS: Boolean;
begin
  lbxshares.Items.Clear;
  if not OS(OS) then Close; //Визначаємо тип системи
  if OS then begin //Код для XP
    Flibhandle := Loadlibrary('_R1.DLL'); //Завантажуємо бібліотеку
    if Flibhandle = 0 then Exit;
//Зв'язуємо функцію
    @NetMRDatant := Getprocaddress(Flibhandle,'NetMRData');
    if not Assigned(NetMRDatant) then //Перевірка
    begin
      Freelibrary(Flibhandle);
      Exit;
    end;
    Sharent := nil; //Очищаємо покажчик на масив структур
    //Виклик функції
    if NetMRDatant(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].SS2_netname));
```

```
  end else begin
    Flibhandle := Loadlibrary('My_MR2.DLL');
//Завантажуємо бібліотеку
    if Flibhandle = 0 then Exit;
    //Зв'язуємо функцію
    @NetMRData := Getprocaddress(Flibhandle,'NetMRData');
    if not Assigned(NetMRData) then
//Перевірка
    begin
      Freelibrary(Flibhandle);
      Exit;
    end;
    if NetMRData(nil,50,@Share,Sizeof(Share),
            @pcentriesread,@pctotalavail) <> 0 then //Виклик функції
    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].DD_netname));
  end;
  Freelibrary(Flibhandle); // вивантажити бібліотеку
end;
```

При виконанні цього коду Listbox заповниться назвами загальних ресурсів, які беруться з масиву структур. Масив заповнюється в результаті виконання функції.

4.2 Захист розробленого програмного забезпечення

Захист розробленого програмного забезпечення буде відбуватися за допомогою CRYPTON – алгоритм симетричного блочного шифрування (розмір блоку 128 біт, ключ довжиною до 256 біт), розроблений південнокорейським криптологом Чьо Лім Хун з південнокорейської компанії Future Systems, яка з кінця 1980-х років працює на ринку забезпечення мереж і захисту інформації. Алгоритм був розроблений в 1998 році в якості шифру – учасника конкурсу AES. Як зізнавався автор, конструкція алгоритму спирається на алгоритм SQUARE[1].

В алгоритмі Crypton немає традиційних для блочних шифрів мережі Фейстеля. Основу даного шифру становить так звана SP-мережа (повторювана циклова функція, що складається із замін-перестановок, орієнтована на розпаралелену нелінійну обробку всього блоку даних). Крім високої швидкості, перевагами таких алгоритмів є полегшення дослідження стійкості шифру до методів диференціального та лінійного криптоаналізу, що є на сьогодні основними інструментами розтину блочних шифрів. На конкурс AES була представлена версія алгоритму Crypton v0.5. Однак, як казав Чьо Лім Хун, йому не вистачало часу для розробки повної версії. І вже на першому етапі конкурсу AES в ході аналізу алгоритмів, версія Crypton v0.5 була замінена на версію Crypton v1.0. Відмінність нової версії від первинної полягала в зміні таблиці замін та в модифікації процесу розширення ключа.

Як і інші учасники конкурсу AES, Crypton призначений для шифрування 128-бітових блоків даних[2]. При шифруванні використовуються ключі шифрування для декількох фіксованих розмірів – від 0 до 256 біт з кратністю 8 бітів. Структура алгоритму Crypton – структура «Квадрата» – багато в чому схожа на структуру алгоритму Square, створеного в 1997 році. Криптографічні перетворення для алгоритмів з даною структурою можуть бути виконані як для цілих рядків і стовпців масиву, так і над окремими його байтами. (Варто зазначити, що алгоритм Square був розроблений авторами майбутнього переможця конкурсу AES – авторами алгоритму Rijndael – Вінсентом Ріджменом і Джоан Дейменом.)

Шифрування

Алгоритм Crypton являє 128-бітовий блок шифруємих даних у вигляді байтового масиву 4×4, над якими в процесі шифрування проводиться кілька раундів перетворень. У кожному раунді передбачається послідовне виконання наступних операцій:

– Таблична заміна γ;

– Лінійне перетворення π;

– Байтова перестановка τ;

– Операція σ.

Таблична заміна γ

Алгоритм Crypton використовує 4 таблиці замін. Кожна з яких заміщає 8-бітне вхідне значення на вихідне такого ж розміру.

Лінійне перетворення π

Тут використовується 4 спеціальні константи. Ці константи об'єднані в маскуючі послідовності

Байтова перестановка τ

Дана перестановка перетворює найпростішим чином рядок даних у стовпець.

Операція σ

Дана операція є побітовим складанням всього масиву даних з ключем раунду. Зауважимо, саме 12 раундів шифрування рекомендується автором алгоритму Чьо Хун Лімом, проте сувора кількість раундів не встановлена.

5 МЕТОДИКА ВПРОВАДЖЕННЯ СИСТЕМИ КІБЕРБЕЗПЕКИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

Програма має простий та інтуїтивно зрозумілий інтерфейс, який зображений на рисунку 5.1. Після початку роботи необхідно вказати через яке саме підключення буде працювати ПЗ. Тобто через яке з'єднання буде проходити взаємодія з комп'ютерною мережею.

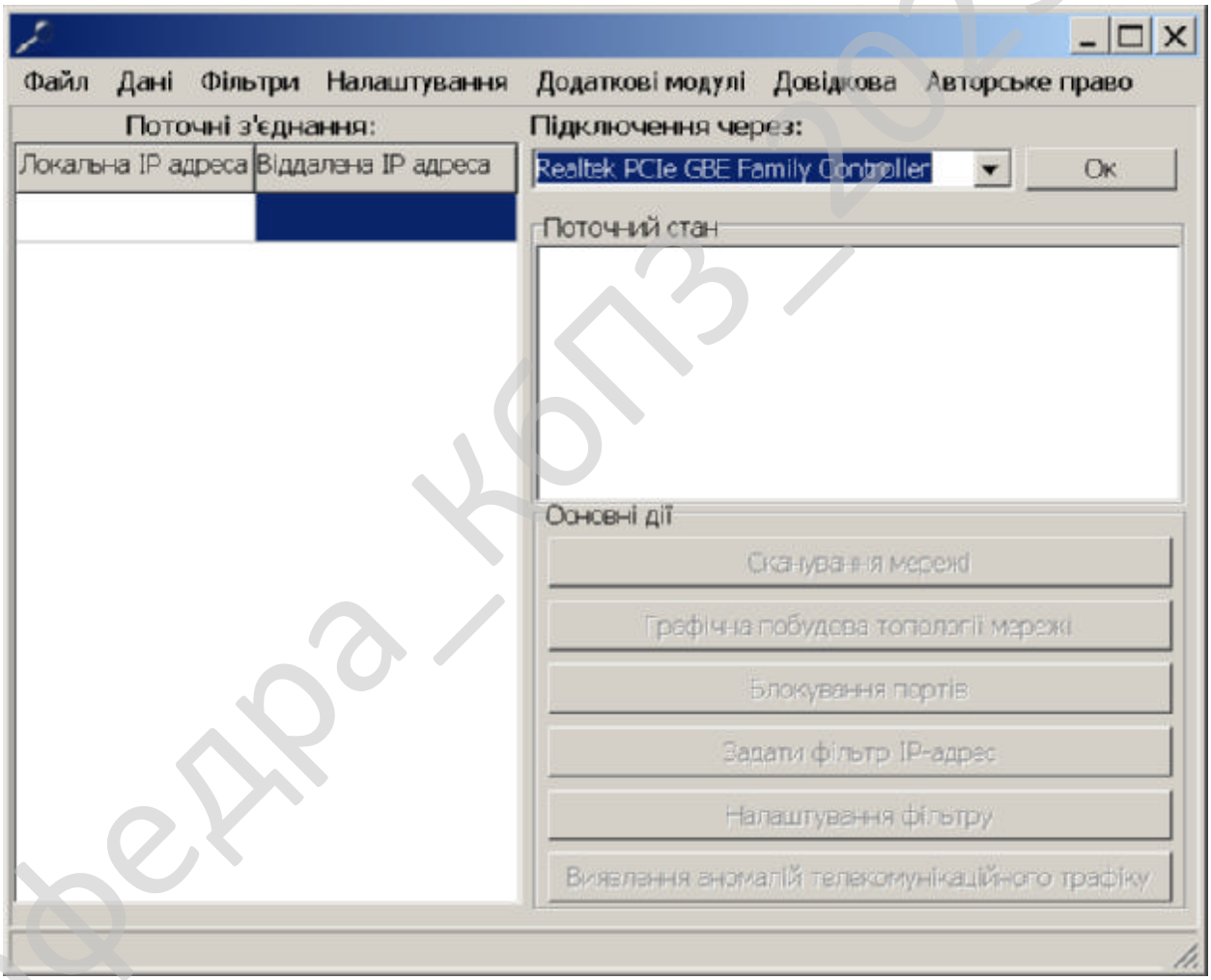

Рисунок 5.1 – Головне вікно програми, підключення до мережної карти

Комп'ютерна мережа це система зв'язку між двома чи більше комп'ютерами. У ширшому розумінні комп'ютерна мережа це система зв'язку

через кабельне чи повітряне середовище, самі комп'ютери різного функціонального призначення і мережеве обладнання.

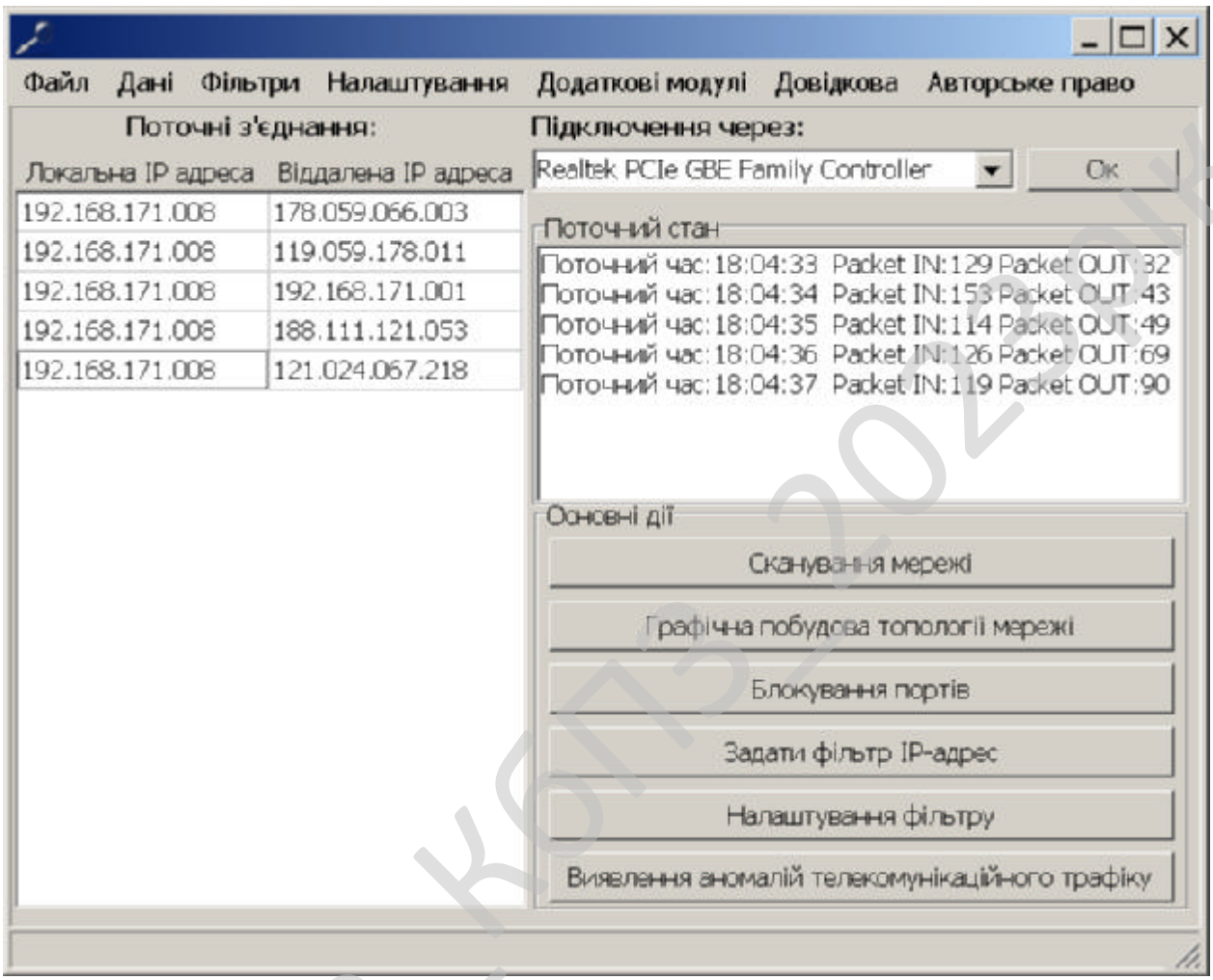

Рисунок 5.2 – Головне вікно програми, робота системи

Для передачі інформації можуть бути використані різні фізичні явища, як правило – різні види електричних сигналів чи електромагнітного випромінювання. Середовищами передавання у комп'ютерних мережах можуть бути телефонні кабелі, та спеціальні мережеві кабелі: коаксіальні кабелі, виті пари, волоконно-оптичні кабелі, радіохвилі, світлові сигнали.

Після обрання підключення програма починає працювати, як це показано на рисунку 5.2.

Інтерфейс розподілено на основні п’ять частин. Розглянемо їх:

1 частина. Меню користувача (верхня частина вікна). В меню користувача відображаються всі дії які можна виконати з програмою: Файл; Дані; Фільтр; Налаштування; Додаткові модулі; Довідка; Авторське право.

2 частина. Поточні з’єднання (ліва частина вікна). Відображує дані поточного з’єднання: Локальна IP адреса; Віддалена IP адреса.

3 частина. Підключення (права верхня частина вікна). Обирає мережеву карту яка використовується.

4 частина. Основні дії (права нижня частина вікна). Функціонал програми: Сканування мережі; Графічна побудова топології мережі; Блокування портів; Задати фільтр IP-адрес; Налаштування фільтру; Виявлення аномалій телекомунікаційного трафіку.

5 частина. Статус бар (нижня частина вікна). Служить для відображення допоміжної інформації при наведені маніпулятором миші на об’єкти. Як приклад при наведені до поточного з’єднання відображу часові дані.

На рисунку 5.3 зображено розроблена форма авторського права.

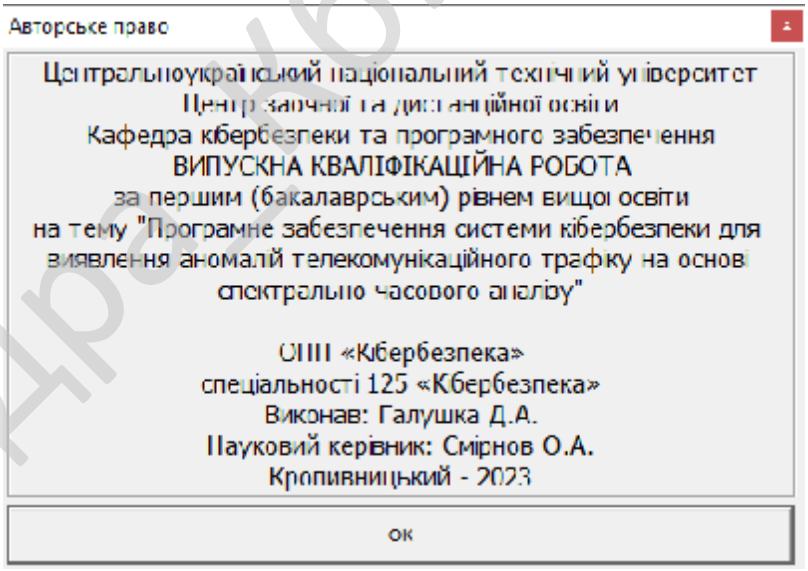

Рисунок 5.3 – Авторське право

6 ОСНОВНІ ВИСНОВКИ

Програмне забезпечення, створене в результаті виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти, призначено для системи кібербезпеки для виявлення аномалій телекомунікаційного трафіку на основі спектрально-часового аналізу.

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

Рішення завдання полягало у вирішенні наступних задач:

– Був проведений огляд існуючих систем для виявлення аномалій телекомунікаційного трафіку на основі спектрально-часового аналізу.

– Досліджена система для виявлення аномалій телекомунікаційного трафіку на основі спектрально-часового аналізу.

– На основі отриманих результатів досліджень створена програмна реалізація системи кібербезпеки для виявлення аномалій телекомунікаційного трафіку на основі спектрально-часового аналізу.

Розроблені під час виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти алгоритми дозволяють успішно вирішувати завдання для виявлення аномалій телекомунікаційного трафіку на основі спектрально-часового аналізу.

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 Delphi 10. Дана мова програмування дозволяє найбільш ефективно обробляти дані призначені для

системи кібербезпеки для виявлення аномалій телекомунікаційного трафіку на основі спектрально-часового аналізу. 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 кібербезпеки й включає програми, що реалізують її функції.

Програма призначена для виконання під управлінням багатозадачної операційної системи кібербезпеки Windows 10/11.

Даються необхідні рекомендації з установки розробленого програмного забезпечення.

Для підвищення рівня безпеки запропоновано застосовувати алгоритм CRYPTON.

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

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

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

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

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

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

7. Смирнов А.А. Анализ и сравнительное исследование перспективных направлений развития цифровых телекоммуникационных систем и

сетей / А.А.Смирнов, В.В.Босько, Е.В.Мелешко // Системи обробки інформації. – Х.: ХУ ПС, 2008. – Вип.7(74). – С.120-123.

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

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

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

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

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

13. Смирнов С. А. Усовершенствованный алгоритм управления доступом к «облачным» телекоммуникационным ресурсам / Мохамад Абу Таам

Гани, А. А. Смирнов, Н. С. Якименко, С. А. Смирнов // Системи обробки інформації: зб. наук. праць. – Х.: ХУПС, 2015. –Вип. 1(126). – С. 150-153.

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

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

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

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

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

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

20. Смирнов С. А. Метод безопасной маршрутизации на базовом множестве путей передачи метаданных в облачные антивирусные системы /

В. Л. Бурячок, С. А. Смирнов // Системи управління, навігації та зв'язку. – Полтава, 2016. – Вип. 4(40). – С. 57-62.

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

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

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

24. Смирнов С. А. Метод управления доступом к «облачным» ресурсам для защиты телекоммуникационных систем / Мохамад Абу Таам Гани, А. А. Смирнов, С. А. Смирнов // Всеукраїнська науково-практична конференція «Інформаційна безпека держави, суспільства та особистості», м. Кіровоград, 16 квітня 2015 р.: зб. тез доп. – Кіровоград: КНТУ, 2015. – С. 50-52.

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

26. Смирнов С.А. Реализация метода управления доступом в интеллектуальных узлах коммутации / А.А. Смирнов, Мохамад Абу Таам Гани, С.А. Смирнов // Збірник тез XVII міжнародного науково-практичного семінару

«Комбінаторні конфігурації та їх застосування». м. Кіровоград. 17-18 квітня 2015 р. – Кіровоград: КНТУ. – 2015. – С. 91-92.

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

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

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

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

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

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

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

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

35. Смирнов С. А. Алгоритмы формирования множества маршрутов передачи метаданных в облачные антивирусные системы / А. А. Смирнов, С. А. Смирнов, А. К. Дидык // Актуальні питання забезпечення кібернетичної безпеки та захисту інформації: зб. наук. праць II Міжнар. наук.-практ. конф., м. Київ, 24-27 лютого 2016 р. – К.: Європейський університет, 2016. – С. 140-142.

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

37. Смирнов С. А. Алгоритм формирования базового множества маршрутов передачи метаданных в облачные антивирусные системы /

А. А. Смирнов, С. А. Смирнов, А. К. Дидык // Інформатика та системні науки (ІСН-2016): зб. тез VII всеукр. наук.-практ. конф., м. Полтава, 10-12 березня 2016 р. – Полтава: ПУЕТ, 2016. – С. 261-263.

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

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

40. Смирнов С. А. Исследование способа контроля линий связи телекоммуникационной системы для облачных антивирусов / А. А. Смирнов, С. А. Смирнов, А. К. Дидык // Збірник тез першої міжнародної науково-практичної конференції «Проблеми науково-технічного та правового забезпечення кібербезпеки у сучасному світі» (ПНПЗК-2016), м. Харків, 30 березня - 1 квітня 2016 р. – Х.: НТУ «ХПІ», 2016. – С. 14.

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

42. Смирнов С. А. Разработка и исследование способа контроля линий связи телекоммуникационных сетей для облачных антивирусных систем / А. А. Смирнов, С. А. Смирнов, А. К. Дидык // Проблеми і перспективи розвитку

ІТ-індустрії: VIII міжнар. наук.-практ. конф., м. Харків, 28-29 квітня 2016 р.: зб. тез. – Х.: ХНЕУ, 2016. – С. 48.

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

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

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

46. Современные телекоммуникации. Технологии и экономика / [В.Л. Банкет, О.В. Бондаренко, П.П. Воробьенко и др.]; под ред. С.А. Довгого. – М.: Эко-Трендз, 2003. – 320 с.

47. Столлингс В. Современные компьютерные сети / Вильям Столлингс. –СПб.: Питер, 2003. – 778 с.

48. Таненбаум Э. Компьютерные сети / Эндрю Таненбаум; пер. с англ. А. Леонтьев. – СПб.: Питер, 2002. – 848 с.

49. Телекоммуникационные системы и сети: учебное пособие. В 3 томах / [В.В. Величко, Е.А. Субботин, В.П. Шувалов, А.Ф. Ярославцев]; под ред. В.П. Шувалова. – М.: Горячая линия-Телеком, 2005, т. 3 – 592 с.

50. Хайкин С. Нейронные сети: полный курс / С. Хайкин. – М.:Вильямс, 2006. – 1103 с.

Додаток А
(обов'язковий)

Технічне завдання

Зміст

1 Найменування та область застосування

Це технічне завдання розповсюджується на розробку системи кібербезпеки для виявлення аномалій телекомунікаційного трафіку на основі спектрально-часового аналізу.

2 Підстава для розробки

Підставою для розробки служить завдання на випускну кваліфікаційну роботу за першим (бакалаврським) рівнем вищої освіти, видане на кафедрі кібербезпеки та програмного забезпечення (нак. № 17-02 від 5.01.2023 року).

3 Мета та призначення розробки

Метою випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти є розробка програмного забезпечення системи кібербезпеки для виявлення аномалій телекомунікаційного трафіку на основі спектрально-часового аналізу.

4 Джерела розробки

Джерелом цієї випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти 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 кібербезпеки з ОС та з користувачем;

– розробка програми, що реалізує спроектовані алгоритми роботи системи.

5.2 Показники призначення

Система повинна забезпечувати:

– системи кібербезпеки для виявлення аномалій телекомунікаційного трафіку на основі спектрально-часового аналізу;

– 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 10/11 і з сумісними з цією платформою пристроями і прикладним програмним забезпеченням.

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 10/11.

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 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 – 82 аркушів.

8 Етапи розробки

8.1 Збір і обробка інформації по темі випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти. Постановка задачі на виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти (складання ТЗ).

8.2 Проведення досліджень або експериментальних робіт для уточнення основних положень випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти.

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 випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти на попередній захист 23.05.2023 р.

11.2 Подання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти на захист 3.06.2023 р.

Додаток Б

(обов’язковий)

Міністерство освіти і науки України

Центральноукраїнський національний технічний університет

ЗАТВЕРДЖУЮ

Керівник випускної кваліфікаційної роботи за

першим (бакалаврським) рівнем вищої освіти

_____ Смірнов О.А.

Програмне забезпечення системи кібербезпеки для виявлення аномалій телекомунікаційного трафіку на основі спектрально-часового аналізу

Лістинг програми

Код документу 12

Носій: CD/DVD-диск / USB-флеш-накопичувач

Загальна кількість аркушів: 49

Літера: РП

Кропивницький – 2023 року

Основна програма

Файл Viyavlennya_anomaliy_net_traffic.dpr основної програми

```
program Viyavlennya_anomaliy_net_traffic;

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

  NETBIOSTypes  : 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_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            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_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' , ' tokenring' , ' FDDI' , ' PPP' , ' loopback' ,
' SLIP'  );
  AdaptTypes    : array[1..28] of string[10] =
    ( ' інший' , ' ' , ' ' , ' ' , ' ' , ' ethernet' , ' ' , ' ' , ' 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 для виявлення аномалій
телекомунікаційного трафіку на основі спектрально-часового аналізу
(Preventivniy_Zahist_ot_Shkidlivih_Program)-----------------------

//////////////////////////
//                                                                  //
// 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 = 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 для виявлення аномалій телекомунікаційного трафіку на основі
спектрально-часового аналізу (Preventivniy_Zahist_ot_Shkidlivih_Program)
  end;

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

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

```
//-----UDP СТРУКТУРА --------------------------

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

//------IP СТРУКТУРА -------------------------

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

```
  PTMibIPAddrRow = ^TMibIPAddrRow;
  TMibIPAddrRow = packed record
    dwAddr: DWORD;
    dwIndex: DWORD;
    dwMask: DWORD;
    dwBCastAddr: DWORD;
    dwReasmSize: DWORD;
    Unused1,
    Unused2: WORD;
  end;
  //
  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;

GetNetworkParams: 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;

GetIpNetTable: function ( pIpNetTable: PTMibIPNetTable;
  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' ) ;
    GetNetworkParams := GetProcAddress (IpHlpModule, ' GetNetworkParams' ) ;
    GetTcpTable := GetProcAddress (IpHlpModule, ' GetTcpTable' ) ;
    GetTcpStatistics := GetProcAddress (IpHlpModule, ' GetTcpStatistics' ) ;
    GetUdpTable := GetProcAddress (IpHlpModule, ' GetUdpTable' ) ;
```

```
    GetUdpStatistics := GetProcAddress (IpHlpModule, ' GetUdpStatistics' ) ;
    GetIpStatistics := GetProcAddress (IpHlpModule, ' GetIpStatistics' ) ;
    GetIpNetTable := GetProcAddress (IpHlpModule, ' GetIpNetTable' ) ;
    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 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;
    NetTree: 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 NetTreeCustomDrawItem(Sender: TCustomTreeView;
      Node: TTreeNode; State: TCustomDrawState; var DefaultDraw: Boolean);
    procedure NetTreeDblClick(Sender: TObject);
    procedure NetTreeGetImageIndex(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 NetContainerToOpen:
PNetResource);
  //  function OpenEnum(const NetContainerToOpen: PNetResource;
 //        ResScope, ResType, ResUsage: DWORD): THandle;
 //   function EnumResources(const ParentNode: TTreeNode;
  //       ResScope, ResType, ResUsage: DWORD; hNetEnum: THandle): UINT;
  end;

type
 TShareInfo2 = packed record
  shi2_netname : 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_netname : 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
NetShareEnumNT:function (     servername:PWChar;
                          level:DWORD;
                          bufptr:Pointer;
                          prefmaxlen:DWORD;
                          entriesread,
                          totalentries,
                          resume_handle:LPDWORD): DWORD; stdcall;

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

var
NetShareDelNT:function (servername: PWideChar;
                        netname: PWideChar;
                        reserved: DWORD): LongInt; stdcall;

var
NetShareDel:function (  pszServer,
                        pszNetName:PChar;
                        usReserved:Word): DWORD; stdcall;

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

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

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

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

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

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

var
```

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

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

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

var
NetFileClose2: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(' NETAPI32.DLL' ); //Завантажуємо бібліотеку
    if FLibHandle = 0 then Exit;
    //Зв' язуємо функцію
    @NetShareEnumNT := GetProcAddress(FLibHandle,' NetShareEnum' );
    if not Assigned(NetShareEnumNT) then //Перевірка
    begin
      FreeLibrary(FLibHandle);
      Exit;
    end;
    ShareNT := nil; //Очищаємо покажчик на масив структур
    //Виклик функції
    if NetShareEnumNT(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_netname));
  end else begin //Код для 9 x-me
    FLibHandle := LoadLibrary(' SVRAPI.DLL' ); //Завантажуємо бібліотеку
    if FLibHandle = 0 then Exit;
    //Зв' язуємо функцію
    @NetShareEnum := GetProcAddress(FLibHandle,' NetShareEnum' );
    if not Assigned(NetShareEnum) then //Перевірка
    begin
      FreeLibrary(FLibHandle);
      Exit;
    end;
    //Виклик функції
    if NetShareEnum(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_netname));
  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(' NETAPI32.DLL' );
    if FLibHandle = 0 then Exit;
    @NetShareDelNT := GetProcAddress(FLibHandle,' NetShareDel' );
    if not Assigned(NetShareDelNT) 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
    NetShareDelNT(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;
    @NetShareDel := GetProcAddress(FLibHandle,' NetShareDel' );
    if not Assigned(NetShareDel) then //Перевірка
    begin
      FreeLibrary(FLibHandle);
      Exit;
    end;
    FillChar(Name9x, SizeOf(Name9x), #0); //Очищаємо масив
    move(ShareName[1],Name9x[0],Length(ShareName)); //Заповнюємо масив
    NetShareDel(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(' NETAPI32.DLL' );
    if FLibHandle = 0 then Exit;
    @NetShareAddNT := GetProcAddress(FLibHandle,' NetShareAdd' );
    if not Assigned(NetShareAddNT) 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_netname := 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); // Кіл-ть максим. підключ.
    ShareNT.shi2_current_uses := 0; // Кіл-ть поточних підкл.

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

    ShareNT.shi2_passwd := ' ' ; //Пароль

    NetShareAddNT(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;
    @NetShareAdd := GetProcAddress(FLibHandle,' NetShareAdd' );
    if not Assigned(NetShareAdd) then
    begin
      FreeLibrary(FLibHandle);
      Exit;
    end;
    FillChar(Share9x.shi50_netname, SizeOf(Share9x.shi50_netname), #0);
    move(TmpName[1],Share9x.shi50_netname[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
    NetShareAdd(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 Кіл-ть 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 для виявлення аномалій
телекомунікаційного трафіку на основі спектрально-часового аналізу
(Preventivniy_Zahist_ot_Shkidlivih_Program)
//

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(' NETAPI32.DLL' );
    if FLibHandle = 0 then Exit;
    @NetSessionEnumNT := GetProcAddress(FLibHandle, ' NetSessionEnum' );
    if not Assigned(NetSessionEnumNT) then
    begin
      FreeLibrary(FLibHandle);
      Exit;
    end;
    SessionInfo502 := nil;
    if NetSessionEnumNT(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;
   @NetSessionEnum := GetProcAddress(FLibHandle, ' NetSessionEnum' );
    if not Assigned(NetSessionEnum) then
    begin
      FreeLibrary(FLibHandle);
      Exit;
    end;
   if NetSessionEnum
(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 для виявлення аномалій телекомунікаційного трафіку на основі
спектрально-часового аналізу (Preventivniy_Zahist_ot_Shkidlivih_Program)
        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(‘ NETAPI32.DLL’ );
    if FLibHandle = 0 then Exit;
    @NetSessionDelNT := GetProcAddress(FLibHandle, ‘ NetSessionDel’ );
    if not Assigned(NetSessionDelNT) then
    begin
      FreeLibrary(FLibHandle);
      Exit;
    end;
    //Перетворимо дані в необхідний вид
    CNameNT := PWChar(WideString(‘ \\’ +lvSessions.Items.Item[i].Caption));
    NetSessionDelNT(nil,CNameNT,nil);
  end else begin
    FLibHandle := LoadLibrary(‘ SVRAPI.DLL’ );
    if FLibHandle = 0 then Exit;
    @NetSessionDel := GetProcAddress(FLibHandle, ‘ NetSessionDel’ );
    if not Assigned(NetSessionDel) 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
    NetSessionDel(nil,CName9x,Key);
  end;
  FreeLibrary(FLibHandle);
end;

//////////////////////////
//
//  Одержання списку відкритих файлів досліджуємої мережі для виявлення аномалій
телекомунікаційного трафіку на основі спектрально-часового аналізу
(Preventivniy_Zahist_ot_Shkidlivih_Program)
//

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(‘ NETAPI32.DLL’ );
    if FLibHandle = 0 then Exit;
    @NetFileEnumNT := GetProcAddress(FLibHandle, ‘ NetFileEnum’ );
    if not Assigned(NetFileEnumNT) then
```

```
    begin
      FreeLibrary(FLibHandle);
      Exit;
    end;
    FileInfoNT := nil;
    if NetFileEnumNT(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;
   @NetFileEnum := GetProcAddress(FLibHandle, ' NetFileEnum' );
    if not Assigned(NetFileEnum) then
    begin
      FreeLibrary(FLibHandle);
      Exit;
    end;
   if NetFileEnum (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(' NETAPI32.DLL' );
    if FLibHandle = 0 then Exit;
    @NetFileClose := GetProcAddress(FLibHandle, ' NetFileClose' );
    if not Assigned(NetFileClose) then
    begin
      FreeLibrary(FLibHandle);
      Close;
    end;
    NetFileClose(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;
```

```
    @NetFileClose2 := GetProcAddress(FLibHandle, ' NetFileClose2' );
    if not Assigned(NetFileClose2) then
    begin
      FreeLibrary(FLibHandle);
      Close;
    end;
    NetFileClose2(nil,StrToInt(lvFiles.Items.Item[i].Caption));
  end;
  FreeLibrary(FLibHandle);
end;

////////////////////////////
//
//  Визначаємо вхідний / вихідний трафік досліджуємої мережі для виявлення
аномалій телекомунікаційного трафіку на основі спектрально-часового аналізу
(Preventivniy_Zahist_ot_Shkidlivih_Program)
//

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 для виявлення аномалій телекомунікаційного трафіку на
основі спектрально-часового аналізу (Preventivniy_Zahist_ot_Shkidlivih_Program)
        SubItems.Add(IntToStr(Table.Table[i].dwOutOctets)); //Усього відправлено
байт у досліджуєму мережу для виявлення аномалій телекомунікаційного трафіку на
основі спектрально-часового аналізу (Preventivniy_Zahist_ot_Shkidlivih_Program)
      end;
```

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


 function OpenEnum(const NetContainerToOpen: PNetResource; ResScope, ResType,
ResUsage: DWORD): THandle;
var
 hNetEnum: THandle;
begin
 Result:=0;
 if (NO_ERROR<>WNetOpenEnum(ResScope, ResType, ResUsage,
                                  NetContainerToOpen, hNetEnum))
 then ShowMessage(‘ Помилка!’ )
 else Result:=hNetEnum;
end;

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

const
 RESOURCE_BUF_ENTRIES = 2000;

var
 ResourceBuffer: array[1..RESOURCE_BUF_ENTRIES] of TNetResource;
 i, ResourceBuf, EntriesToGet: dword;
 NewNode: TTreeNode;
begin
 Result:=0;
 while true do
  begin
   ResourceBuf:=sizeof(ResourceBuffer);
   EntriesToGet:=RESOURCE_BUF_ENTRIES;
   if (NO_ERROR<>WNetEnumResource(hNetEnum, 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
       // WNetEnumResource, 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 NetContainerToOpen: PNetResource);
var
 hNetEnum: THandle;
begin
 hNetEnum:=OpenEnum(NetContainerToOpen, ResScope, ResType, ResUsage);
 if (hNetEnum=0)
 then Exit;
 EnumResources(ParentNode, ResScope, ResType, ResUsage, hNetEnum);
 if (NO_ERROR<>WNetCloseEnum(hNetEnum))
 then ShowMessage(' WNetCloseEnum Помилка' );
end;

procedure TMainForm.Button1Click(Sender: TObject);
var
 ResScope, ResType, ResUsage: dword;
begin
 Button1.Caption:=' Пошук мережних ресурсів. Чекайте...' ;
 Button1.Enabled:=false;
 //
 NetTree.Items.Clear;
 case rgScope.ItemIndex of
  1: ResScope:=RESOURCE_GLOBALNET;
  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(NetTree.Items.Add(nil, ' Network Resources' ),
                          ResScope, ResType, ResUsage, nil);
 //
 Button1.Caption:=' Обновити список ресурсів' ;
 Button1.Enabled:=true;

end;

procedure TMainForm.NetTreeCustomDrawItem(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.NetTreeDblClick(Sender: TObject);
begin
ShellExecute(0,' open' ,PChar(NetTree.Selected.Text),' ' ,' ' ,SW_SHOW);
end;

procedure TMainForm.NetTreeGetImageIndex(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.
```

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 '  ),
    ( 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 Network News Transfer
Protocol }
    ( Prt: 123; Srv: ' NTP    '  ),     { Протокол Network Time protocol
}
    ( Prt: 135; Srv: ' DCOMRPC'  ),     { Протокол Location Service
}
    ( Prt: 137; Srv: ' NBNAME '  ),     { NETBIOS сервіс імен          }
    ( Prt: 138; Srv: ' NBDGRAM'  ),     { NETBIOS сервіс датаграм     }
    ( Prt: 139; Srv: ' NBSESS '  ),     { NETBIOS сервіс сесій         }
    ( Prt: 143; Srv: ' IMAP   '  ),     { Протокол Internet Message Access
Protocol }
    ( Prt: 161; Srv: ' SNMP   '  ),     { Протокол Simple Netw. 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_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' , ' NETMGMT' , ' ICMP' , ' EGP' ,
    ' GGP' , ' HELO' , ' RIP' , ' IS_IS' , ' ES_IS' ,
    ' CISCO' , ' BBN' , ' OSPF' , ' BGP' , ' BOOTP' ,
    ' AUTO_STAT' , ' STATIC' , ' NOT_DOD'  );

type

// для IpHlpNetworkParams
  TNetworkParams = 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 IpHlpNetworkParams (var NetworkParams: TNetworkParams): integer ;
procedure Get_NetworkParams( 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 виявлення
аномалій телекомунікаційного трафіку на основі спектрально-часового аналізу
(Preventivniy_Zahist_ot_Shkidlivih_Program)}

procedure Get_NetworkParams( List: TStrings );
var
    NetworkParams: TNetworkParams ;
    I, ErrorCode: integer ;
begin
    if not Assigned( List ) then EXIT;
    List.Clear;
    ErrorCode := IpHlpNetworkParams (NetworkParams) ;
    if ErrorCode <> 0 then
    begin
        List.Add (SysErrorMessage (ErrorCode));
        exit;
    end ;
    with NetworkParams do
    begin
        List.Add( ' Ім'я хосту          : '  + HostName );
        List.Add( ' Домен            : '  + DomainName );
        List.Add( ' NETBIOS тип : '  + NETBIOSTypes[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 IpHlpNetworkParams (var NetworkParams: TNetworkParams): 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 := GetNetworkParams( Nil, @InfoSize );  // дані
    if result <> ERROR_BUFFER_OVERFLOW then exit ; // дані
    GetMem (FixedInfo, InfoSize) ;                 // дані
    try
    result := GetNetworkParams( FixedInfo, @InfoSize );   // дані
    if result <> ERROR_SUCCESS then exit ;
    NetworkParams.DnsServerTot := 0 ;
    with FixedInfo^ do
    begin
        NetworkParams.HostName := trim (HostName) ;
        NetworkParams.DomainName := trim (DomainName) ;
        NetworkParams.ScopeId := trim (ScopeID) ;
        NetworkParams.NodeType := NodeType ;
        NetworkParams.EnableRouting := EnableRouting ;
        NetworkParams.EnableProxy := EnableProxy ;
        NetworkParams.EnableDNS := EnableDNS ;
        NetworkParams.DnsServerNames [0] := DNSServerList.IPAddress ;  // дані
        if NetworkParams.DnsServerNames [0] <> ' '  then
                                    NetworkParams.DnsServerTot := 1 ;
        PDnsServer := DnsServerList.Next;
        while PDnsServer <> Nil do
        begin
            NetworkParams.DnsServerNames [NetworkParams.DnsServerTot] :=
                                    PDnsServer^.IPAddress ;  // дані
            inc (NetworkParams.DnsServerTot) ;
            if NetworkParams.DnsServerTot >=
                          Length (NetworkParams.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 виявлення аномалій
телекомунікаційного трафіку на основі спектрально-часового аналізу
(Preventivniy_Zahist_ot_Shkidlivih_Program)}
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
  IPNetRow      : TMibIPNetRow;
  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 := GetIPNetTable( 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 := GetIpNetTable( PTMIBIPNetTable( pBuf ), @TableSize, false );
  if ErrorCode = NO_ERROR then
  begin
    NumEntries := PTMIBIPNetTable( pBuf )^.dwNumEntries;
    if NumEntries > 0 then
    begin
      inc( pBuf, SizeOf( DWORD ) ); // беремо розмір останьої таблиці
      for i := 1 to NumEntries do
      begin
        IPNetRow := PTMIBIPNetRow( PBuf )^;
        with IPNetRow do
          List.Add( Format( ' %8x | %12s | %16s | %10s' ,
                            [dwIndex, MacAddr2Str( bPhysAddr, dwPhysAddrLen ),
                            IPAddr2Str( dwAddr ), ARPEntryType[dwType]
                            ]));
        inc( pBuf, SizeOf( IPNetRow ) );
      end;
    end
    else
      List.Add( '  ARP-кеш пустий.'  );
  end
  else
    List.Add( SysErrorMessage( ErrorCode ) );

  // необхідно відновити показник!
  finally
    dec( pBuf, SizeOf( DWORD ) + NumEntries -SizeOf( IPNetRow ) );
    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 виявлення
аномалій телекомунікаційного трафіку на основі спектрально-часового аналізу
(Preventivniy_Zahist_ot_Shkidlivih_Program); }
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( ' Переназначено            : '  + IntToStr( dwRedirects ) );
      ICMPIn.Add( ' Exo запит        : '  + IntToStr( dwEchos ) );
      ICMPIn.Add( ' Exo 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( 'Переназначено            : '  + IntToStr( dwRedirects ) );
      ICMPOut.Add( 'Exo 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 для виявлення аномалій телекомунікаційного трафіку на основі спектрально-часового аналізу (Preventivniy_Zahist_ot_Shkidlivih_Program)

```
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 досліджуємої мережі для виявлення аномалій телекомунікаційного трафіку на основі спектрально-часового аналізу (Preventivniy_Zahist_ot_Shkidlivih_Program)

```
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.
```