Міністерство освіти і науки України
Український державний університет науки і технологій

Факультет «Комп'ютерні технології і системи»

Кафедра «Комп'ютерні інформаційні технології»

Пояснювальна записка

до кваліфікаційної роботи бакалавра

на тему: «Розробка демонстраційної програми аналізу пакетів TCP протоколу»
за освітньою програмою: «121 Інженерія програмного забезпечення»
зі спеціальності: «121 Інженерія програмного забезпечення»
Виконав: студент групи «ПЗ1911»

(підпис студента) /Ілля ХОХЛОВ/
(Ім'я ПРІЗВИЩЕ)

Керівник: (підпис) /доц. Вадим АНДРОЩЕНКО/
(посада, Ім'я ПРІЗВИЩЕ)

Нормоконтролер: (підпис) /доц. Світлана ВОЛКОВА/
(посада, Ім'я ПРІЗВИЩЕ)

Засвідчую, що у цій роботі немає запозичень з праць інших авторів без відповідних посилань.
Студент (підпис)

Дніпро 2023 рік

Ministry of Education and Science of Ukraine
Ukrainian State University of Science and Technologies

Faculty «Computer technologies and systems»

Department «Computer information technology»

Explanatory Note

to Bachelor's Thesis

on the topic: «Mobile application for testing children's knowledge of mathematics»
according to educational curriculum «Software engineering»
in the Speciality: «121 Software engineering»

Done by the student of the group PZ1811: /Illia KHOKHLOV/

Scientific Supervisor: /Vadym ANDRUSHENKO/

Normative controller: /Svitlana VOLKOVA/

Dnipro – 2023

Міністерство освіти і науки України
Український державний університет науки і технологій

Факультет: Факультет «Комп'ютерні технології і системи»
Кафедра: «Комп'ютерні інформаційні технології»
Рівень вищої освіти: бакалавр
Освітня програма: «Інженерія програмного забезпечення»
Спеціальність: «121 Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ
Завідувач кафедри КІТ
_____/Вадим ГОРЯЧКІН/
(підпис)

Дата _____

З А В Д А Н Н Я

на кваліфікаційну роботу бакалавра
студенту Хохлову Іллі Володимировичу
1. Тема роботи: «Розробка демонстраційної програми аналізу TCP протоколів»
Керівник роботи: Андрющенко Вадим
Олександрович, доцентзатверджені наказом № 77 ст
від 08.12.2021
2. Строк подання студентом роботи: 16.06.2023р.
3. Вихідні дані до роботи: _____

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

Вступ

Аналіз проблеми трафіку мереж, основаних на протоколі TCP-IP

Проектні рішення при розробці програми

Реалізація у вихідних кодах програми аналізу пакетів TCP протоколу

Висновки

Перелік використаних джерел

Презентація

Відео роботи програми

Стадія	Зміст	Строки виконання
Технічне завдання	Постановка задачі, збір інформації, виріб та обґрунтування критеріїв розробки. Попередній вибір методів рішення задач. Визначення вимог до технічних засобів. Узгодження і затвердження технічного завдання.	31.01.23 -18.02.23
Робочий проект	Програмування та відлагодження програми.	19.02.23 - 20.05.23
	Тестування програми	20.05.23 - 27.05.23
	Розробка, узгодження і затвердження програмної документації.	27.05.23 - 12.06.23

Студент _____ Ілля Хохлов

(підпис) (Ім'я ПРІЗВИЩЕ)

Керівник роботи _____ доц. Вадим Андрющенко

(підпис) (Ім'я ПРІЗВИЩЕ)

РЕФЕРАТ

Пояснювальна записка складається з 7 розділів:

– вступ – в даному розділі описується сутність розробки, її актуаль-ність. Складається з 2 сторінок;

– Аналіз проблеми трафіку мереж, основаних на протоколі TCP-IP – у цьому розділі ми описуємо суть процесу аналізу трафіку, класифікації трафіку, аналізи недоліків сучасних систем зопобігання вторгненням; Складається з 21 сторінки

– Проектні рішення при розробці – у цьому розділі описується вибір мови програмування, розробка користувацького інтерфейсу, переваги та недоліки середовища розробки; Складається з 14 сторінок

– Реалізація у вихідних кодах програми аналізу пакетів TCP протоколу – у цьому розділі описується особливості вихідних кодів проектованої програми, тестування програми, інструкція користувача, використання програмного продукту; Складається з 13 сторінок.

– загальні висновки – підсумки всієї роботи. Складається з 1 сторі-нки;

– список використаних джерел – включає в себе бібліографічний сок використаної літератури. Складає 3 сторінок;

– додатки – містить лістинги програми

– .Кількість таблиць: 1 штук. Кількість рисунків: 26 штуки.

ЗМІСТ

ВСТУП

Аналіз мережевого трафіку – досить актуальна та важлива тема. Під «аналізом мережного трафіку» розумітимемо сукупну назву технологій та їх реалізацій, що дозволяють проводити накопичення, обробку, класифікацію, контроль та модифікацію мережевих пакетів залежно від їхнього вмісту в реальному часі. Одним із ускладнюючих факторів, при розгляді цього питання, є двоїстість розвитку засобів аналізу мережного трафіку: з однієї сторони – це розвиток алгоритмів та підходів до аналізу трафіку, з іншо- – розвиток програмних та апаратних засобів для більш ефективного рішення цього завдання. Це часто призводить як до плутанини у термінології, так і до свідомої маніпуляції фактами та цифрами у маркетингових цілях.

Актуальність обраної теми обумовлена тим, що на даний момент активно розробляються та застосовуються різні методи виявлення і запобігання вторгненням, але вони не завжди є ефективними на практиці. Внаслідок цього всі технології захисту постійно вивчаються та покращуються.

Існуючі системи поєднує загальна риса – захист локальної мережі від зловмисного впливу ззовні. У моїй роботі розглядається побудова аналітики внутрішнього трафіку таким чином, щоб на основі неї адміністратор міг ухвалити рішення про прийняття своєчасних дій і цим самим захистити зовнішню мережу від впливу з локальної мережі. Якщо поширити таку схему в роботі більшості підмереж, забезпечення безпеки мережевої інфраструктури вийде на новий рівень.

Основна мета дипломної роботи – аналіз проблеми трафіку мереж, основаних на протоколі TCP-IP. Для досягнення цієї мети передбачається виконання наступних кроків:

– Провести ретроспективний огляд розвитку прикладної сфери аналізу мережного трафіку розуміння історичного шляху цієї технології.

– Дослідити необхідність та оптимальність застосування тих чи інших підходів та алгоритмів для вирішення конкретних прикладних завдань.

– Виділити загальну схему аналізу, що використовується у переважній кількості конкретних систем аналізу мережного трафіку.

– Провести огляд поточного рівня технологій та аспектів їх прикладної реалізації.

– Провести дослідження існуючих рішень для аналізу трафіку в комп’ютерних мережах.

– Розробити програмне забезпечення для аналізу пакетів даних основаних на протоколі TCP-IP.

– Провести тестування розробленого програмного забезпечення.

– Розробити інструкцію користувача.

1 АНАЛІЗ ПРОБЛЕМИ ТРАФІКУ МЕРЕЖ, ОСНОВАНИХ НА ПРОТОКОЛІ TCP-IP

1.1 Суть процесу аналізу трафіку та галузі, де він застосовуватися

Найбільш повний розвиток технологія аналізу мережевого трафіку отримала, починаючи з другої половини 2000-х років, у зв'язку з кількома факторами:

– Безперервне зростання обсягів даних, що передаються.

– Зростання ширини каналів, які забезпечують можливості передачі цих об'ємів.

– Збільшення кількості різноманітності даних, що передаються, зокрема тих, які можуть використовуватися для складання різних профілів, як окремих користувачів, так і різних груп.

– Зростання як різноманітності мережних загроз та атак, так і їх кількісні характеристики.

Ці фактори призвели до зростання потреб з боку провайдерів інтернету (Internet Service Providers, ISP) та різних компаній. Інтереси цих груп різні, але водночас мають значні перетину. Так, наприклад, загальною сферою інтересів є захист мережевих ресурсів, який, у свою чергу, ділиться на низку напрямків:

– Антивірусні рішення (AV).

– Розвинені міжмережні екрани Next Generation Firewalls (NGFW).

– Системи виявлення та запобігання мережевим атакам Intrusion detection/prevention systems IDS/IPS.

– Системи захисту від DDoS-атак.

У той же час, специфічною галуззю інтересів провайдерів Інтернету є:

– Забезпечення якості зв'язку в години найбільшого навантаження (ЧПН) з урахуванням економії на розширенні каналів зв'язку, що орендуються.

– Отримання конкурентної переваги за рахунок можливості пропонувати вигідніші індивідуальні тарифи з урахуванням індивідуального профілю користування мережним каналом.

– Регулювання смуги пропускання деяких видів трафіку. Однією з основних проблем є P2P трафік, який може займати значну частину каналу, що орендується провайдером (до 60- 80%[1]), приводячи до того, що щоб забезпечити необхідну якість сервісу (quality of service, QoS) провайдеру доводиться прискореними (по порівняно з прогнозами зростання абонентської бази та потреб користувача) темпами розширювати цей канал.

Основною сферою інтересів компаній, що пропонують свої товари та послуги з використанням Інтернету, є «профілі» користувачів з погляду їхніх інтересів та переваг. Подібні профілі можна опосередковано виявити, зокрема, за допомогою списку сайтів, які користувач відвідує, набору пошукових запитів, мережевих додатків, які він використовує.

До іншої групи належать компанії, що надають різні інтернет-сервіси, наприклад, за допомогою технології віртуалізації мережевих функцій (Network Function Virtualization, NFV). До таких сервісів можна віднести:

– хмарні послуги;

– сервіси захисту;

– зберігання тощо.

Для цих компаній специфічним є питання управління великими обсягами вхідного трафіку — потрібне балансування та інтелектуальне управління.

Відповідно до наведеного вище історичним розвитком потреб у галузі мережевих сервісів відбувався розвиток технологій аналізу мережного трафіку, що лягають в основу апаратних, програмних та гібридних рішень.

Можна виділити два основні напрямки розвитку:

– Зростання «глибини» аналізу для окремого мережного пакета, тобто збільшення рівня моделі OSI, дані якого аналізуються.

– Повнота обліку стану потоку, до якого належить пакет, а також інших потоків, пов'язаних із даними.

1.1.1 Глибина аналізу мережевих пакетів

По цій осі технології аналізу трафіку розвивалися послідовно, кожна наступна успадкувала частину попередніх механізмів і додавала свої. Можна виділити три рівні розвитку технології, що наведені на рис. 1.1.

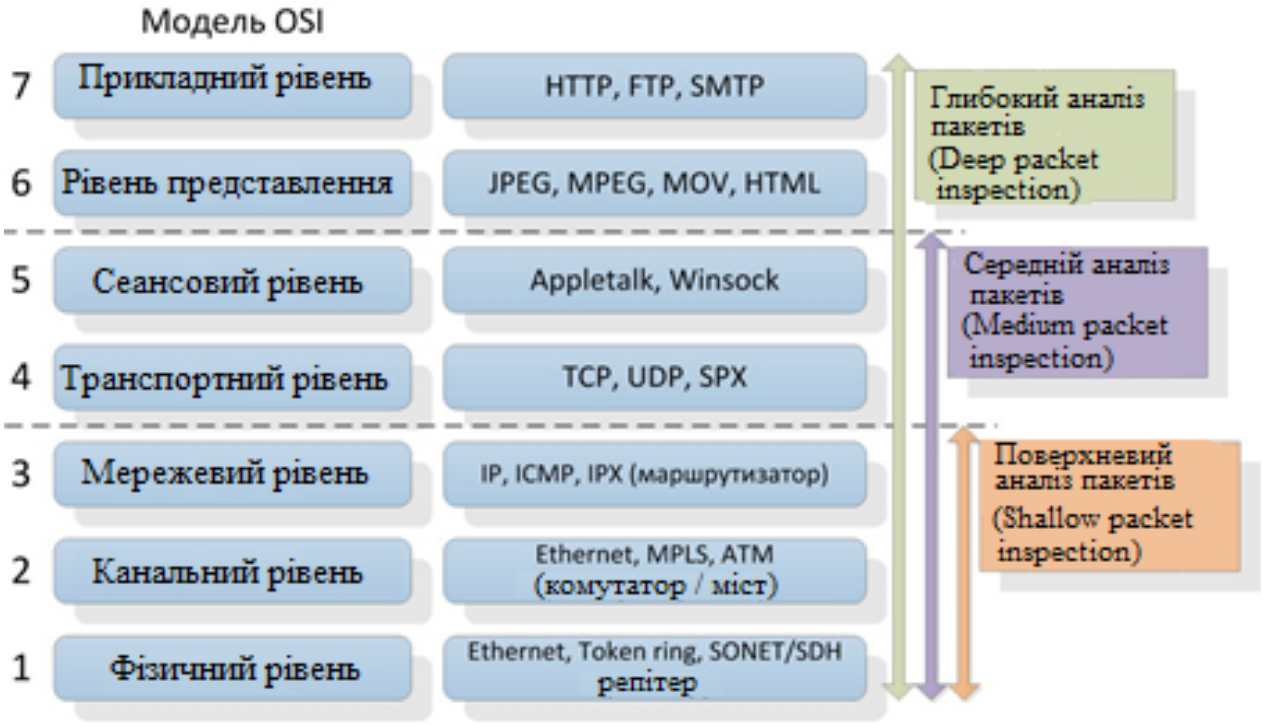


Рисунок 1.1 – Рівні розвитку технології аналізу мережевого трафіку за «глибиною»

Поверхневий аналіз пакетів (SPI)

Технологія аналізу трафіку, що ґрунтується виключно на заголовках пакета рівнів L1-L3 за моделлю OSI. Пред'являє низькі вимоги до обчислювальних ресурсів, що дозволяє аналізувати більші обсяги трафіку. Технологія поширена, на її основі працює більшість міжмережевих екранів операційних систем (зокрема в ОС Windows XP/Vista та OS X), маршрутизаторів та інших мережевих пристроїв. На основі реалізовані мережеві списки контролю доступу лише на рівні IP адрес і портів (Access Control List, ACL). Таким чином, ця технологія добре підходить для

розмежування доступу ззовні до окремих комп'ютерів (IP) та сервісів (портів) внутрішньої мережі.

Середній аналіз пакетів (MPI)

Технологія аналізу трафіку, що ґрунтується на інспектуванні сесій та сеансів зв'язку, ініційованих додатком, але встановлюваних шлюзом-посередником. Також застосовується термін "проксі додатків" (application proxy). В рамках даної технології вміст пакетів аналізується частково та за визначеними правилами.

Пристрої, що реалізують цей функціонал, розміщуються між провайдером інтернету і кінцевим користувачем. Дані пристрої розбирають заголовки до транспортного рівня та невелику частину даних пакета для зіставлення розібраної частини з деяким списком аналізу (parse list), з наступною реакцією в разі їх виявлення. Дані списки зазвичай коротші за списки ACL і надають ширший діапазон дій на відміну від «дозволити/заборонити» у випадку ACL. Ці списки також виразніші, оскільки дозволяють прив'язуватися не до IP-адрес, а до формату даних пакетів і даних деяких протоколів рівня програми, наприклад, URL-адрес у разі протоколу HTTP. За допомогою MPI можна, наприклад, заблокувати можливість отримання flash-файлів або картинок з певних інтернет-сервісів (на рівні представлення OSI) або заблокувати частину команд (на рівні програми OSI) в окремих протоколах. Набір протоколів зазвичай дуже обмежений. Наприклад, у перших версіях CheckPoint FireWall-1 (CheckPoint FW-1) підтримувалися протоколи Telnet, FTP, HTTP. Згодом ці набори трохи розширювалися. Також відомо, що ця технологія використовується в продуктах компаній McAfee та Symantec. Міжмережні екрани, що використовують цю технологію, відносяться до другого покоління [2].

Глибокий аналіз пакетів (DPI)

Іноді вживають більш тонкий термін — DPP (Deep Packet Processing), що передбачає наступні дії з пакетами, як модифікація, фільтрація або перенаправлення. Сьогодні обидва терміни часто використовуються як

взаємозамінні [3]. Ця технологія є логічним розвитком MPI. У межах цього підходу аналізатор переглядає вміст кожного пакета повністю. Однією з важливих відмінностей від попередніх технологій є те, що системи на базі DPI можуть приймати рішення не тільки за вмістом пакетів, але й за не прямими ознаками, притаманними певним мережевим програмам і протоколам. Для цього можна використовувати статистичний аналіз. Наприклад, аналіз частоти зустрічі певних символів, довжин пакетів, відстань між мітками часу послідовних пакетів тощо. Також, порівняно з попередніми підходами, значно розширено перелік застосувань технології: класифікація, обмеження смуги, пріоритезація, маркування, кешування тощо. Технологія DPI отримала розвиток, насамперед, через стрімке зростання обчислювальних можливостей процесорів, їх швидкодії і, можливостей для більш повного і точного аналізу мережевих даних.

На відміну від MPI, ця технологія спочатку розроблялася для високошвидкісної обробки та ідентифікації великої кількості програм у реальному часі. Таким чином, рішення на основі DPI добре масштабуються як по ширині мережного каналу (відомі рішення, що працюють на каналах близько 100 Гбіт/сек), так і за кількістю додатків, що ідентифікуються (в існуючих рішеннях — близько декількох тисяч). З погляду реалізації, основний компонент будь-якого рішення DPI – модуль класифікації, що відповідає за класифікацію мережевих потоків. При цьому в залежності від цілей застосування DPI класифікація може виконуватися з різною точністю:

– тип протоколу або програми (наприклад, Web, P2P, VoIP);
– конкретний протокол рівня програми (HTTP, BitTorrent, SIP);
– додаток, який використовує протокол (Google Chrome, µTorrent, Skype).

Важливо відзначити, що відповідність між класами різних рівнів точності не є однозначною, що показано на рис. 1.2.

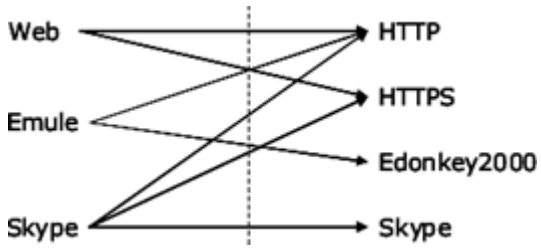


Рисунок 1.2 – Відмінність між ідентифікацією додатків (ліворуч) та протоколів (праворуч)

Технологія DPI на даний момент є поточним стандартом де-факто для засобів аналізу мережевого трафіку і відноситься до галузі критично важливих технологій, необхідних для забезпечення як мережевої безпеки, так і вимог законодавства. Внаслідок цього останнім часом на міжнародному рівні було прийнято низку стандартів, вимог та рекомендацій щодо особливостей реалізації, внутрішнього устрою та набору функцій відповідних засобів [4]. Ця технологія рідко застосовується в міжмережевих екранах - це скоріше область IDS/IPS систем, як винятки можна вказати екрани Hogwash і Shield. Проте міжмережеві екрани, що належать до четвертого покоління [5] можуть враховувати дані IDS/IPS систем у процесі аналізу.

1.1.2 Облік стану потоку під час аналізу мережевого трафіку

Другим напрямом розвитку технології аналізу можна назвати облік стану протоколу (потоку) у процесі аналізу – так звані stateless/statefull види аналізу. Цей напрямок є актуальним лише для протоколів, що використовують транспортний протокол із встановленням з'єднання (connection-oriented). Це означає, що перед будь-яким обміном командами і даними відбувається процес встановлення з'єднання, в ході якого сторони обмінюються фіксованою послідовністю пакетів, яка часто називається рукостисканням (handshake), а після завершення обміну відбувається аналогічний процес закриття з'єднання. До connection-oriented протоколів, зокрема, належить протокол TCP, але з UDP. Однак слід врахувати, що поверх UDP може бути реалізований інший транспортний протокол із встановленням з'єднання. Як приклад можна

навести протокол Quick UDP Internet Connections (QUIC) [6] — протокол транспортного рівня із встановленням з'єднання, що використовує UDP. З цього випливає, що, в загальному випадку, не можна повністю виключити статевий аналіз для UDP пакетів.

Для опису відмінностей описаних підходів потрібно дати визначення поняття «потік пакетів». Відомі різні визначення даного поняття. Частина з найбільш широко використовуваних наведена на сайті Center for Applied Internet Data Analysis (CAIDA)[23]. У цій роботі ми будемо використовувати «односторонній потік транспортного рівня» — послідовність пакетів, що передаються із заданої IP-адреси та TCP/UDP порту на дану IP-адресу та TCP/UDP порт, із зазначенням протоколу транспортного рівня (TCP/UDP). Таким чином, потік задається п'ятіркою "srcIP, srcPort, dstIP, dstPort, protocol". З урахуванням даного визначення, можна сформулювати відмінність statefull від stateless підходу. Воно полягає в тому, що у випадку статевогопідходу враховується той факт, до якого саме потоку відноситься аналізований пакет, і результат (стан) аналізу попередніх пакетів цього ж потоку, якщо даний пакет не перший. Якщо пакет перший – перевіряється, що він є коректним пакетом встановлення з'єднання.

Слід зазначити, що поняття «statefull» недостатньо чітке і може мати різні градації з різним «станом», що призводить до різного балансу точність аналізу/ресурсоємність/швидкість роботи [7]. Один із варіантів градації можна бачити на рис. 1.3.

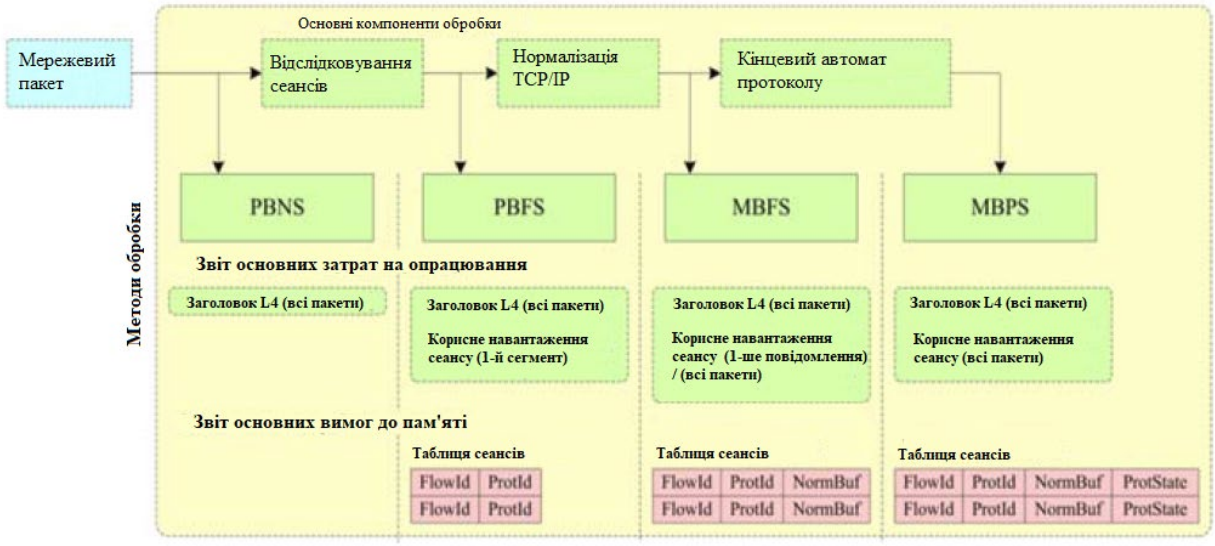


Рисунок 1.3 – Градації повноти обліку стану потоку

Список рівнів обліку стану потоку, який відображено на рис. 1.3 наступний:

– Аналіз окремих пакетів без урахування потоків та станів (Packet Based No State, PBNS).

– Аналіз пакетів у межах потоків (Packet Based Per Flow State, PBFS).

– Аналіз повідомлень у межах потоку (Message Based Per Flow State, MBFS), тобто. проведено складання IP-фрагментів в IP-пакети (IP-нормалізація) і складання TCP-сегментів в TCP-сеанси (TCP-нормалізація).

– Аналіз повідомлень у межах протоколу (Message Based Per Protocol State, MBPS), тобто. враховується стан автомата протоколу (можливість приймати той чи інший тип повідомлень). Приклад автомата станів протоколу HTTP наведено на рис. 5. Вершини відповідають станам, ребра — умовам переходу, до яких можуть належати прийом/надсилання повідомлення, результати обробки повідомлень, закінчення часу.

Базові реалізації технології DPI часто ставляться до stateless-аналізу, тобто аналіз виконується лише на рівні окремих пакетів, стан між аналізом кількох пакетів одного мережного потоку не зберігається. Цього рівня точності вистачає для багатьох практичних програм і дозволяє значно економити ресурси (див. рис. 1.3). У той самий час, існують завдання, котрим такого рівня точності мало. Як приклади можна навести дві технології, що

використовують statefull підхід - інспекція пакетів зі зберіганням стану (statefull packet inspection, SPI) і глибокий аналіз вмісту (deep content inspection, DCI).

1.1.3 Захоплення мережних пакетів

Програмні та апаратні засоби, що здійснюють захоплення трафіку, належать до класу сніферів (sniffers). Для вирішення задачі перехоплення трафіку можуть використовувати як стандартні мережні серверні карти, так і спеціалізовані мережні картки. З розвитком технологій багато з описаних властивостей реалізуються і основі стандартних мережевих карт. Технологія реалізації таких додаткових функцій називається TCP Offload Engine (TOE). Вона включає наступні різні технології, базовими з яких є:

– Large Segment Offload (LSO) або Giant send offload (GSO)— сегментація великих TCP-пакетів під час відправлення;

– Large Receive Offload (LRO) — складання окремих мережевих пакетів, що надходять, у великі сегменти;

– Checksum Offload — перевірка контрольних сум у заголовках IPv4, IPv6, TCP та UDP;

– IP Security (IPSec) Offload — шифрування/дешифрування трафіку протоколу IPSec.

Проблеми, що виникають при переході до мережевих з'єднань, що підтримують високі швидкості передачі даних, пов'язані в основному з кількома факторами [8]:

– Обмежена пропускна здатність апаратури.

– Архітектурними обмеженнями при взаємодії апаратури з ОС та ОС з додатками користувача.

– Об'ємом пам'яті, необхідним для зберігання даних. В основному систем аналізу трафіку працюють із використанням бібліотек Libpcap (Linux) і WinPcap (Windows). Дані бібліотеки працюють у режимі користувача та

використовують драйвери рівня ядра Berkeley Packet Filter та Netgroup Packet Filter.

Серед проблем цих рішень, що призводять до зниження продуктивності, можна виділити:

– Подвійне копіювання даних пакета (з картки в пам'ять ядра, з пам'яті ядра в пам'ять процесу користувача).

– Величезна кількість переривань від карти мережі (на кожний пакет, щоб вона була скопійована в буфер ядра).

– Велика кількість перемикань між режимами ядра та користувача (на кожний пакет при його копіюванні в пам'ять процесу користувача).

– Недостатнє використання паралелізму на рівні окремих ядер та процесорів (за замовчуванням усі переривання обробляються одним ядром).

– Проблеми із синхронізацією під час доступу до даних із кількох потоків виконання. Якщо отримані дані повинні оброблятися в кілька потоків між цими потоками виникає ситуація змагання за ресурси.

Залежно від кількості копій даних пакетів, які виконуються в процесі перехоплення, рішення поділяються наступним чином:

– 0-копія (zero-copy). Для реалізації підходу з нульовим копіюванням потрібна апаратна підтримка з боку мережевої картки – вона повинна містити власний DMA контролер, який копіює дані з картки на згадку про програму користувача, без додаткового копіювання через пам'ять ядра. Прикладом може бути бібліотека PF_RING ZC у зв'язку з мережевими картами Intel чи Napatech [9];

– 1-copy. Для реалізації цього підходу можливі кілька варіантів - розробка аналізатора на рівні ядра, що є дуже складним завданням або пряме відображення пам'яті ядра в пам'ять процесу користувача;

– 2 copy. Стандартне рішення на базі LibPcap чи WinPcap.

Для вирішення перелічених проблем реалізовано кілька спеціалізованих драйверів та мережевих стеків, які включають такі комерційні рішення як

Sniffer10G від Emulex та Myricom. Дані рішення використовують схему з кільцевого буферу як більш ефективну, також вони є досить оптимізованими для багатопроцесорної і багатоядерної техніки. Зокрема вони реалізують наступний функціонал:

– Обробка перехоплення пакетів з використанням великої кількості ниток виконання (одна нитка на вхідну чергу).

– Балансування навантаження між ядрами (одне ядро – одна вхідна черга).

– Пакетна фільтрація всередині мережі.

Для цих функцій використовується як апаратна підтримка з боку архітектури, і підтримка з боку ОС (спеціалізоване API). Серед технологій можна виділити такі:

– Набір близьких технологій Interrupt Moderation, Adaptive Interrupt Moderation, Interrupt Coalescing, Interrupt Blanking, Interrupt Throttling, що дозволяють керувати затримкою доставки переривань за рахунок таймера, що налаштовується, і обробляти отримання/відправлення безлічі пакетів за одне переривання.

– MSI-X — розподіл I/O переривань по кількох процесорах та ядрах.

– New API (NAPI) — інтерфейс рівня ядра ОС Linux, що дозволяє використовувати техніку зменшення кількості переривань (interrupt mitigation) із боку мережевих пристроїв.

– Receive-side Scaling (RSS) — технологія, що надає можливість динамічного балансування навантаження вхідних мережних пакетів по кількох ядрах та процесорах (переривання надходять на різні процесори). Існують реалізації для масштабування на випадки понад 64 процесори. Ця технологія підтримується в сімействі Windows з появою Scalable Networking Pack. У ОС Linux аналог цієї технології називається Linux Scalable I/O.

Також існує низка апаратних технологій від різних виробників процесорів, призначених для прискорення вводу/виводу:

– Intel Integrated I/O – технологія прямого підключення шини PCI Express 3.0 до процесора (без окремого PCI-контролера), реалізована у сімействі Intel Xeon E5.

– Direct Cache Access (DCA) – надання пристроям вводу/виводу, таким як мережні адаптери, можливості розміщення даних безпосередньо в кеш процесора Intel.

1.1.4 Класифікація мережного трафіку

Тема класифікації мережного трафіку як така дуже обширна. Тому, необхідно перерахувати варіанти класифікації за її результатами, тобто об'єкти, що виходять із даного алгоритму, їхніми властивостями та можливістю подальшого їх опрацювання. Отже, виділено такі основні класифікації:

1. Тип трафіку не є досить вичерпним способом класифікації і, як правило, або не піддається подальшому аналізу, або піддається досить простою додатковою класифікацією, що уточнює. Залежно від сфери застосування типи можуть бути різними. Серед прикладів можна зазначити:

– P2P, відео-стрімінг, веб-трафік — у разі систем збирання статистики та моніторингу;

– трафік мережевої атаки/нормальний трафік — у разі захисту від мережевих атак;

– трафік, що містить/не містить об'єкти копірайту, у разі систем контролю копірайту.

2. Протокол прикладного рівня (ППР) є досить змістовним, тому його можуть використовувати системи збору статистики та моніторингу для підвищення рівня точності. Основним способом подальшого опрацювання є розбір протоколу, який містить 2-ві основні функції - складання сесії прикладного рівня, та вилучення даних протоколу з окремих його полів в разі необхідності (мета інформація рівня протоколу).

3. Програма передачі даних (application identification), дає найбільш детальний рівень класифікації. На цьому рівні здійснюються ті ж види опрацювання, що і на рівні ППР, а також отримувати та інтерпретувати дані (метаінформацію) конкретного додатка, що відповідає вищому рівню їх подання.

У різних прикладних задачах результати ідентифікації протоколів і додатків можуть інтерпретуватися і, відповідно, піддаватися різній подальшій обробці (як і разі ідентифікації типу трафіку).

Наприклад, у разі захисту від шкідливого коду, під протоколом може розумітися командний (command-and-control, C&C) протокол ботнета, а під додатком — конкретний вірус. Відповідно, вилучена метаінформація - команди ботнета, дані, що передаються їм, а мета аналізу – з'ясування його функціоналу, оцінка поширеності та дослідження можливостей його деактивації.

У разі системи складання профілю користувача для подальшої демонстрації таргетованої реклами (наприклад, iMarker) у ролі протоколу може виступати HTTP, у ролі програми – браузер, а об'єктом аналізу є запит користувача до пошукової системи, який піддається подальшому текстовому аналізу для отримання ключових слів. Вибір конкретної прикладної задачі може значно впливати як на вибір алгоритму класифікації, так і на його параметри та продуктивність.

1.2 Існуючі засоби аналізу трафіку в комп’ютерних мережах

1.2.1 Класифікація засобів моніторингу та аналізу

Інструменти, які пропонуються для моніторингу та аналізу обчислювальних мереж, можна поділити на декілька груп [10]:

Системи управління мережею (Network Management Systems) - централізовані системи, що збирають дані про стан мережевих пристроїв та інформацію про трафік у мережах. Функціональні можливості таких програм

не обмежуються моніторингом та аналізом мережі. Додатково, в напівавтоматичному або автоматичному (залежно від реалізації) режимі, здійснюються дії з управління мережею: налаштування та зміна адресних таблиць комутаторів та іншого обладнання, увімкнення та вимкнення портів пристроїв. До систем цієї категорії відносяться HPOpenView, SunNetManager, IBMNetView.

Вбудовані системи діагностики та управління (Embedded systems). Системи цього типу виконані як програмно-апаратних модулів, які встановлюються в комунікаційне устаткування, чи – операційну систему як програмних модулів. Вони дозволяють керувати та діагностувати лише тим пристроєм, на якому знаходяться. Прикладом таких систем є модуль керування концентратором Distributed 5000, який виконує функції автосегментації портів після виявлення несправності, приписування портів внутрішнім сегментам концентратора та інші. Зазвичай, вбудовані модулі управління виконують роль SNMP-агентів, передаючи дані про стан пристрою в систему управління.

Засоби керування системою (System Management). Інструменти цієї групи виконують функції, аналогічні функцій систем управління, але стосовно іншим об'єктам. У першому випадку об'єктами управління є програмні та апаратні забезпечення комп'ютерних мереж, а в другому – комунікаційне обладнання. При цьому частина функцій цих двох видів систем може дублюватися.

Аналізатори протоколів (Protocol analyzers) – це програмні чи апаратно-програмні системи, використовувані лише моніторингу та аналізу трафіку в мережах. Класним аналізатором вважається той, що вміє захоплювати та декодувати пакети великої кількості протоколів, що застосовуються у мережах – приблизно кількох десятків. Дана група має можливість встановити необхідні логічні умови для захоплення окремих пакетів та виконувати їх декодування із розшифровкою змісту кожного поля пакета.

Коли починають проектувати або модернізувати мережу, часто виникає

потреба в кількісному вимірі характеристик мережі: наприклад, затримки, що виникають на різних етапах, частота виникнення вибіркових подій, інтенсивність потоків даних по лініях зв'язку, час реакції на запити.

Обладнання для діагностики та сертифікації кабельних систем. Умовно можна виділити 4 підтипи такого обладнання: кабельний сканер, мережевий монітор, мультиметр та прилад сертифікації кабельної системи.

Експертні системи поєднують людські знання про виявлення причин аномальної роботи мереж та можливі способи повернення мережі в робочий стан. Як правило вони представляються у вигляді окремих підсистем інших засобів моніторингу та аналізу мереж, розглянутих раніше.

Багатофункціональний пристрій аналізу та діагностики. Через широке поширення локальних мереж виникла потреба у розробці недорогих портативних приладів із функціоналом кількох пристроїв: кабельних сканерів, програм мережного управління та аналізаторів протоколів. Як приклад можна навести Compas компанії MicrotestInc або 675 LANMeter компанії FlukeCorp.

Також варто виділити ще два способи моніторингу мережі:

– маршрутизатор-орієнтований (вбудований безпосередньо в маршрутизатор та не потребує додаткової установки іншого забезпечення);

– не орієнтований на маршрутизатори (підібране фахівцем необхідне апаратне та програмне забезпечення в залежності від потреби).

1.2.2 Системи виявлення та запобігання вторгненням

1.2.2 Системи виявлення та запобігання вторгненням

Впровадження подібних систем захисту інформації є необхідністю всім серйозних мережевих інфраструктур, оскільки існують програми, які постійно вишукують уразливості у будь-якому устаткуванні, підключеному до глобальної мережі. Наприклад, пошуковий двигун Shodan [11] в автоматичному режимі збирає інформацію про підключені пристрої, які не мають будь-якої частини системи безпеки. Користувачі Shodan знаходять

системи керування крематорієм, газовою станцією тощо, які не мають реквізитів доступу або вони налаштовані за умовчанням. Отже, до них можна легко проникнути та зменшити працездатність.

Проти такого впливу і спрямовані системи виявлення та запобігання вторгненням, тому вони є часто використовуваним інструментом у безпеці [12].

Система виявлення вторгнень (СВВ) (англ. Intrusion Detection System (IDS)) – програмний чи апаратний засіб, призначений виявлення фактів неавторизованого доступу (вторгнення чи мережевої атаки) в комп'ютерну систему чи мережу.

Система запобігання вторгненням (СЗВ) (англ. Intrusion Prevention System (IPS)) – програмний або апаратний засіб, що здійснює моніторинг мережі або системи в реальному часі з метою виявлення, запобігання або блокування шкідливої активності.

Системи запобігання вторгнень можна вважати розширенням систем виявлення вторгнень, оскільки завдання відстеження атак залишається однаковим. Але СЗВ має відслідковувати вторгнення в реальному часі і одразу здійснювати дії щодо запобігання атакам. Для цього вони використовують: скидання з'єднань, блокування потоків трафіку в мережі, видачу сигналів оператору. Крім цього, такі системи можуть дефрагментувати пакети, змінювати порядок TCP пакетів для захисту від пакетів зі зміненими SEQ і ACK номерами тощо [13].

Дані системи використовуються для автоматизації процесу контролю над подіями, що протікають у комп'ютерній системі чи мережі, та аналізу цих подій з метою пошуку ознак проблем безпеки. Оскільки кількість різних способів та видів організації несанкціонованих вторгнень у мережі останнім часом значно збільшилася, то системи виявлення вторгнень стали обов'язковою частиною безпекової інфраструктури для більшості організацій. Цьому сприяють як велика кількість літератури з цього питання, яку потенційні зловмисники уважно вивчають, так і більш витончені підходи до

виявлення спроб проникнення в інформаційні системи.

Сучасні системи виявлення вторгнень мають різну архітектуру, основними з яких є: мережна та локальна. Мережеві системи встановлюють на виділених для цього комп'ютерах так, щоб вони могли аналізувати трафік, що протікає по локальній обчислювальної мережі. Локальні системи розміщуються на тих комп'ютерах, які потребують захисту, і вивчають певні події (програмні виклики або дії користувача).

Крім архітектури СВВ також можуть розрізняти за методикою виявлення: частина систем шукає аномальну поведінку, інша - зловмисне [14].

1.2.3 Методики виявлення аномальної та зловмисної поведінки користувачів

Системи виявлення аномальної поведінки (від англ. anomaly detection) засновані на тому, що СОВ відомі ознаки, що характеризують правильну або допустиму поведінку об'єкта спостереження. Під «Нормальною» або «правильною» поведінкою розуміються дії, що виконуються об'єктом і не суперечать безпековій політиці [15].

Системи виявлення зловмисної поведінки (misuse detection) засновані на тому, що наперед відомі ознаки, що характеризують поведінку зловмисника. Найбільш поширеною реалізацією технології виявлення зловмисної поведінки є експертні системи (наприклад, системи Snort, RealSecure IDS, Enterasys Advanced Dragon IDS).

Структурна схема технології, які у даних системах зображена на рис. 1.4.

Рисунок 1.4 – Існуючі технології СВВ

1.2.4 Аналіз недоліків сучасних систем виявлення вторгнень

Всі системи виявлення вторгнень можна поділити на системи, орієнтовані на пошук:

- сигнатури відомих атак;
- аномальну взаємодію контрольованих об'єктів;
- спотворення еталонної профільної інформації.

В даний час майже відсутні системи гібридного типу, а також системи, що використовують інформацію розподіленого у часі та просторі характеру. Більшість систем використовує лише сигнатурний метод розпізнавання атакуючих впливів або лише пошук аномалій у поведінці контрольованої мережі.

Також у багатьох відомих систем відсутній імітатор атак або будь-який інший засіб для перевірки коректності розгорнутої та експлуатованої сервісно-орієнтованої архітектури (СОА). Це дало б можливість забезпечити більш простий та надійний засіб тестування конфігураційних параметрів, в залежності до вимог комп'ютерної мережі.

Також цікавою розробкою є багатоагентні системи, але у вітчизняних роботах немає вказівок на алгоритми виявлення атак, що використовуються

або розроблені. А також поточні версії відомих імітаторів не функціонують у реальному режимі часу (оскільки це не дозволяє робити обраний базовий інструментарій).

Загалом відсутність імітаторів атак для оцінки ефективності СОА не є основною проблемою даного напряму. Найбільшими недоліками таких систем виявлення є примітивність простого сигнатурного пошуку, низька ефективність при виявленні розподілених за часом і місцем складних атак, недостатня інтегрованість інформації на рівні хосту та мережі для виявлення комбінованих атак та несанкціонованих проникнень.

В якості експлуатаційних недоліків можна відмітити велику кількість обчислювальних операцій для простого поділу належності події на «свій-чужий» і неможливість обробки всієї інформації, що надходить в реальному режимі часу на звичайних персональних комп'ютерах, так як швидкість обробки мережевого або іншого трафіку подій часто повільніше реального часу в 1.5-2 рази. Тому у деяких системах аналіз відбувається у відкладеному режимі. Це може означати, що реалізація атаки на інформаційні та обчислювальні ресурси, що захищені, не буде помічена вчасно і вже тим більше не буде відображена за допомогою наявних засобів захисту. У цьому режимі засоби виявлення атак можуть бути використані в кращому випадку як засіб занесення в журнал всіх етапів атаки для подальшої експертизи.

Більшість сучасних СОА не розробляються для роботи на різних операційних системах та довільних апаратно-обчислювальних платформах. Тому робота на кількох операційних системах більшості продуктів (як західних, і вітчизняних) є неможливою. Ці системи не використовують переваги розробки та оптимізації коду для вибраних операційних систем та апаратних платформ, що є їх одним із найістотніших недоліків.

Також в жодній програмній чи апаратно-програмній системі не передбачено режиму «гарячої заміни», що дозволило б у разі виведення з ладу основного комплексу швидко ввести в роботу комплекс резервування та відновити знищений рубіж оборони мережевого периметра.

Незважаючи на це, є позитивний момент у розвитку систем виявлення аномалій – прагнення розробників інтегрувати свої системи з існуючими засобами захисту (міжмережеві екранани, блокатори каналів, QoS-диспетчери).

1.3 Технічне завдання на розробку власного програмного продукту для аналізу пакетів TCP протоколу

Огляд сучасних підходів та засобів до проектування та розроблення програмного забезпечення дозволив обрати для створення власної системи ефективні технології та інструментальні засоби: IDE – Visual Studio 2019; Мова програмування – C#.

На основі виконаного аналізу предметної області та визначення недоліків систем виявлення вторгнень можна сформулювати постановку задачі.

Розробити систему, яка забезпечить захоплення 100% трафіку та надасть ефективні методи аналізу з навігацією за результатами.

Розробити зручний графічний інтерфейс для роботи з програмою. Основні дії та взаємодія між користувачем та системою повинні супроводжуватися відповідними повідомленнями для користувача.

Створити навігаційне меню для можливості швидкого та зручного отримання доступу до потрібної функції в системі.

Розробити супровідну документацію до створеної системи.

Дипломна роботи припускає розробку додатку засобами об’єктно-орієнтованого середовища програмування .

Розробка та реалізація додатку, що включає 3 основні модулі:

1) Модуль авторизації користувачів програми;

1) модуль вводу/редагування інформації;

2) модуль аналізу трафіку пакетів.

Реалізація додатку виконується з використанням технологій .NET. Результат –Windows додаток.

Загальні характеристики:

Під час роботи з програмою повинні виконуватися основні функції, а саме:

– додаватися, редагуватись та видалятись дані про користувачів системи;

– аналізуватись результат сканування трафіку на виявлення аномальної поведінки ;

– повинна бути можливість збереження необхідної інформації.

Загальні операції системи:

– реакція програми на вибір меню користувача;

– реакція програми на натискання кнопок в програмі;

– реакція програми на введення неправильних даних.

Функціональні вимоги:

– можливість додавати, редагувати та видаляти дані про користувачів системи;

– можливість перехоплення трафіку мережевих пакетів для виявлення аномальної поведінки;

– можливість зберігання необхідної інформації.

Нефункціональні вимоги:

– для роботи програми на комп’ютері повинна бути встановлена бібліотека класів .NET framework 4.5;

– для забезпечення роботи програми потрібно мати лише клавіатуру та мишку.

1.4 Висновок

У першому розділі обґрунтовано актуальність проблеми аналізу мережевого трафіку для захисту від несанкціонованого впливу та розглянуто існуючі способи вирішення цієї проблеми. Останнім часом у сфері систем управління спостерігаються дві досить чітко виражені тенденції:

– Інтеграція в одному продукті функцій управління мережами та системами;

– розподіл системи управління, при якій існує кілька консолей, що збирають інформацію про стан пристроїв і підсистем, а потім видають керуючі дії.

Це з тим, більшість представлених систем вузькоспеціалізовані і спрямовані хороше виконання свого функціоналу. Відповідно до фахівців доводиться використовувати зв'язку з кількох продуктів, щоб повністю покрити всі можливі вразливості мережі: системи моніторингу та обліку трафіку перевіряють працездатність обладнання та мережі, виявляють оптимальність поточної апаратно-програмної конфігурації; системи виявлення та запобігання вторгненням – дозволяють виявити загрозу всередині та зовні мережі.

Сучасний підхід до побудови систем виявлення мережевих вторгнень та виявлення ознак комп'ютерних атак на інформаційні системи сповнений недоліків та вразливостей, що дозволяють, на жаль, зловмисним впливам успішно долати рубежі захисту інформації.

На ринку представлено близько десятка систем запобігання та виявлення вторгнень [16], але всі вони мають один важливий недолік: оскільки їхня робота побудована на відпрацюванні правил і шаблонів, не всі випадки вторгнень уловлюються; крім того, вони слабо справляються з шкідливим трафіком.

Крім цього, варто зазначити, що продуктів вітчизняних виробників на

даному ринку представлено мало. Також, описані раніше методики який завжди доступні, оскільки є комерційної таємницею [17]. Тому виникає необхідність у розробці нової системи, яку можна застосовувати у комплексі мережевих інструментів.

За підсумками першого розділу було виконано перше завдання роботи – вивчити проблеми забезпечення безпеки мережі та дослідити сучасні методики аналізу мережевого трафіку.

2 ПРОЕКТНІ РІШЕННЯ ПРИ РОЗРОБЦІ ПРОГРАМИ АНАЛІЗУ ПАКЕТІВ TCP ПРОТОКОЛУ

2.1. Вибір мови програмування

Для побудови користувацького інтерфейсу будемо використовувати WinForms та мову програмування C#.

C# широко використовується професіоналами для розробки великих програмних продуктів завдяки наступним аспектам мови [18]:

– Сучасна мова програмування загального призначення.
– Підтримка об’єктно-орієнтованої парадигми.
– Підтримка компонентно-орієнтованої парадигми.
– Мова легка для вивчення навчитися.
– Добре структурована.
– Дозволяє розробляти ефективні програми.
– Мова має підтримку різних комп'ютерних платформах.
– Це частина .Net Framework.

C# спроектовано таким чином, що мова відповідає традиційним мовам високого рівня, C та C++ і є об'єктно-орієнтованою мовою програмування. Мова дуже схожа на Java, має численні сильні функції програмування, які роблять його привабливим для багатьох програмістів у всьому світі.[18]

.NET Framework - це платформа для розробників із відкритим кодом, яку можна використовувати для створення широкого кола програм. Цей безкоштовний крос-платформний фреймворк підтримує декілька мов і має великі бібліотеки коду, які спрощують створення додатків для мобільних пристроїв, робочих столів та Інтернету. [19]

Платформа .NET була розроблена для досягнення наступних цілей: [20]:

– Сумісність;
– Підтримка різних платформам;
– Мовна незалежність;
– Бібліотека базових класів;
– Легка розробка;
– Безпека.

Для розробки користувацького інтерфейсу платформа .Net має декілька технологій, одна з яких – WinForms. Не дивлячись на те, що ця технологія досить не нова, її важко назвати застарілою. Вона надає широкий спектр різних інструментів для побудови зручного та сучасного інтерфейсу. Крім того, IDE, які підтримують C# та .Net, надають зручний інтерфейс для графічної побудови користувацького застосунку, який розробляється.

Отже, C# та платформа .Net має низку характеристик, які задовольняють вимоги щодо розробки клієнтської частини системи складського обліку. Набір готових класів у стандартній бібліотеці, лаконічний зрозумілий синтаксис мови та зручний конструктор користувацького інтерфейсу зробить розробку зручною та достатньо швидкою. Об’єктно-орієнтована парадигма дозволить спроектувати систему таким чином, що розширення функціоналу буде без накладних розходів ресурсів розробки. Платформа .Net забезпечить безпеку, ефективність програмного забезпечення, а також підтримку декількох платформ.

Для розробки інформаційної бази використовувався Microsoft Access. MS Access – це система управління базами даних, програма, що входить до складу пакету офісних програм Microsoft Office [17].

Переваги використання [22]:

– простий графічний інтерфейс, який дозволяє створити власну базу даних вбудованими засобами;

– зберігає всі дані у одному файлі, хоч і розподіляє їх у різних таблицях, як і належить реляційної СУБД. До цих даних належить як інформація у таблицях, а й інші об'єкти бази даних;

– поширеність, яка зумовлена тим, що Access є продукт компанії Microsoft;

– постійно оновлюється виробником, підтримує безліч мов;

– повністю сумісний з операційною системою Windows;

– орієнтованість на користувача з різною професійною підготовкою, що виявляється у наявності великої кількості Майстрів, розвинену систему довідки та зрозумілий інтерфейс;

– широкі можливості імпорту/експорту даних у різні формати, від таблиць Excel і текстових файлів, до практично будь-який серверної СУБД через механізм ODBC;

– Наявність розвинених інтегрованих засобів розробки додатків. Більшість програм, що розповсюджуються серед користувачів;

– Наявність вбудованої мови макрокоманд.

Недоліки:

– обмежені можливості щодо забезпечення розрахованого на багато користувачів середовища;

– має нескладні способи захисту з використанням пароля БД (можливе застосування додаткових заходів щодо захисту від несанкціонованого доступу з використанням процедур VBA);

– в питаннях підтримки цілісності даних відповідає лише моделям БД невеликої та середньої складності;

– Не розповсюджується безкоштовно.

Отже СУБД Access було мною вибрано туму, що розроблена програма може легко переноситись на будь-які системи сімейства Windows і не потребує додаткового встановлення сервера бази даних. Реляційність бази даних дає можливість ефективно використовувати пам’ять та уникнути дублювання інформації. Завдяки відкритому коду не потрібно витрачати зайві ресурси на різні ліцензії, тобто розробка та використання системи на базі СУБД Access значно дешевша. Підтримка проекту відбувається вже багато років розробниками з усього світу, через що надійність та безпека продукту на високому рівні. Таким чином, СУБД Access повністю покриває вимоги щодо розробленого проекту.

Для роботи з Базою даних реалізовуємо класи (рис. 2.1). В кожному класі є методи, за допомогою яких можна: вибирати, вставляти, редагувати та видаляти дані. Також є методи, які працюють з вибіркою даних із декількох таблиць.

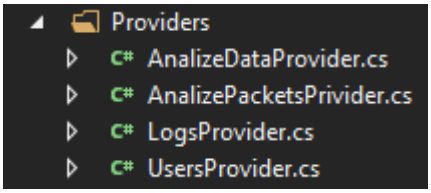


Рисунок 2.1 – Класи для роботи з БД

2.2 Середовище розробки Visual Studio

Microsoft Visual Studio - повнофункціональне інтегроване середовище розробки (IDE) за допомогою популярних мов програмування, серед яких C, C++, VB.NET, C#, F#, JavaScript, Python (рис. 2.2).

Функціональність Visual Studio охоплює всі етапи розробки програмного забезпечення, надаючи сучасні інструменти для написання коду, проектування графічних інтерфейсів, складання, налагодження та тестування програм. Можливості Visual Studio можуть бути доповнені шляхом підключення потрібних розширень.

Редактор коду Visual Studio підтримує підсвічування синтаксису, вставку фрагментів коду, відображення структури та пов'язаних функцій. Істотно прискорити роботу допомагає технологія IntelliSense – автозавершення коду у міру введення з клавыатури символів.

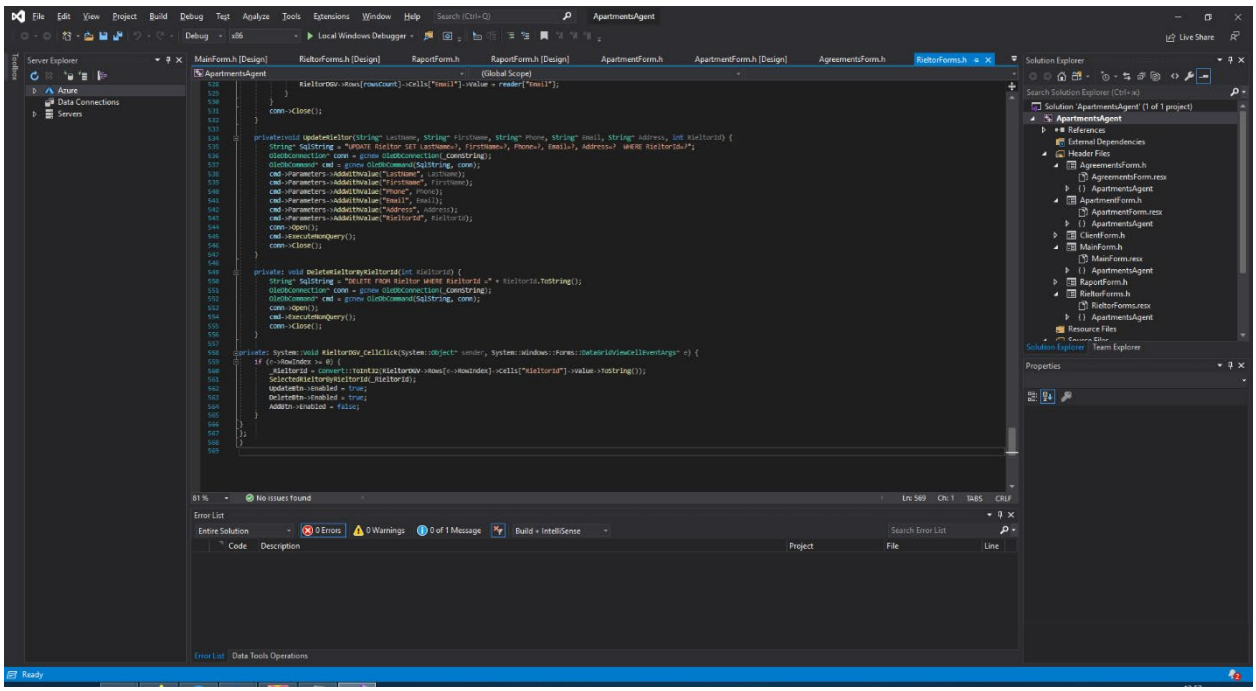


Рисунок 2.2 – Інтерфейс середовища розробки Visual Studio

Вбудований налагоджувач Visual Studio використовується для пошуку та виправлення помилок у вихідному коді, у тому числі на низькому апаратному рівні. Інструменти діагностики дозволяють оцінити якість коду з погляду продуктивності та використання пам'яті.

Дизайнер форм Visual Studio незамінний при розробці програм з графічним інтерфейсом, допомагаючи спроектувати зовнішній вигляд майбутньої програми та роботу кожного елемента інтерфейсу.

Для командних проектів Visual Studio пропонує підтримку групової роботи, дозволяючи виконувати спільне редагування та налагодження будь-якої частини коду в реальному часі, а як систему управління версіями використовувати Team Foundation або Git.

Основним розширенням файлу, асоційованим з Microsoft Visual Studio, є SLN - Visual Studio Solution File (Файл рішення Visual Studio), при відкритті якого в програму завантажуються всі дані та проекти, пов'язані з програмним рішенням, що розробляється.

Функціональна структура середовища включає:

– редактор вихідного коду, який включає безліч додаткових функцій, як автодоповнення IntelliSense, рефракторинг коду тощо;

– налагоджувач коду;

– редактор форм, призначений для спрощеного конструювання графічних інтерфейсів;

– веб-редактор;

– дизайнер класів;

– дизайнерсхем баз даних.

2.3 Переваги та недоліки середовища розробки

Інтегроване середовище розробки (IntegratedDevelopmentEnvironment - IDE) VS пропонує ряд високорівневих функціональних можливостей, що виходять за рамки базового керування кодом.

Нижче наведено основні переваги IDE-середовища VS:

– Підтримка багатьох мов під час розробки.

– Менше за код для написання. Для створення більшості програм потрібна пристойна кількість стандартного стереотипного коду та Web-сторінки ASP. NET тому не виняток.

– Інтуїтивний стиль кодування. За замовчуванням VS форматує код у міру його введення, автоматично вставлює необхідні відступи та застосовує колірне кодування для виділення елементів типу коментарів.

– Висока швидкість розробки. Наприклад, функція IntelliSense (вміє перехоплювати помилки та пропонувати правильні варіанти), функції пошуку та заміни (дозволяють шукати ключові слова як в одному файлі, так і у всьому проекті) та багато ін.

– Можливості налагодження. Інструменти налагодження у VS є є дуже зручними для відстежування помилок та діагностики дивної поведінки.

– Простий синтаксис язика забезпечує низький поріг входження.

– Можливість компіляції як у машинний код, так і в P-код.

– Безпека типів забезпечує захист від помилок, пов'язаних із застосуванням покажчиків та доступом до пам'яті.

– Можливість використання більшості функцій WinAPI [18] для розширення функціональних можливостей програми.

Недоліки:

– Підтримка операційних систем лише сімейства Windows та Mac OS X (Виняток - VB1 for DOS).

– Відсутність повноцінного механізму наслідування реалізації об’єктів. Існуюче у мові успадкування дозволяє успадковувати лише інтерфейси, але з їх реалізацію.

– Практично всі вбудовані функції мови реалізовані через бібліотеку часу виконання, яка, своєю чергою, робить багато «додаткової» роботи з перевірки та/або перетворення типів, що уповільнює швидкість роботи програм.

2.4 Архітектура проекту

Розроблений продукт повинен відповідати характеристикам якості, таким як: стійкість, корисність, доступність, масштабованість, відкритість, гнучкість, можливість тестування. Це вимагає від процесу розробки додаткові обмеження/правила, а саме:

– дотримання шаблонів і стилів;

– документування розробки на різних рівнях;

– тестування компонентів, окремих модулів, підсистем;

– управління проектами, процесами.

З урахуванням вимог до забезпечення стійкості та гнучкості системи при її розробці було обрано шаблон Layers, який розбиває систему на дві частини: клієнт та сервер.

Проектування системи буде покладатись на предметну область (DDD

підхід) та принципи SOLID [25].

При розробці клієнта був обраний користувацький інтерфейс Windows Forms.

Сервер, в свою чергу, буде складатись з таких модулів:

1) BLL (англ. Business Logic Layer) – логіка та всі необхідні обчислення додатку на мові бізнесу;
2) DAL (англ. Data Acces Layer) – рівень доступу до даних.
3) DB (англ. Data Base) – база даних для зберігання даних.

З урахування всіх вище перерахованих шаблонів структура проекту буде виглядати наступним чином:

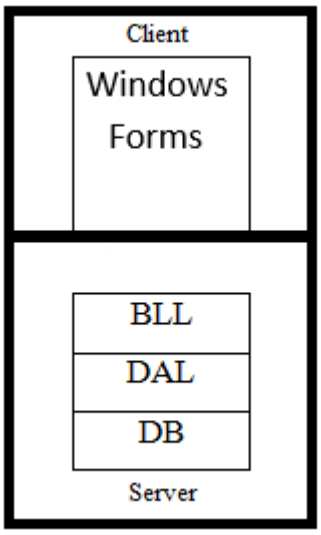


Рисунок 2.3 – Структура проекту

Кожен компонент буде реалізувати контракт (інтерфейс), який надає на гнучкість компоненту.

2.5 Діаграми класів проекту

2.5.1 Розробка користувацького інтерфейсу

В даному проекті для розробки користувацького інтерфейсу було використано Windows Forms.

Windows Forms - це платформа користувача інтерфейсу для створення класичних додатків Windows. Вона забезпечує один з найефективніших способів створення класичних додатків за допомогою візуального

конструктора в Visual Studio.

User interface (UI) елементи – це частини, які дизайнери використовують для створення програм або веб-сайтів. Вони додають інтерактивність в інтерфейс користувача, надаючи користувачеві точки зіткнення при навігації по них. Кнопки, смуги прокручування, пункти меню та чекбокси.

Інтерфейсу користувача (UI) використовує UI елементи для створення візуальної мови і забезпечення узгодженості продукту, що робить його зручним для користувача і простим у навігації без особливих зусиль з боку користувача.

UI елементи зазвичай поділяються на одну з наступних чотирьох категорій:

– Елементи керування введенням (Input Controls) – дозволяють користувачам вводити інформацію до системи.

– Компоненти навігації (Navigation Components) – допомагають користувачам переміщатися продуктом або веб-сайтом. Загальні навігаційні компоненти включають панелі вкладок та головне меню програми.

– Інформаційні компоненти (Informational Components) – діляться інформацією з користувачем.

– Контейнери (Containers) – містять зв’язаний контент разом.

На рис. 2.4 показана діаграма програми «TCP/IP аналізер» з правами адміністратора системи.

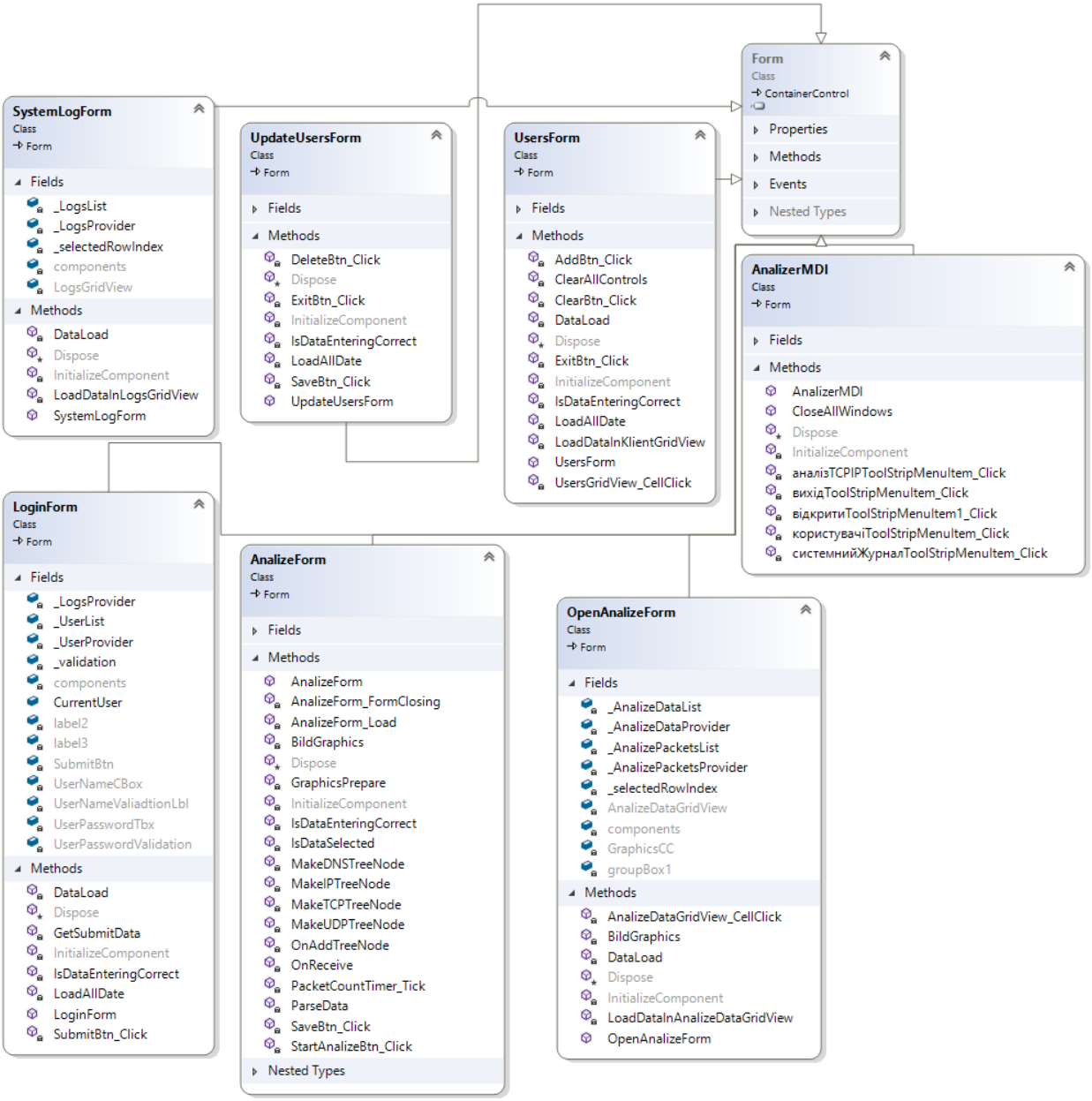


Рисунок 2.4 – Діаграма інтерфейсу програми «TCP/IP аналізер»

Програма складається з семи класів рівня UI та є похідними від класу Form, тобто мають графічний інтерфейс.

Клас «AnalizerMDI» становить головне вікно програми. Найважливішим елементом управління в головному вікні є розташування головного меню, за допомогою якого можна відкривати інші форми та вийти з програми.

Клас «UsersForm» призначений для опрацювання даних користувачів системи.

Клас «UpdateUsersForm» призначений для редагування інформації про вибраного користувача.

Клас «LoginForm» призначений для опрацювання введених даних

облікового запису користувача для входу в систему.

Клас «SystemLogForm» призначений для виведення подій системи.

Клас «AnalizeForm» призначений для аналізу пакетів TCP протоколу.

Клас «OpenAnalizeForm» призначений для виведення збережених даних аналізованих пакетів.

2.5.2 Розробка шару доступу до даних

Шар доступу до даних (Data Access Layer – DAL) у програмному забезпеченні – це шар комп'ютерної програми, який надає спрощений доступ до даних, що зберігаються у постійному сховищі якогось типу, такому як реляційна база даних. Цей акронім в основному використовується в оточенні Microsoft.NET.

DAL може повертати посилання на об'єкт (у термінах об'єктно-орієнтованого програмування) з його атрибутами замість рядків полів із таблиці бази даних. Це дозволяє створювати клієнтські (або модулі користувача) модулі з більш високим рівнем абстракції. Така модель може бути реалізована шляхом створення класу з методами доступу до даних, які безпосередньо посилаються на відповідний набір процедур бази даних. Інша реалізація може потенційно отримувати або записувати записи або з файлової системи. DAL приховує складність сховища даних, що лежить в основі даних.

Замість використання таких команд як «створити», «видалити» або «оновити» в певній таблиці в базі, клас і кілька процедур, що зберігаються, можуть бути створені в базі. Ці процедури можуть викликатися методом усередині класу, який поверне об'єкт, що містить запитані значення. Або команди створення, видалення та оновлення можуть бути виконані всередині простих функцій, що зберігаються у шарі доступу до даних.

Також методи бізнес-логіки із програми можуть бути співвіднесені до шару доступу до даних.

Для роботи з базою даних було реалізовано 7 класів. Назва кожного класу починається з відповідній їй назві таблиці в базі даних та закінчується приставкою «Provider».

На рис. 2.5 приведено діаграму класів з методами для роботи з базою даних.

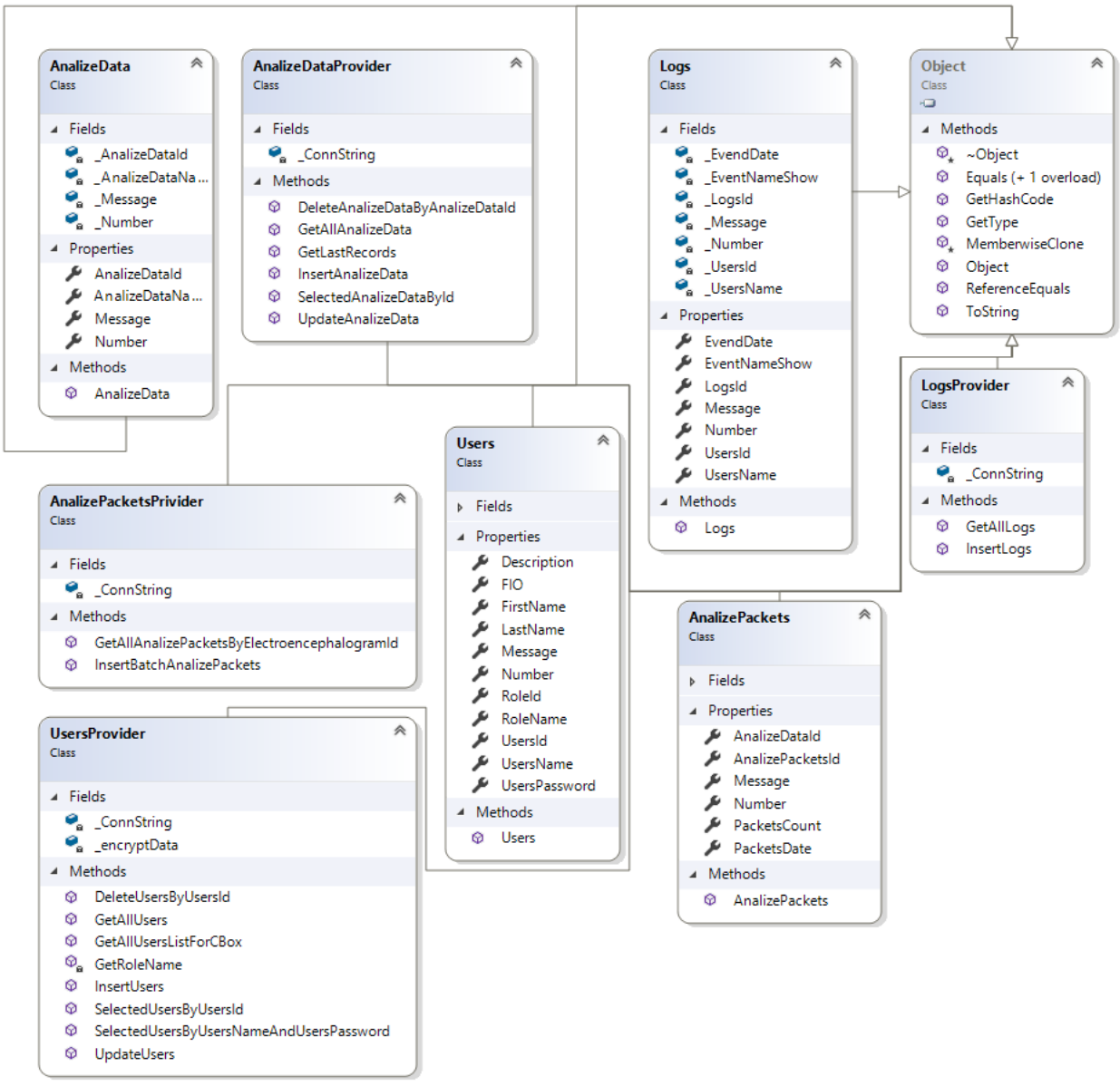


Рисунок 2.5 – Діаграма класів з методами для роботи з базою даних

Класи, що призначені для опрацювання даних шару DAL:

«AnalizePacketsPrivider» – призначений для опрацювання даних збережених даних аналізу пакетів.

«AnalizeDataProvider» – призначений для опрацювання списків збереженої інформації.

«LogsProvider» – використовується для опрацювання даних подій системи.

«UsersProvider» – призначений для опрацювання інформації користувачів системи.

2.6 Розробка алгоритму роботи програми аналізу пакетів TCP протоколу

Перед написанням коду програми, був розроблений алгоритм аналізу пакетів TCP протоколу для вибраної IP адреси вузла. Даний алгоритм зображено на рис. 2.6.

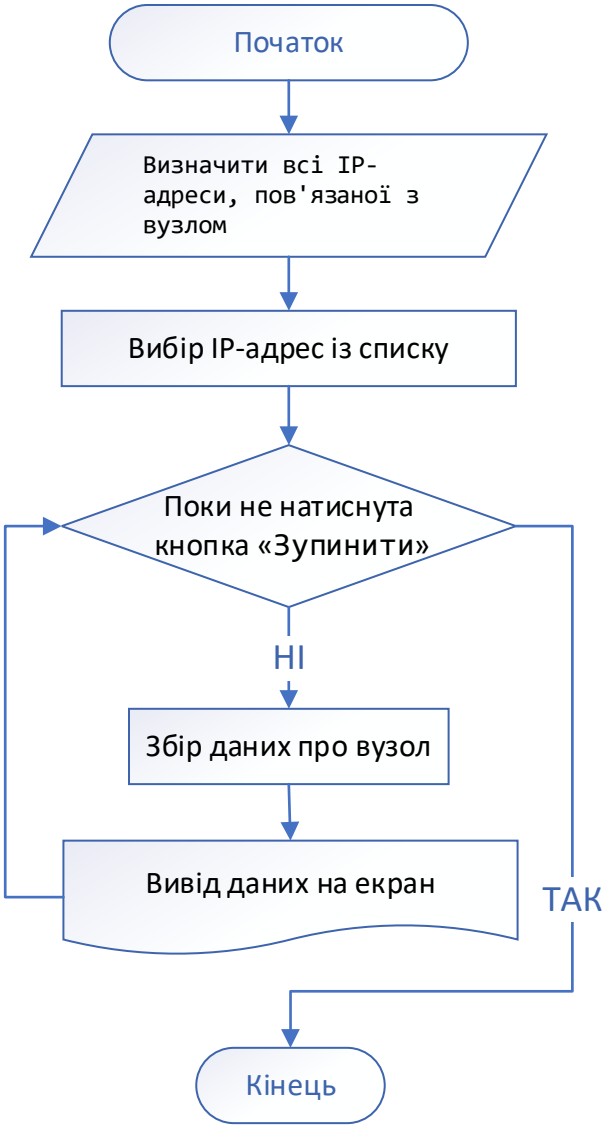


Рисунок 2.6 – Алгоритм роботи аналізу пакетів TCP протоколу

Після вибору меню «Аналіз пакетів» завантажується форма та

викликається подія «AnalizeForm_Load», в якій за допомогою методу «GetHostEntry» визначаються всі IP адреси вузла. Всі адреси записуються у випадаючий список, в якому користувач може вибрати будь-яку із них. Вибравши адресу необхідно натиснути кнопку «Аналізувати», після чого буде збиратись вся інформація для цієї IP адреси та виводитись на екран у деревовидній структурі. Окрім цього буде виведено графік перехоплених пакетів даного вузла у реальному часі. Для того, щоб зупинити процес аналізу необхідно натиснути кнопку «Зупинити».

2.7 Висновок

Побудова сучасних інформаційних система займає дуже багато часу. Вона починається з аналізу предметної області, класного планування системи, описання вимог, моделювання її поведінки за допомогою uml діаграм. Визначається шаблони, які будуть використовуватись при розробці.

Після планування починається стадія розробки. Розробка починається зі створення користувацького інтерфейсу і закінчується базою даних.

При розробці слід враховувати, що вимоги змінюються швидко і потрібно будувати систему так, щоб вона була гнучкою.

Розробка системи виконується по окремим компонентам. Кожний створений компонент потрібно класно тестувати, щоб мінімізувати помилки на наступних етапах розробки.

Якщо дотримуватися всіх вищеперерахованих вимог, то можна побудувати гнучку і стійку інформаційну систему.

3 РЕАЛІЗАЦІЯ У ВИХІДНИХ КОДАХ ПРОГРАМИ АНАЛІЗУ ПАКЕТІВ TCP ПРОТОКОЛУ

3.1 Особливості вихідних кодів проектованої програми аналізу пакетів

На початку роботи засобами MS Access було створено базу даних, ERD діаграма якої зображена на рис. 3.1.

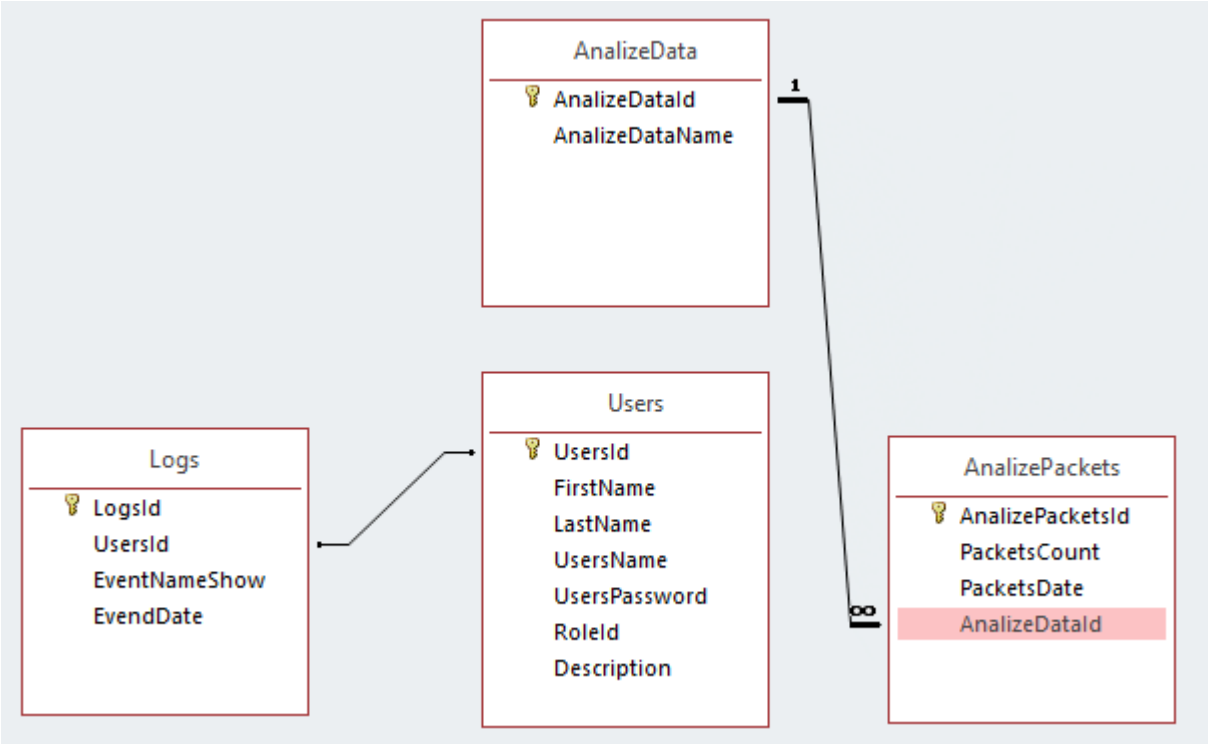


Рисунок 3.1 – ERD діаграма бази даних проекту

Для під’єднання БД до середовища Microsoft Visual Studio 2019 у файлі проекту "App.config" створюємо змінну "CONNECT" та задаємо значення параметрів змінної «CONNECT» (Лістинг 2.1).

Лістинг 2.1 Параметри змінної "CONNECT"

<add key="CONNECT" value="Provider=Microsoft.Jet.OLEDB.4.0;Data Source=|DataDirectory|\DataBase.mdb;"/>

Для роботи з створеною базою даних використовуємо простір імен System.Data.OleDb.

Далі було створено головне вікно проекту та додано до нього елемент menuStrip.

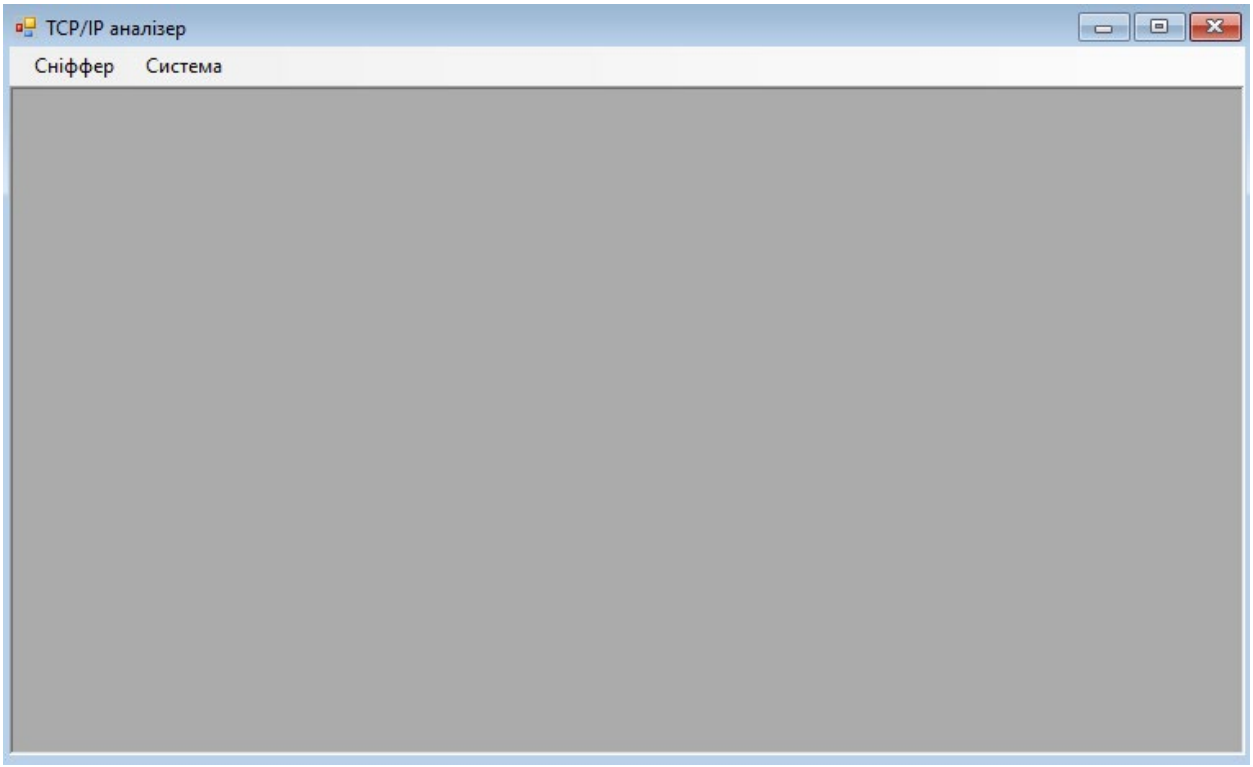


Рисунок 3.2 - Створене головне вікно та меню системи

На кожен із пунктів меню додано певний код, який відповідає за створення екземплярів форм та закриття попередньо відкритого вікна. Аналогічний код застосовуємо для всіх пунктів меню.

```
private void аналізTCPIPToolStripMenuItem_Click(object sender, EventArgs e) {
  CloseAllWindows();
  AnalizeForm analizeForm = new AnalizeForm();
  analizeForm.MdiParent = this;
  analizeForm.WindowState = FormWindowState.Maximized;
  analizeForm.Show();
}
```

Рисунок 3.3 – Програмування пунктів меню

Для роботи з базою даних було створено відповідні класи, всі вони розміщені у папці із назвою «Providers» в проекті рішення.

Для з’єднання з базою даних використовується метод «Open» екземпляру класу «OleDbConnection». Для закриття з’єднання з базою даних використовується метод «Close» того ж екземпляру класу.

Для опрацювання даних аналізу пакетів TCP протоколу був створений клас «AnalizePacketsPrivider», який містить в собі 2 публічних методи для додавання інформації про перехоплені пакети та вибірки інформації

збережених пакетів із бази даних.

Код методу для додавання інформації про перехоплені пакети в базу даних представлений на рис. 3.4.

```
public void InsertBatchAnalizePackets(List<AnalizePackets> AnalizePackets) {
  string SqlString = "INSERT INTO AnalizePackets (PacketsCount, PacketsDate, AnalizeDataId) Values(?, ?, ?)";
  using (OleDbConnection conn = new OleDbConnection(_ConnString)) {
    using (OleDbCommand cmd = new OleDbCommand(SqlString, conn)) {
      cmd.CommandType = CommandType.Text;
      conn.Open();
      for (int i = 0; i < AnalizePackets.Count; i++) {
        cmd.Parameters.AddWithValue("PacketsCount", AnalizePackets[i].PacketsCount);
        cmd.Parameters.AddWithValue("PacketsDate", AnalizePackets[i].PacketsDate.ToString());
        cmd.Parameters.AddWithValue("AnalizeDataId", AnalizePackets[i].AnalizeDataId);
        cmd.ExecuteNonQuery();
        while (cmd.Parameters.Count > 0) {
          cmd.Parameters.RemoveAt(0);
        }
      }
      conn.Close();
    }
  }
}
```

Рисунок 3.4 – Метод «InsertBatchAnalizePackets» для додавання інформації про перехоплені пакети

Для методу, вибірки інформації про перехоплені пакети був написаний код, що показаний на рис. 3.5.

```
public List<AnalizePackets> GetAllAnalizePacketsByElectroencephalogramId(int AnalizeDataId) {
  int i = 0;
  string SqlString = "SELECT * FROM AnalizePackets WHERE AnalizeDataId=" + AnalizeDataId;

  List<AnalizePackets> AnalizePackets = new List<AnalizePackets>();
  using (OleDbConnection conn = new OleDbConnection(_ConnString)) {
    using (OleDbCommand cmd = new OleDbCommand(SqlString, conn)) {
      conn.Open();
      using (OleDbDataReader reader = cmd.ExecuteReader()) {
        while (reader.Read()) {
          AnalizePackets oneAnalizePackets = new AnalizePackets();
          oneAnalizePackets.Number = ++i;
          oneAnalizePackets.AnalizePacketsId = Convert.ToInt32(reader["AnalizePacketsId"]);
          oneAnalizePackets.PacketsCount = Convert.ToInt32(reader["PacketsCount"]);
          oneAnalizePackets.PacketsDate = Convert.ToDateTime(reader["PacketsDate"]);
          oneAnalizePackets.AnalizeDataId = Convert.ToInt32(reader["AnalizeDataId"]);
          AnalizePackets.Add(oneAnalizePackets);
        }
      }
      conn.Close();
    }
  }
  return AnalizePackets;
}
```

Рисунок 3.5 – Метод «GetAllAnalizePacketsByAnalizeDataId» для вибірки інформації про перехоплені пакети

Наступним кроком було створення форми для пакетів TCP протоколу (рис. 3.6).

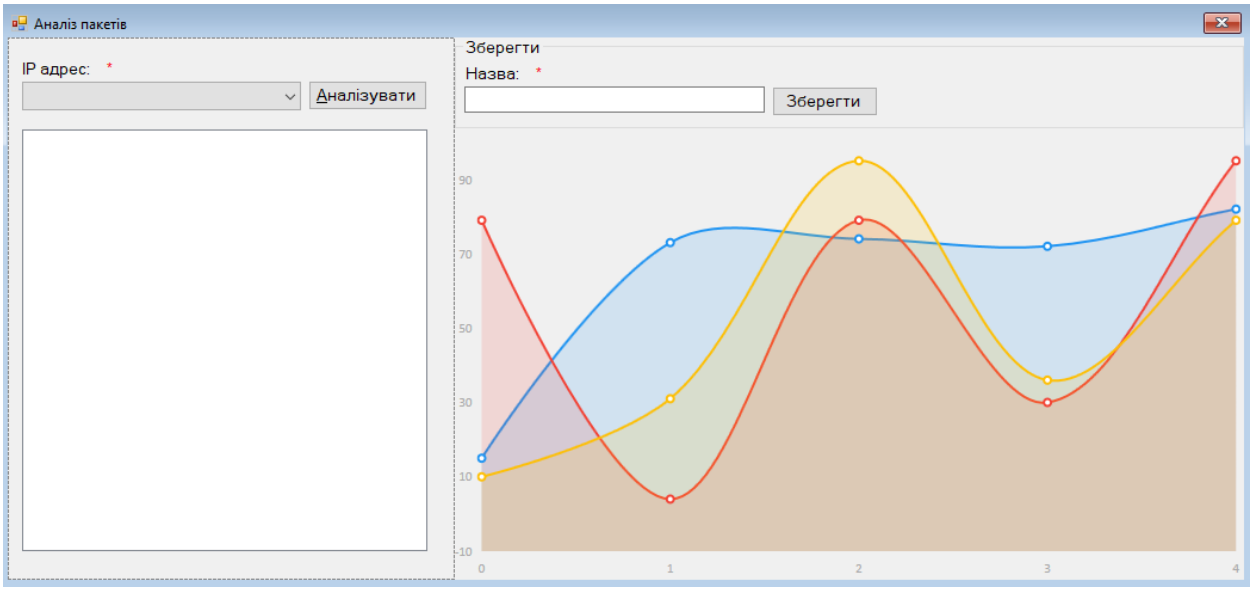


Рисунок 3.6 – Форма для опрацювання інформації про перехоплені пакети

При завантаження форми у події завантаження форми викликається метод «AnalizeForm_Load», який збирає дані про про всі IP адреси вузла (рис. 3.7).

```
private void AnalizeForm_Load(object sender, EventArgs e) {
    string strIP = null;
    IPHostEntry HosyEntry = Dns.GetHostEntry((Dns.GetHostName()));
    if (HosyEntry.AddressList.Length > 0) {
        foreach (IPAddress ip in HosyEntry.AddressList) {
            strIP = ip.ToString();
            IPHostInterfacesCBox.Items.Add(strIP);
        }
    }
}
```

Рисунок 3.7 – Метод для збирання інформації про всі IP адреси вузла

Для кнопки «Аналізувати» був розроблений обробник подій «StartAnalizeBtn_Click», що викликається при натисканні на неї. Код методу обробника події «StartAnalizeBtn_Click» показаний на рис. 3.8.

```
private void StartAnalizeBtn_Click(object sender, EventArgs e) {
  if (IsDataSelected()) {
    if (!PacketCountTimer.Enabled) {
      _AnalizePacketsList.Clear();
      _PocketsValues.Clear();
      _DateTimes.Clear();
      PacketCountTimer.Start();
      StartAnalizeBtn.Text = "&Зупинити";
      _BContinueCapturing = true;

      try {
        //Почати захоплення пакетів...
        //Для нюхання сокет для захоплення пакетів має бути необробленим сокетом із сімейством адрес типу internetwork, а протоколом – IP
        _MainSocket = new Socket(AddressFamily.InterNetwork,
            SocketType.Raw, ProtocolType.IP);
        //Прив'язуємо сокет до вибраної IP-адреси
        _MainSocket.Bind(new IPEndPoint(IPAddress.Parse(IPHostInterfacesCBox.Text), 0));
        //Встановлюємо параметри сокету
        _MainSocket.SetSocketOption(SocketOptionLevel.IP,            //Застосовується лише до IP-пакетів
                                    SocketOptionName.HeaderIncluded, //Встановлюємо включення заголовка в true
                                    true);

        byte[] byTrue = new byte[4] { 1, 0, 0, 0 };
        byte[] byOut = new byte[4] { 1, 0, 0, 0 }; //Захоплюємо вихідні пакети

        //Socket.IOControl аналогічний методу  WSAIoctl
        _MainSocket.IOControl(IOControlCode.ReceiveAll, byTrue, byOut); //Еквівалент константи SIO_RCVALL Winsock 2

        //Починаємо отримувати пакети асинхронно
        _MainSocket.BeginReceive(_ByteData, 0, _ByteData.Length, SocketFlags.None,
            new AsyncCallback(OnReceive), null);

      } catch (Exception ex) {
        MessageBox.Show(ex.Message, "Аналіз пакетів", MessageBoxButtons.OK, MessageBoxIcon.Error);
      }
    } else {
      StartAnalizeBtn.Text = "&Аналізувати";
      PacketCountTimer.Stop();
      _BContinueCapturing = false;
      //Щоб припинити захоплення пакетів, закриваємо сокет
      _MainSocket.Close();
    }
  }
}
```

Рисунок 3.8 – Код методу обробника події «StartAnalizeBtn_Click»

Для збереження даних інформації аналізу вузла був також написаний код, що показаний на рис. 3.9.

```
private void SaveBtn_Click(object sender, EventArgs e) {
  if (IsDataEnteringCorrect()) {
    StartAnalizeBtn.Text = "&Аналізувати";
    PacketCountTimer.Stop();
    _AnalizeDataProvider.InsertAnalizeData(AnalizeDataNameTBox.Text);
    int lastAnalizeDataId = _AnalizeDataProvider.GetLastRecords();
    for (int i = 0; i < _AnalizePacketsList.Count; i++) {
      _AnalizePacketsList[i].AnalizeDataId = lastAnalizeDataId;
    }
    _LogsProvider.InsertLogs(LoginForm.CurrentUser.UsersId, "Було проведено аналіз пакетів та збережено дані під назвою '" +
      AnalizeDataNameTBox.Text + "'", DateTime.Now);
    _AnalizePacketsPrivider.InsertBatchAnalizePackets(_AnalizePacketsList);
    _PocketsValues.Clear();
    _DateTimes.Clear();
    treeView.Nodes.Clear();
    AnalizeDataNameTBox.Text = String.Empty;
  }
}
```

Рисунок 3.9 – Код методу обробника події «SaveBtn_Click»

Після закриття форми викликається подія, що закриває з'єднання із сокетом (рис. 3.10).

```
private void AnalizeForm_FormClosing(object sender, FormClosingEventArgs e) {
  if (_BContinueCapturing) {
    _MainSocket.Close();
  }
}
```

Рисунок 3.10 – Подія закриття з’єднання із сокетом

3.2 Тестування розробленої програми та оцінка ефективності отриманого рішення

Тестування — це процес аналізу та дослідження, який надає змогу виявити інформацію про якість продукту відносно умов, в яких він буде застосовуватись. Методика тестування також включає в себе процес пошуку дефектів, помилок, несправностей. Також це є випробуванням програмних складових з метою оцінити готовність програмного продукту до використання. Результат тестування оцінюється за наступними критеріями:

— відповідність вимогам, які надавалися розробниками та проектувальниками;

— відповідність вихідних даних;

— прийнятний час виконання функцій;

— практичність;

— відповідність вимогам замовника.

Кількість тестів навіть для простих програмних компонентів може бути ледь не нескінченним, тому тактика тестування має полягати в тому, що будуть проведені тільки необхідні тести з урахуванням доступного часу та ресурсів. Як результат, програмні засоби тестуються стандартним виконанням програми з метою виявлення багів, помилок або інших дефектів.

Існує багато видів тестування: одні зазвичай виконують самі розробники, а інші — спеціалізовані групи. В нашому ж випадку буде використовуватись тестування системи.

Тестування програмних засобів було проведено використовуючи методику «чорної скриньки».

Вона базується на використанні шаблонів тестування або ж тест-кейсів. Це означає, що буде створено декілька ситуацій у яких перевіряється працездатність додатку, коректності основних функцій.

3.1 – Тест-кейси

<table>
<tr><td rowspan="3">Код тест-кейса</td><td colspan="3">Опис тест-кейса</td></tr>
<tr><td colspan="2">Хід тестування</td><td>Очікуваний результат</td></tr>
<tr><td>Дата тестування</td><td>Результат</td><td>Примітка</td></tr>
<tr><td rowspan="3">001</td><td colspan="3">Перевірка автентифікації користувача програми</td></tr>
<tr><td colspan="2">1. Запустити додаток;
2. Провести автентифікацію користувача.</td><td>Вивід відповідного вікна із функціоналом для роботи</td></tr>
<tr><td>10.08.2022</td><td>Пройдено</td><td>–</td></tr>
<tr><td rowspan="3">002</td><td colspan="3">Перевірка правильності методу аналізу пакетів</td></tr>
<tr><td colspan="2">1. Запустити додаток;
2. Провести автентифікацію;
3. Вибрати відповідний пункт меню;
4. Вибрати IP адресу із списку;
5. Натиснути кнопку «Аналізувати»</td><td>Аналіз даних проведено успішно, дані виводяться у відповідності до перехоплених пакетів</td></tr>
<tr><td>10.08.2022</td><td>Пройдено</td><td>–</td></tr>
<tr><td rowspan="2">003</td><td colspan="3">Збереження інформації у базу даних</td></tr>
<tr><td colspan="2">1. Запустити додаток;
2. Провести автентифікацію;
3. Вибрати відповідний пункт меню;</td><td>Інформація про аналіз даних успішно збереглась у базі даних</td></tr>
</table>

	4. Вибрати IP адресу із списку; 5. Натиснути кнопку «Аналізувати»; 6. Натиснути кнопку «Зупинити»; 7. Натиснути кнопку «Зберегти».		
	10.08.2022	Пройдено	–
004	Зчитування інформації із бази даних		
	1. Запустити додаток; 2. Провести автентифікацію; 3. Вибрати відповідний пункт меню; 4. Вибрати із списку збережений аналіз. 1.		Успішне виведення результатів збереженої інформації
	10.08.2022	Пройдено	–

3.3 Інструкція користувача

3.3.1 Опис процедури розгортання програмного продукту, створеного на платформі .NET

Важливим фактором, який необхідно врахувати при розробці програмного забезпечення є потреба в ресурсах.

Мінімальні вимоги до апаратних засобів для запуску та функціонування розробленого програмного забезпечення:

– Процесор PentiumIV/Xeon2.4Ггц.
– Оперативна пам'ять: 1024 Мб вище.
– Вільний дисковий простір менше 120Мб.
– Миша.
– Клавіатура.

– Монітор.

Вимоги до програмних засобів:

– Операційна система сімейства Windows: починаючи з Windows 7 – до Windows 10.

– Наявність бібліотеки класів .NET framework 4.5 або вище.

Необхідні дії, щоб розгорнути програмний продукт на комп'ютері користувача у вигляді автономного застосунку:

1. Створити на цільовому диску каталог для застосунку, напр. «TCPIP Analizer»;
2. Скопіювати каталог «Debug» із проекту.
3. Для перевірки коректності запуску програми виконайте подвійне клацання лівою кнопкою миші на файлі «TCPIPAnalizer.exe» (операційна система Windows).

3.3.2 Використання програмного продукту

На початку необхідно запустити додаток «TCP/IP аналізер». Після запуску програми буде виведено вікно, де користувачу буде запропоновано ввести ім’я та пароль.

Після успішної автентифікації, користувач в залежності від його ролі відкриється відповідний фунціонал програми.

Якщо ж була вибрана роль «адміністратор», тоді програма такі надає такі можливості:

- додавання користувачів системи;
- проведення аналізу пакетів TCP протоколу;
- можливість збереження даних аналізу;
- перегляд логів системи.

Для того, щоб провести аналіз пакетів TCP протоколу, користувачу необхідно перейти по меню програми «Сніффер» –> «Аналіз TCP/IP» та

вибравши необхідний IP адрес необхідно натиснути кнопку «Аналізувати». Після чого, програма почне проводити перехоплення пакетів даних із вибраного IP адресу . На рисунку 3.21 показано перехоплення даних із IP адресу «192.168.1.104» .

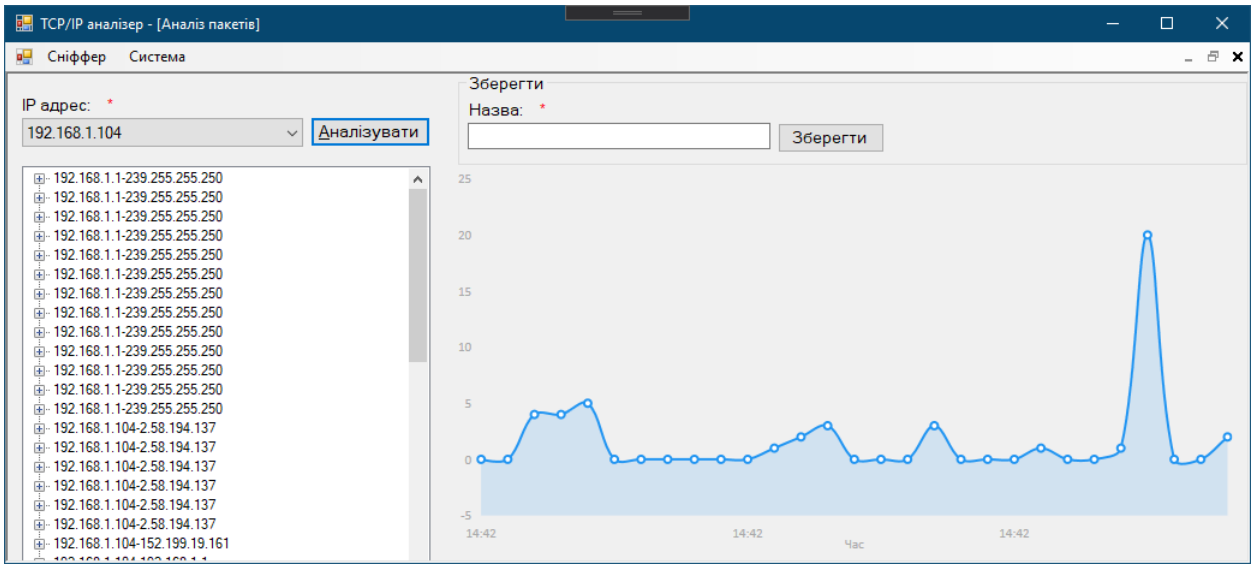


Рисунок 3.11 – Вікно для аналізу перехоплених пакетів даних

У лівій частині відображається інформація про всі відправлені та перехоплені пакети даних із IP адрес, що надсилаються на вибрану адресу, а в правій частині вікна відображається графік, що відображає кількість перехоплених пакетів, що дозволить виявити аномальну поведінку мережі.

Також дані можна зберегти у базу даних, для цього необхідно вказати назву та натиснути кнопку «Зберегти».

Для перегляду збереженої інформації необхідно перейти по меню програми «Сніффер» –> «Відкрити». Після цього можна вибрати будь-які збережені дані вибравши необхідну назву із списку (рис. 3.12).

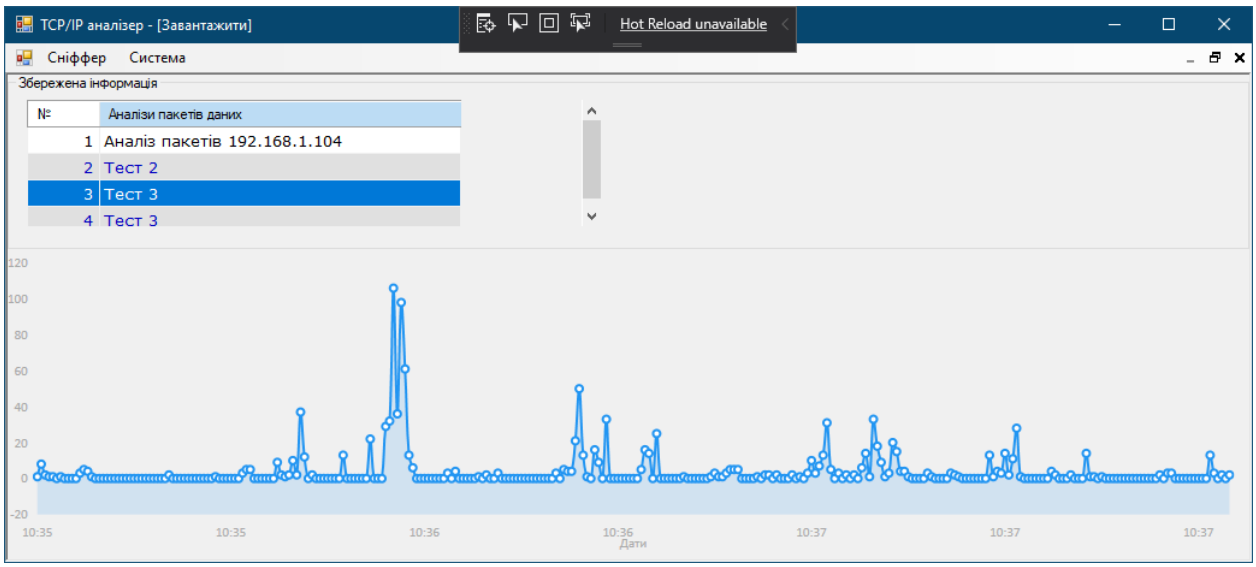


Рисунок 3.12 – Завантаження даних

Користувач з правами адміністратора може створювати користувачів системи і надавати їм роль в системі тестування. Для цього необхідно перейти по головному меню «Система» –> «Користувачі». Щоб додати нового користувача системи необхідно заповнити всі обов'язкові поля, а саме: прізвище, ім'я, логін та пароль (рис. 3.13). Дуже важлива, щоб поля пароля та підтвердження пароля співпадали, якщо ж цього не буде – програма не дозволить додати нового користувача в систему.

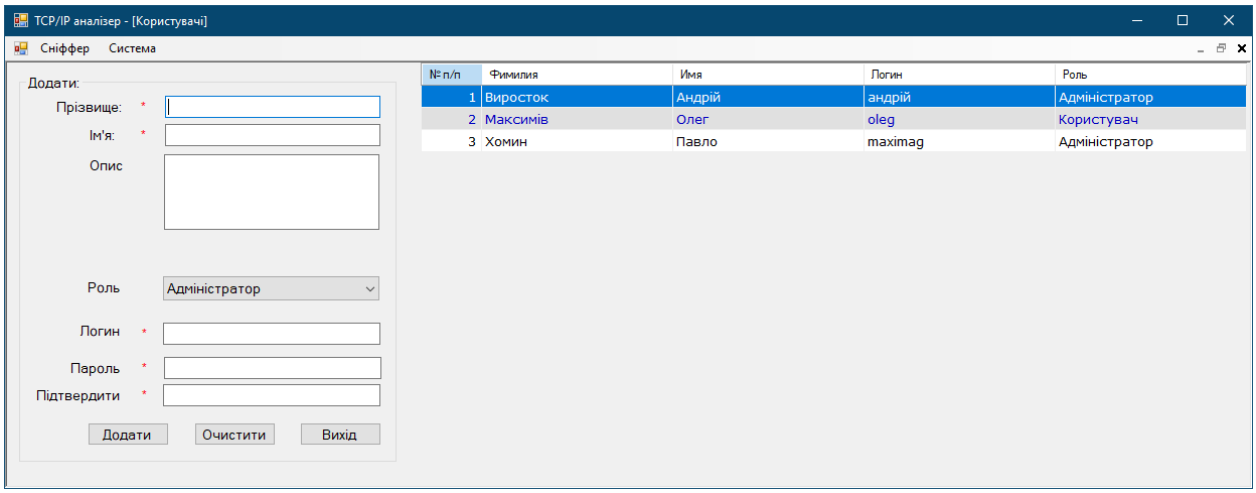


Рисунок 3.13 – Додавання нового користувача в систему

Також дані можна редагувати в разі необхідності. Для цього можна вибрати необхідний із списку обліковий запис на натиснути на нього. Після чого відкриється відповідне вікно для редагування даних (рис. 3.14).

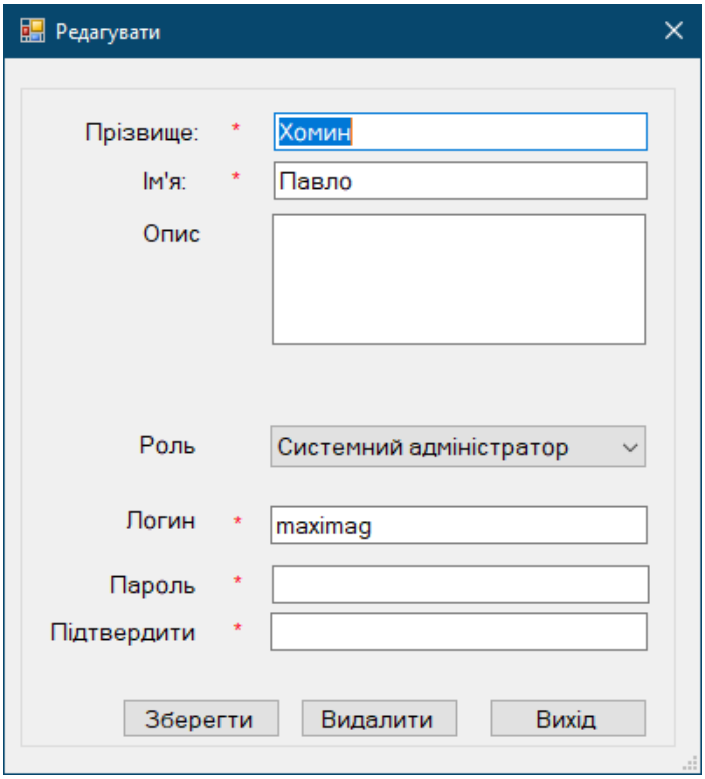


Рисунок 3.14 – Редагування даних користувача системи

Якщо ж поля «Пароль» та «Підтвердити» не співпадають, програма виведе повідомлення про це (рис. 3.15).

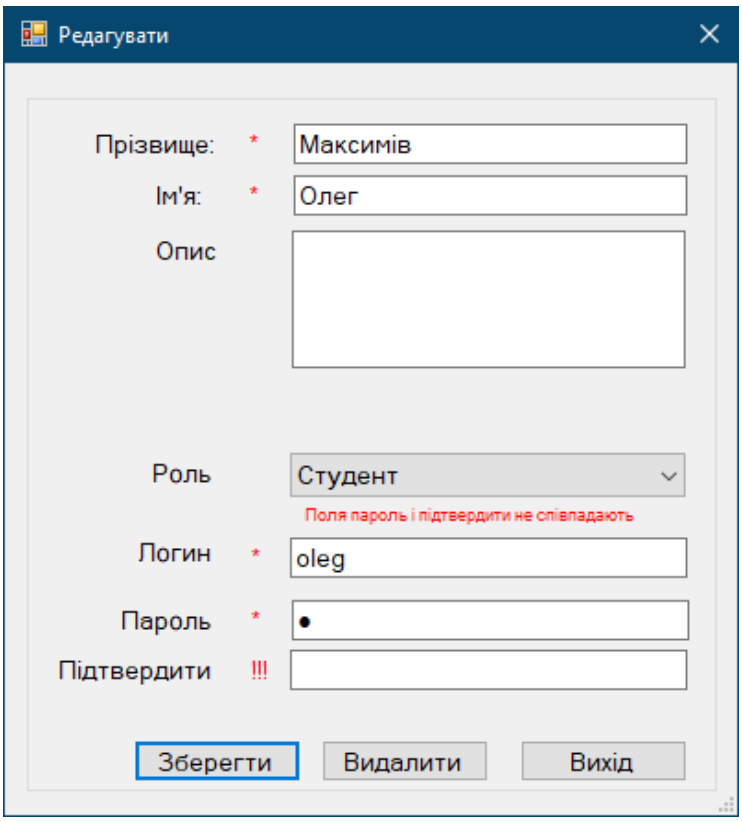
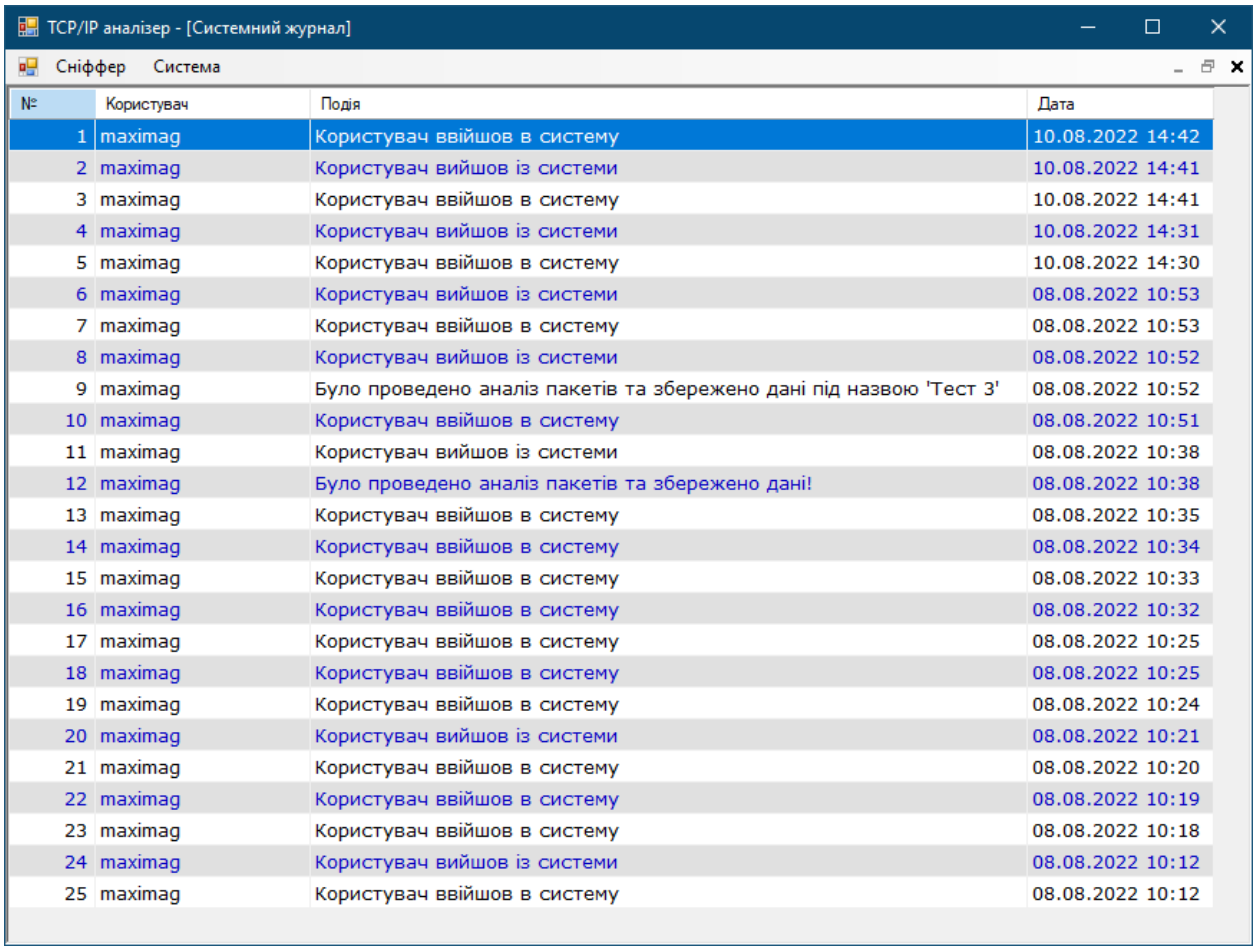


Рисунок 3.15 – Пароль» та «Підтвердити» не співпадають

Також в системі можна бачити активність користувачів, та те, що вони робили в системі. Це можливо зробити за допомогою облікового запису, який має права системного адміністратора. Для цього перейдемо по меню «Система» –> «Системний журнал». В цьому вікні виводяться всі події системи.



№	Користувач	Подія	Дата
1	maximag	Користувач ввійшов в систему	10.08.2022 14:42
2	maximag	Користувач вийшов із системи	10.08.2022 14:41
3	maximag	Користувач ввійшов в систему	10.08.2022 14:41
4	maximag	Користувач вийшов із системи	10.08.2022 14:31
5	maximag	Користувач ввійшов в систему	10.08.2022 14:30
6	maximag	Користувач вийшов із системи	08.08.2022 10:53
7	maximag	Користувач ввійшов в систему	08.08.2022 10:53
8	maximag	Користувач вийшов із системи	08.08.2022 10:52
9	maximag	Було проведено аналіз пакетів та збережено дані під назвою 'Тест 3'	08.08.2022 10:52
10	maximag	Користувач ввійшов в систему	08.08.2022 10:51
11	maximag	Користувач вийшов із системи	08.08.2022 10:38
12	maximag	Було проведено аналіз пакетів та збережено дані!	08.08.2022 10:38
13	maximag	Користувач ввійшов в систему	08.08.2022 10:35
14	maximag	Користувач ввійшов в систему	08.08.2022 10:34
15	maximag	Користувач ввійшов в систему	08.08.2022 10:33
16	maximag	Користувач ввійшов в систему	08.08.2022 10:32
17	maximag	Користувач ввійшов в систему	08.08.2022 10:25
18	maximag	Користувач ввійшов в систему	08.08.2022 10:25
19	maximag	Користувач ввійшов в систему	08.08.2022 10:24
20	maximag	Користувач вийшов із системи	08.08.2022 10:21
21	maximag	Користувач ввійшов в систему	08.08.2022 10:20
22	maximag	Користувач ввійшов в систему	08.08.2022 10:19
23	maximag	Користувач ввійшов в систему	08.08.2022 10:18
24	maximag	Користувач вийшов із системи	08.08.2022 10:12
25	maximag	Користувач ввійшов в систему	08.08.2022 10:12

Рисунок 3.16 – Вікно «Системний журнал»

Треба сказати, що дана система є не сильно функціональною, але вона є досить простою в користуванні. ЇЇ інтерфейс є інтуїтивно зрозумілим для користувача.

3.4 Висновок

В ході виконання роботи мовою C# в середовищі Visual Studio 2019 реалізовано систему, призначену для аналізу пакетів TCP протоколу.

Реалізовано такі функціональні вимоги:

- можливість додавати, редагувати та видаляти облікові записи користувачів;
- можливість аналізу пакетів TCP протоколу;
- можливість збереження інформації про аналіз пакетів;
- можливість завантаження збереженої інформації;
- фіксування активності користувачів системи. Ведуться записи подій системи.

ВИСНОВКИ

Вивчення сучасних методик аналізу мережного трафіку показало, що статистичне дослідження трафіку провайдера, що виходить з локальної мережі у зовнішню мережу, на даний момент активно не використовується в системах захисту. Основний напрямок подібних систем – моніторинг працездатності обладнання, навантаження на канали; а для локальних мереж – це захист від загроз, що приходять із зовнішньої мережі.

Розроблена архітектура показала себе гідно в рамках прототипу, але для подальшої розробки необхідно дослідити інші бази даних, щоб підібрати найоптимальнішу з них для роботи в заданих умовах, і продумати архітектуру даних, що зберігаються в ній.

Створені алгоритми статистичної обробки є першим наближенням серйозної аналітики. У майбутньому слід врахувати, що отримані дані мають великий обсяг, додати оптимізацію, а також проконсультуватися з передбачуваними користувачами системи, щоб реалізувати актуальні для потреб алгоритми дослідження трафіку.

У рамках дипломної роботи було розроблено прототип системи аналізу трафіку, який вирішує такі завдання:

– Опрацьовує дамп трафіку;
– Перетворює і зберігає мережеві пакети в базу даних;
– Відображає розроблену нами аналітику.

Передбачуваний ефект у разі застосування подібних систем у великого числа провайдерів виглядає перспективним. Варто враховувати, що призначення системи не захист трафіку, а аналіз великого потоку та виявлення можливих статистично некоректних випадків, на основі яких провайдер може ухвалити рішення про виконання якихось протиборчих дій.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Платформи глибокого аналізу трафіку та управління трафіком додатків [Електронний ресурс] Режим доступу: http://www.inlinetelecom.com/solutions/access_network/platform_depth_analysis_of_traff ic_and_traffic_control_applications/
2. David L. Cannon. CISA Certified Information Systems Auditor Study Guide, 2nd Edition, 2008, ISBN: 978-0-470-23152-4
3. What is Deep Packet Inspection [Електронний ресурс] Режим доступу: https://www.techtarget.com/searchnetworking/definition/deep-packet-inspection-DPI
4. Requirements for Deep Packet Inspection in Next Generation Networks [Електронний ресурс] Режим доступу: https://www.itu.int/rec/T-REC-Y.2770-201211-I/en
5. David L. Cannon. CISA Certified Information Systems Auditor Study Guide, 2nd Edition, 2008, ISBN: 978-0-470-23152-4
6. QUIC [Електронний ресурс] Режим доступу: https://tools.ietf.org/html/draft-tsvwg-quic-protocol-00
7. F. Risso, M. Baldi, O. Morandi, A. Baldini, P. Monclus, “Lightweight, payload-based traffic classification: An experimental evaluation” in Proc. IEEE ICC, 2018, pp. 5869– 5875
8. F. Risso, and L. Degioanni, “An Architecture for High Performance Network Analysis”, Proceedings of the 6th IEEE Symposium on Computers and Communications (ISCC 2001), Hammamet, Tunisia, July 2001
9. PF_RING ZC (Zero Copy) [Електронний ресурс] Режим доступу: https://www.ntop.org/products/packet-capture/pf_ring/pf_ring-zc-zero-copy/
10. Local network connectivity [Електронний ресурс] Режим доступу: https://www.researchgate.net/publication/352522421_Local_network_connectivity_optimization_an_evaluation_of_heuristics_applied_to_complex_spatial_networks_a_transportation_case_study_and_a_spatial_social_network

11. Alternative Search Internet Engines to Use in 2022 [Електронний ресурс] Режим доступу: https://kinsta.com/blog/alternative-search-engines/

12. IDS vs. IPS [Електронний ресурс] Режим доступу: https://www.varonis.com/blog/ids-vs-ips

13. Understanding TCP Sequence and Acknowledgment Numbers [Електронний ресурс] Режим доступу: https://packetlife.net/blog/2010/jun/7/understanding-tcp-sequence-acknowledgment-numbers/

14. How To Prevent The Top Cyber Attacks In 2022 [Електронний ресурс] Режим доступу: https://purplesec.us/prevent-cyber-attacks/

15. IDS/IPS - intrusion detection and prevention systems [Електронний ресурс] Режим доступу: https://www.stamus-networks.com/en-us/landing/a-practical-guide-for-migrating-from-legacy-intrusion-detection-system-ids-to-modern-alternative

16. IPS Solutions [Електронний ресурс] Режим доступу: https://www.catonetworks.com/intrusion-prevention-system/ips-solutions/

17. Cisco MARS monitoring, analysis and response system [Електронний ресурс] Режим доступу: https://www.cisco.com/c/en/us/obsolete/security/cisco-security-monitoring-analysis-and-response-system.html

18. .NET 4.7 [Електронний ресурс] Режим доступу: https://temofeev.ru/info/articles/predstavlyaem-net-5/

19. Джон Скит, C# для професіоналів: тонкощі програмування, 608 ст. ISBN 978-5-8459-1909-0, 978-1-617-29134-0; 2014, Вільямс.

20. Литвинов О.А., Карпенко Н.В. Тестування інформаційних систем: модульне, інтеграційне, системне [Текст] – Д.: Ліра, 2016. – 283 с.

21. Литвинов О.А., Герасимов В.В., Карпенко Н.В. Об’єктно-орієнтована розробка інформаційних систем [Текст] – Д.: Ліра, 2018. – 448 с.

22. Програма для роботи з базами даних Microsoft Access: [Електронний ресурс] // Режим доступу: https://www.microsoft.com/uk-ua/microsoft-365/access.

23. Сайт для побудови ER-діаграм : [Електронний ресурс] // Режим доступу: https://app.diagrams.net/.

24. Архітектурні шаблони: [Електронний ресурс] // Режим доступу: https://devzone.org.ua/post/nayvazhlivishi-arkhitekturni-shabloni-yaki-neobkhidno-znati

25. SOLID: the first 5 Principles of OOD https://www.digitalocean.com/community/conceptual_articles/s-o-l-i-d-the-first-five-principles-of-object-oriented-design

Додатки

Додаток А

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЗАТВЕРДЖУЮ
Проректор Українського державного університету науки і технологій
Анатолій РАДКЕВИЧ
18.02.22

РОЗРОБКА ДЕМОНСТРАЦІЙНОЇ ПРОГРАМИ АНАЛІЗУ ПАКЕТІВ TCP ПРОТОКОЛУ

Технічне завдання

ЛИСТ ЗАТВЕРДЖЕННЯ

1357

Представники підприємства-розробника
Завідувач кафедри КІТ
Вадим ГОРЯЧКІН
18.02.22

Керівник розробки
Вадим АНДРЮЩЕНКО
18.02.22

Виконавець
Ілля ХОХЛОВ
18.02.22

Норм-контролер
Світлана ВОЛКОВА
18.02.22

2022

ЗАТВЕРДЖЕНО
1357

РОЗРОБКА ДЕМОНСТРАЦІЙНОЇ ПРОГРАМИ АНАЛІЗУ ПАКЕТІВ TCP ПРОТОКОЛУ

Технічне завдання

1357

Листів 16

2023

ЗМІСТ

1 Вступ

Програма для аналізу пакетів TCP протоколу призначена для ознайомлення користувача із особливостями пересилання трафіку із гарантованою доставкою повідомлень. Дозволяє у наочному режимі спостерігати за суттю процесів пересилання TCP-пакетів .

Особливістю TCP протоколу є гарантія того, що відправлені пакети будуть доставлені отримувачу [1], для чого застосовуються спеціальні процедури [2], які можна унаочнити за допомогою проектованої програми.

Користувач з використанням такої демонстраційної програми може не тільки ознайомитися із принципами роботи протоколів стеку TCP-IP, а й проводити елементарний аналіз особливостей функціонування тієї або іншої комп’ютерної мережі.

Програмний продукт має бути безкоштовним і вільним для поширення, що надає йому конкурентної переваги та робить більш привабливим на фоні інших аналогів. В той же час необхідно усвідомлювати, що розробка в першу чергу призначена для задач навчання та візуалізації навчального матеріалу, а не для цілей професійного захисту інформації або, тим більше, втручання в роботу комп’ютерних мереж [3].

2 ПІДСТАВИ ДЛЯ РОЗРОБКИ

Підставою для розробки є наказ від 08.12.21 №77ст ректора Українського державного університету науки і технологій “Про призначення наукових керівників та затвердження тем бакалаврських робіт” за спеціальністю 121 “Інженерія програмного забезпечення» факультету “Комп’ютерних технологій і систем” по кафедрі “Комп’ютерні інформаційні технології”.

Тема дипломної роботи - “РОЗРОБКА ДЕМОНСТРАЦІЙНОЇ ПРОГРАМИ АНАЛІЗУ ПАКЕТІВ TCP ПРОТОКОЛУ”. Керівник - доцент Андрющенко В.А.

3 ПРИЗНАЧЕННЯ РОЗРОБКИ

Функціональне призначення – програмний продукт має відображувати службову інформацію про пакети протоколу TCP [4], що їх відправляє адресатові та отримує у відповідь мережний інтерфейс персонального комп’ютера.

Експлуатаційне призначення – за допомогою програмного продукту виконується підготовка з дисциплін, що включають вивчення теми “Комп’ютерні мережі. Стек протоколів TCP-IP” для студентів комп’ютерних спеціальностей. Це дає змогу більше детально розглянути тему встановлення зв’язку між відправником та отримувачем при використанні транспортного протоколу гарантованої доставки повідомлень та і взагалі проаналізувати процес спілкування комп’ютерів у мережах TCP-IP [5].

4 ВИМОГИ ДО ПРОГРАМНОГО ПРОДУКТУ

4.1 Вимоги до функціональних характеристик

Програмний продукт повинний забезпечувати можливість перегляду пакетів, які обробляє мережний інтерфейс комп’ютера.

До пакетів, що обробляються мережним інтерфейсом належать [6]:

- пакети, що відправляються комп’ютером у мережу через його мережний інтерфейс;
- пакети, що отримуються комп’ютером із мережі через його мережний інтерфейс;
- пакети, що надсилаються комп’ютером самому собі через інтерфейс оберненої петлі TCP-IP (адреса 127.0.0.1, зареєстрована як localhost у локальному DNS правилі, системний файл hosts).

Програмний продукт повинен відображувати перелік усіх мережних інтерфейсів, наявних на тому комп’ютері, де він завантажується (в тому числі, обернену петлю loopback TCP-IP).

Повинна бути присутньою можливість вибору одного з переліку усіх наявних мережних інтерфейсів та відслідковування пакетів, що ним обробляються. До операцій обробки пакетів належать:

- операція відправлення TCP пакетів;
- операція отримання TCP пакетів.

Програмний продукт повинен відображувати детальну інформацію про кожен оброблений пакет [7], а саме:

- зафіксований момент часу, коли пакет було оброблено (отримано або відправлено);
- мережна адреса вибраного інтерфейсу;
- мережний порт, через який передаються пакети;

– прапорці пакету (зокрема, SYN, PSH, URG, FIN, RST);

– кількість переданих байтів (показник SEQ);

– кількість отриманих байтів (показник ACK);

– довжина даних у пакетів (показник LEN).

Програмний продукт має відображувати характеристики пакетів в залежності від даних, що у даний момент часу передає мережна плата, тобто програма здійснює моніторинг, але не управління потоком TCP пакетів.

Вимоги до вхідних даних від користувача:

- IP-адреса мережного інтерфейсу, який піддається моніторингу, і з якого слід знімати TCP пакети, що через нього передаються.

Вимоги до вихідних даних:

– відображення потоку вхідних пакетів, де для кожного з них відображується детальна інформація, наведена вище;

– відображення потоку вихідних пакетів, і для кожного відображується детальна інформація.

4.2 Вимоги до надійності

Вимоги до надійності наступні:

– при активації режиму спостереження слід увімкнути відповідний індикатор який має сигналізувати про активний режим роботи програми;

– наявність архівної копії початкового тексту програми на зовнішньому носії;

– наявність резервної копії компільованої версії програми на зовнішньому носії.

4.3 Вимоги експлуатації

Програмний продукт повинен використовуватись у приміщеннях які

відповідають умовам роботи ЕОМ, а саме мають такі кліматичні, санітарні та гігієнічні умови, які відповідають ДНАОП 0.00-1.13-99 (див. табл. 1).

Таблиця 1. Кліматичні умови

Пора року	Категорія робіт згідно з ГОСТ 12.01-005-88	Температура повітря, град.С	Відносна вологість повітря, %	Швидкість руху повітря, м/с
		Оптимальна	Оптимальна	Оптимальна
Холодна	легка-1-а	22-24	40-60	0,1
	легка-1-б	21-23	40-60	0,1
Тепла	легка-1-а	23-25	40-60	0,1
	легка-1-б	22-24	40-60	0,2

Працювати з програмою може людина, що має навички роботи з персональним комп’ютером під управлінням ОС Windows та ознайомлена з керівництвом користувача програмного продукту.

4.4 Вимоги до складу та параметрів технічних засобів

Продукт, що розробляється повинен використовуватись на персональних комп’ютерах, що мають наступні мінімальні системні характеристики:

- діагональ екрану – 14’;
- роздільна здатність дисплею – HD (1024x768);
- оперативна пам’ять – 2 ГБ;
- дискова пам’ять – від 50 МБ;
- операційна система – WINDOWS;
- частота процесора – 1 ГГц;
- комунікації – мережна плата з підключенням до мережі Інтернет.

4.5 Вимоги до інформаційної та програмної сумісності

Програмний продукт розробляється для всіх видів операційних систем сімейства “Windows” починаючи від версії XP SP3 та більш сучасних.

4.6 Вимоги до маркування і упаковки

Упаковка програмного продукту, включаючи документацію повинна бути захищена від пошкоджень різного роду (механічних, кліматичних).

На упаковці повинно бути вказана назва продукту, номер версії, мінімальні системні вимоги.

На зворотній стороні упаковки вказується розробник та його юридична адреса.

4.7 Вимоги до транспортування та зберігання

Транспортування повинне забезпечувати збереження програмного продукту його цілісність і запобігання несанкціонованого доступу до нього. Програмний продукт може передаватися на цільовий комп’ютер за допомогою фізичного носія (зокрема на лазерному диску типу CD чи DVD – за умови, якщо цільовий комп’ютер обладнаний відповідним приводом, або на карті флеш-пам’яті, що підключається до USB порту чи спеціального зчитувача ПК). Допустимим каналом потрапляння програми на цільовий комп’ютер є передача її через комунікаційні канали мережі Інтернет.

5 ВИМОГИ ДО ПРОГРАМНОЇ ДОКУМЕНТАЦІЇ

До складу документації, що додається до програмного забезпечення, мають входити:

- лістинг початкового тексту програми;
- опис програми (опис алгоритму, застосовані технології та засоби розробки);
- керівництво користувача.

Вся документація програмного додатку повинна задовольняти вимоги до програмної документації.

.

6 СТАДІЇ ТА ЕТАПИ РОЗРОБКИ

Таблиці 2. – Стадії та етапи розробки

Стадія	Зміст	Строки виконання
Технічне завдання	Постановка задачі, збір інформації, виріб та обґрунтування критеріїв розробки. Попередній вибір методів рішення задач. Визначення вимог до технічних засобів. Узгодження і затвердження технічного завдання.	31.01.22 -18.02.22
Робочий проект	Програмування та відлагодження програми.	19.02.22 - 20.05.22
	Тестування програми	20.05.22 - 27.05.22
	Розробка, узгодження і затвердження програмної документації.	27.05.22 - 12.06.22

7 ПОРЯДОК І КОНТРОЛЬ ПРИЙМАННЯ

Контроль за виконанням роботи здійснює науковий керівник розробки доц. Андрющенко В. О.

Прийом здійснюється уповноваженою комісією.

8 БІБЛІОГРАФІЧНИЙ СПИСОК

1. Буров Є.В. Комп’ютерні мережі: підручник. – Львів: «Магнолія 2006», 2010. – 262 с.

2. Вишневский В. М. Теоретические основы проектирования компьютерных сетей. - М.: Техносфера, 2003.

3. Голубев В.О., Гавловський ВД., Цимбалюк В.С. Інформаційна безпека: проблеми боротьби зі злочинами у сфері використання комп'ютерних технологій / За заг. ред. Р.А. Калюжного, М.Я. Швеця. — Запоріжжя: Просвіта, 2001. — 252 с.

4. Кулаков Ю.О. Комп’ютерні мережі: навч. посіб./ Ю.О. Кулаков, І.А. Жуков. – К.: вид-во Нац.авіц.ун-ту «НАУ-друк», 2009. – 392 с.

5. Лозінова Г. М. Комп'ютерні мережі: Навч. посібник. - К.: Кондор, 2003.

6. Олифер В.Г., Олифер Н.А.. Компьютерные сети. Принципы, технологии, протоколы. - СПб, “Питер”, 2001.

7. Поляк-Брагинский А. В. Администрирование сети на примерах. - СПб: БХВ-Петербург, 2005.

Лістинги програми

Лістинг 1. Код класу «AnalizeForm»

```
using LiveCharts;
using LiveCharts.Wpf;
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Net;
using System.Net.Sockets;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using TCPIPAnalizer.AppCode;
using TCPIPAnalizer.Providers;
using TCPIPAnalizer.Forms.Systems;

namespace TCPIPAnalizer.Forms.Sniffer {
  public partial class AnalizeForm : Form {
    private Socket _MainSocket;                    //Сокет, який перехоплює всі вхідні пакети
    private byte[] _ByteData = new byte[4096];
    private bool _BContinueCapturing = false; //Прапорець, який перевіряє, чи потрібно перехоплювати пакети чи ні

    private delegate void AddTreeNode(TreeNode node);
    private int _Count = 0;

    private AnalizeDataProvider _AnalizeDataProvider = new AnalizeDataProvider();
    private AnalizePacketsPrivider _AnalizePacketsPrivider = new AnalizePacketsPrivider();
    private List<AnalizePackets> _AnalizePacketsList = new List<AnalizePackets>();
    private ChartValues<double> _PocketsValues = new ChartValues<double>();
    private ValidationMy _Validation = new ValidationMy();
    private List<string> _DateTimes = new List<string>();
    private LineSeries _PocketsLine = new LineSeries();
    private SeriesCollection _Series = new SeriesCollection();
    private LogsProvider _LogsProvider = new LogsProvider();

    public AnalizeForm() {
      InitializeComponent();
      GraphicsPrepare();
    }
    private void StartAnalizeBtn_Click(object sender, EventArgs e) {
      if (IsDataSelected()) {
        if (!PacketCountTimer.Enabled) {
          _AnalizePacketsList.Clear();
          _PocketsValues.Clear();
          _DateTimes.Clear();
          PacketCountTimer.Start();
          StartAnalizeBtn.Text = "&Зупинити";
          _BContinueCapturing = true;
          try {
            //Почати захоплення пакетів...
            //Для нюхання сокет для захоплення пакетів має бути необробленим сокетом із сімейством адрес типу internetwork, а протоколом — IP
            _MainSocket = new Socket(AddressFamily.InterNetwork,
                SocketType.Raw, ProtocolType.IP);
            //Прив’язуємо сокет до вибраної IP-адреси
            _MainSocket.Bind(new IPEndPoint(IPAddress.Parse(IPHostInterfacesCBox.Text), 0));
            //Встановлюємо параметри сокету

            _MainSocket.SetSocketOption(SocketOptionLevel.IP, //Застосовується лише до IP-пакетів

            SocketOptionName.HeaderIncluded, //Встановлюємо включення заголовка в true
                                true);

            byte[] byTrue = new byte[4] { 1, 0, 0, 0 };
            byte[] byOut = new byte[4] { 1, 0, 0, 0 }; //Захоплюємо вихідні пакети

            //Socket.IOControl аналогічний методу  WSAIoctl

            _MainSocket.IOControl(IOControlCode.ReceiveAll, byTrue, byOut); //Еквівалент константи SIO_RCVALL Winsock 2

            //Починаємо отримувати пакети асинхронно
            _MainSocket.BeginReceive(_ByteData, 0, _ByteData.Length, SocketFlags.None,
                new AsyncCallback(OnReceive), null);

          } catch (Exception ex) {
            MessageBox.Show(ex.Message, "Аналіз пакетів", MessageBoxButtons.OK, MessageBoxIcon.Error);
          }
        } else {
          StartAnalizeBtn.Text = "&Аналізувати";
          PacketCountTimer.Stop();
          _BContinueCapturing = false;
          //Щоб припинити захоплення пакетів, закриваємо сокет
          _MainSocket.Close();
        }
      }
    }
    private void OnReceive(IAsyncResult ar) {
      try {
        int nReceived = _MainSocket.EndReceive(ar);
        //Алізуємо отримані байти...
        ParseData(_ByteData, nReceived);
        if (_BContinueCapturing) {
          _ByteData = new byte[4096];
          //Ще один виклик BeginReceive, для продовження отримання вхідних пакетів
          _MainSocket.BeginReceive(_ByteData, 0, _ByteData.Length, SocketFlags.None,
              new AsyncCallback(OnReceive), null);
```

```
        }
      } catch (ObjectDisposedException) {
      } catch (Exception ex) {
        MessageBox.Show(ex.Message, "Аналіз пакетів",
MessageBoxButtons.OK, MessageBoxIcon.Error);
      }
    }
    private void ParseData(byte[] byteData, int nReceived) {
      TreeNode rootNode = new TreeNode();
      //Оскільки всі пакети протоколу інкапсульовані в
дейтаграмі IP, ми починаємо з аналізу заголовка IP і
дивимось, які дані протоколу він передає
      IPHeader ipHeader = new IPHeader(byteData,
nReceived);
      TreeNode ipNode = MakeIPTreeNode(ipHeader);
      rootNode.Nodes.Add(ipNode);

      //Тепер відповідно до протоколу, що передається
дейтаграмою IP, ми аналізуємо поле даних дейтаграми
      switch (ipHeader.ProtocolType) {
        case Protocol.TCP:
          TCPHeader tcpHeader = new
TCPHeader(ipHeader.Data,            //IPHeader.Data
зберігає наявні дані переноситься дейтаграмою IP
                                    ipHeader.MessageLength);
//Довжина поля даних

          TreeNode tcpNode =
MakeTCPTreeNode(tcpHeader);
          rootNode.Nodes.Add(tcpNode);
          //Якщо порт дорівнює 53, то основним
протоколом є DNS
          //Примітка: DNS може використовувати TCP або
UDP, тому перевірка виконується двічі
          if (tcpHeader.DestinationPort == "53" ||
tcpHeader.SourcePort == "53") {
            TreeNode dnsNode =
MakeDNSTreeNode(tcpHeader.Data,
(int)tcpHeader.MessageLength);
            rootNode.Nodes.Add(dnsNode);
          }
          break;
        case Protocol.UDP:
          UDPHeader udpHeader = new
UDPHeader(ipHeader.Data,            //IPHeader.Data
зберігає дані, які передаються IP-дейтаграмою
                                  (int)ipHeader.MessageLength);
//Довжина поля даних
          TreeNode udpNode =
MakeUDPTreeNode(udpHeader);
          rootNode.Nodes.Add(udpNode);

          //Якщо порт дорівнює 53, то основним
протоколом є DNS
          //Примітка: DNS може використовувати TCP або
UDP, тому перевірка виконується двічі
          if (udpHeader.DestinationPort == "53" ||
udpHeader.SourcePort == "53") {

            TreeNode dnsNode =
MakeDNSTreeNode(udpHeader.Data,
                                    //Довжина заголовка UDP
завжди становить вісім байтів, тому ми віднімаємо її із
загальної суми,  щоб знайти довжину даних

Convert.ToInt32(udpHeader.Length) - 8);
            rootNode.Nodes.Add(dnsNode);
          }
          break;
        case Protocol.Unknown:
          break;
      }
      AddTreeNode addTreeNode = new
AddTreeNode(OnAddTreeNode);
      rootNode.Text = ipHeader.SourceAddress.ToString() +
"-" + ipHeader.DestinationAddress.ToString();
      //Потокове безпечне додавання вузлів
      treeView.Invoke(addTreeNode, new object[] {
rootNode });
    }

    //Допоміжна функція, яка повертає інформацію, що
міститься в заголовку UDP, як вузол дерева
    private TreeNode MakeIPTreeNode(IPHeader ipHeader)
{
      TreeNode ipNode = new TreeNode();
      ipNode.Text = "IP";
      ipNode.Nodes.Add("Ver: " + ipHeader.Version);
      ipNode.Nodes.Add("Header Length: " +
ipHeader.HeaderLength);
      ipNode.Nodes.Add("Differntiated Services: " +
ipHeader.DifferentiatedServices);
      ipNode.Nodes.Add("Total Length: " +
ipHeader.TotalLength);
      ipNode.Nodes.Add("Identification: " +
ipHeader.Identification);
      ipNode.Nodes.Add("Flags: " + ipHeader.Flags);
      ipNode.Nodes.Add("Fragmentation Offset: " +
ipHeader.FragmentationOffset);
      ipNode.Nodes.Add("Time to live: " + ipHeader.TTL);

      ////////////////////////////////
      _Count++;
      ipNode.Nodes.Add("Count: " + _Count);
      switch (ipHeader.ProtocolType) {
        case Protocol.TCP:
          ipNode.Nodes.Add("Protocol: " + "TCP");
          break;
        case Protocol.UDP:
          ipNode.Nodes.Add("Protocol: " + "UDP");
          break;
        case Protocol.Unknown:
          ipNode.Nodes.Add("Protocol: " + "Unknown");
          break;
      }
      ipNode.Nodes.Add("Checksum: " +
ipHeader.Checksum);
      ipNode.Nodes.Add("Source: " +
ipHeader.SourceAddress.ToString());
      ipNode.Nodes.Add("Destination: " +
ipHeader.DestinationAddress.ToString());
      return ipNode;
    }

    //Допоміжна функція, яка повертає інформацію, що
міститься в заголовку UDP, як вузол дерева
    private TreeNode MakeTCPTreeNode(TCPHeader
tcpHeader) {
      TreeNode tcpNode = new TreeNode();
      tcpNode.Text = "TCP";
      tcpNode.Nodes.Add("Source Port: " +
tcpHeader.SourcePort);
      tcpNode.Nodes.Add("Destination Port: " +
tcpHeader.DestinationPort);
      tcpNode.Nodes.Add("Sequence Number: " +
tcpHeader.SequenceNumber);
      if (tcpHeader.AcknowledgementNumber != "")
        tcpNode.Nodes.Add("Acknowledgement Number: " +
tcpHeader.AcknowledgementNumber);
      tcpNode.Nodes.Add("Header Length: " +
```

```
tcpHeader.HeaderLength);
      tcpNode.Nodes.Add("Flags: " + tcpHeader.Flags);
      tcpNode.Nodes.Add("Window Size: " +
tcpHeader.WindowSize);
      tcpNode.Nodes.Add("Checksum: " +
tcpHeader.Checksum);
      if (tcpHeader.UrgentPointer != "")
        tcpNode.Nodes.Add("Urgent Pointer: " +
tcpHeader.UrgentPointer);
      return tcpNode;
    }

    //Допоміжна функція, яка повертає інформацію, що
міститься в заголовку UDP, як вузол дерева
    private TreeNode MakeUDPTreeNode(UDPHeader
udpHeader) {
      TreeNode udpNode = new TreeNode();
      udpNode.Text = "UDP";
      udpNode.Nodes.Add("Source Port: " +
udpHeader.SourcePort);
      udpNode.Nodes.Add("Destination Port: " +
udpHeader.DestinationPort);
      udpNode.Nodes.Add("Length: " + udpHeader.Length);
      udpNode.Nodes.Add("Checksum: " +
udpHeader.Checksum);
      return udpNode;
    }

    //Допоміжна функція, яка повертає інформацію, що
міститься в заголовку UDP, як вузол дерева
    private TreeNode MakeDNSTreeNode(byte[] byteData,
int nLength) {
      DNSHeader dnsHeader = new DNSHeader(byteData,
nLength);
      TreeNode dnsNode = new TreeNode();
      dnsNode.Text = "DNS";
      dnsNode.Nodes.Add("Identification: " +
dnsHeader.Identification);
      dnsNode.Nodes.Add("Flags: " + dnsHeader.Flags);
      dnsNode.Nodes.Add("Questions: " +
dnsHeader.TotalQuestions);
      dnsNode.Nodes.Add("Answer RRs: " +
dnsHeader.TotalAnswerRRs);
      dnsNode.Nodes.Add("Authority RRs: " +
dnsHeader.TotalAuthorityRRs);
      dnsNode.Nodes.Add("Additional RRs: " +
dnsHeader.TotalAdditionalRRs);
      return dnsNode;
    }
    private void OnAddTreeNode(TreeNode node) {
      treeView.Nodes.Add(node);
    }
    private void AnalizeForm_Load(object sender,
EventArgs e) {
      string strIP = null;
      IPHostEntry HosyEntry =
Dns.GetHostEntry((Dns.GetHostName()));
      if (HosyEntry.AddressList.Length > 0) {
        foreach (IPAddress ip in HosyEntry.AddressList) {
          strIP = ip.ToString();
          IPHostInterfacesCBox.Items.Add(strIP);
        }
      }
    }
    private void AnalizeForm_FormClosing(object sender,
FormClosingEventArgs e) {
      if (_BContinueCapturing) {
        _MainSocket.Close();
      }
    }
    private void PacketCountTimer_Tick(object sender,
EventArgs e) {
      AnalizePackets oneAnalizePackets = new
AnalizePackets();
      oneAnalizePackets.PacketsCount = _Count;
      oneAnalizePackets.PacketsDate = DateTime.Now;
      _AnalizePacketsList.Add(oneAnalizePackets);
      _Count = 0;
      BildGraphics();
    }
    private void SaveBtn_Click(object sender, EventArgs e)
{
      if (IsDataEnteringCorrect()) {
        StartAnalizeBtn.Text = "&Аналізувати";
        PacketCountTimer.Stop();

_AnalizeDataProvider.InsertAnalizeData(AnalizeDataNam
eTBox.Text);
        int lastAnalizeDataId =
_AnalizeDataProvider.GetLastRecords();
        for (int i = 0; i < _AnalizePacketsList.Count; i++) {
          _AnalizePacketsList[i].AnalizeDataId =
lastAnalizeDataId;
        }

_LogsProvider.InsertLogs(LoginForm.CurrentUser.UsersId
, "Було проведено аналіз пакетів та збережено дані під
назвою '" +
          AnalizeDataNameTBox.Text + "'", DateTime.Now);

_AnalizePacketsPrivider.InsertBatchAnalizePackets(_Anali
zePacketsList);
        _PocketsValues.Clear();
        _DateTimes.Clear();
        treeView.Nodes.Clear();
        AnalizeDataNameTBox.Text = String.Empty;
      }
    }
    private bool IsDataEnteringCorrect() {
      bool isCorrect = true;
      if
(_Validation.IsDataEntering(AnalizeDataNameTBox.Text)
) {
        AnalizeDataNameValiadtionLbl.Text =
NamesMy.ProgramButtons.RequiredValidation;
      } else {
        AnalizeDataNameValiadtionLbl.Text =
NamesMy.ProgramButtons.ErrorValidation;
        isCorrect = false;
      }
      return isCorrect;
    }
    private bool IsDataSelected() {
      bool isCorrect = true;
      if (IPHostInterfacesCBox.SelectedItem != null) {
        IPHostInterfacesValiadtionLbl.Text =
NamesMy.ProgramButtons.RequiredValidation;
      } else {
        IPHostInterfacesValiadtionLbl.Text =
NamesMy.ProgramButtons.ErrorValidation;
        isCorrect = false;
        MessageBox.Show("Виберіть інтерфейс для
захоплення пакетів.", "Увага!",  MessageBoxButtons.OK,
MessageBoxIcon.Error);
      }
      return isCorrect;
    }
    private void GraphicsPrepare() {
      _PocketsValues.Add(0);
```

```
_DateTimes.Add(DateTime.Now.ToShortTimeString());

      GraphicsCC.AxisX.Add(new LiveCharts.Wpf.Axis() {
        Title = "Час",
        Labels = _DateTimes
      });

      _PocketsLine.Title = "Кількість пакетів";
      _PocketsLine.Values = _PocketsValues;

      _Series.Add(_PocketsLine);
      GraphicsCC.Series = _Series;
    }
    private void BildGraphics() {

_PocketsValues.Add(_AnalizePacketsList[_AnalizePackets
List.Count - 1].PacketsCount);

_DateTimes.Add(_AnalizePacketsList[_AnalizePacketsList
.Count - 1].PacketsDate.ToShortTimeString());
    }
  }
}
```

Лістинг 2. Код класу «OpenAnalizeForm»

```
using LiveCharts;
using LiveCharts.Wpf;
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using TCPIPAnalizer.AppCode;
using TCPIPAnalizer.Providers;

namespace TCPIPAnalizer.Forms.Sniffer {
  public partial class OpenAnalizeForm : Form {
    private int _selectedRowIndex = 0;
    private AnalizeDataProvider _AnalizeDataProvider =
new AnalizeDataProvider();
    private List<AnalizeData> _AnalizeDataList = new
List<AnalizeData>();
    private AnalizePacketsPrivider _AnalizePacketsProvider
= new AnalizePacketsPrivider();
    private List<AnalizePackets> _AnalizePacketsList =
new List<AnalizePackets>();

    public OpenAnalizeForm() {
      InitializeComponent();
      DataLoad();
    }

    private void DataLoad() {
      int firstRowIndex = 0;
      if
(AnalizeDataGridView.FirstDisplayedScrollingRowIndex
> 0) {
        firstRowIndex =
AnalizeDataGridView.FirstDisplayedScrollingRowIndex;
      }
      try {
        _AnalizeDataList =
_AnalizeDataProvider.GetAllAnalizeData();
        LoadDataInAnalizeDataGridView(_AnalizeDataList);
        if (_selectedRowIndex ==
AnalizeDataGridView.Rows.Count) {
          _selectedRowIndex =
AnalizeDataGridView.Rows.Count - 1;
        }
        if (_selectedRowIndex >= 0) {

AnalizeDataGridView.FirstDisplayedScrollingRowIndex =
firstRowIndex;

AnalizeDataGridView.Rows[_selectedRowIndex].Selected
= true;
        }
      } catch { }
    }

    private void
LoadDataInAnalizeDataGridView(List<AnalizeData>
AnalizeDataList) {
      AnalizeDataGridView.DataSource = null;
      AnalizeDataGridView.Columns.Clear();
      AnalizeDataGridView.AutoGenerateColumns = false;
      AnalizeDataGridView.RowHeadersVisible = false;

      AnalizeDataGridView.DataSource = AnalizeDataList;

      if (AnalizeDataList.Count > 0) {
        if (AnalizeDataList[0].Message ==
NamesMy.NoDataNames.NoDataInAnalizeData) {
          DataGridViewColumn messageColumn = new
DataGridViewTextBoxColumn();
          messageColumn.DataPropertyName = "Message";
          messageColumn.Width =
AnalizeDataGridView.Width -
NamesMy.SizeOptins.MinusSizePanel;

AnalizeDataGridView.Columns.Add(messageColumn);
        } else {
          DataGridViewColumn DetailIdColumn = new
DataGridViewTextBoxColumn();
          DetailIdColumn.DataPropertyName =
"AnalizeDataId";

AnalizeDataGridView.Columns.Add(DetailIdColumn);
          AnalizeDataGridView.Columns[0].Visible = false;

          DataGridViewColumn numberColumn = new
DataGridViewTextBoxColumn();
          numberColumn.HeaderText = "№ ";
          numberColumn.DataPropertyName = "Number";
          numberColumn.DefaultCellStyle.Alignment =
DataGridViewContentAlignment.MiddleRight;
          numberColumn.Width =
NamesMy.SizeOptins.NumberSize;

AnalizeDataGridView.Columns.Add(numberColumn);

          DataGridViewColumn AnalizeDataNameColumn =
new DataGridViewTextBoxColumn();
          AnalizeDataNameColumn.HeaderText = "Аналізи
пакетів даних";
          AnalizeDataNameColumn.DataPropertyName =
"AnalizeDataName";
          AnalizeDataNameColumn.Width =
NamesMy.SizeOptins.NameSize;

AnalizeDataGridView.Columns.Add(AnalizeDataNameCol
umn);


        }
        for (int i = 0; i <
AnalizeDataGridView.Columns.Count; i++) {
```

```
AnalizeDataGridView.Columns[i].HeaderCell.Style.BackColor = Color.LightGray;
        }
      }
    }

    private void AnalizeDataGridView_CellClick(object sender, DataGridViewCellEventArgs e) {
      if (e.RowIndex >= 0 && AnalizeDataGridView[0, e.RowIndex].Value.ToString() != _AnalizeDataList[0].Message) {
        _selectedRowIndex = Convert.ToInt32(AnalizeDataGridView[0, e.RowIndex].Value.ToString());
      }
      BildGraphics();
    }

    private void BildGraphics() {
      GraphicsCC.AxisX.Clear();
      _AnalizePacketsList = _AnalizePacketsProvider.GetAllAnalizePacketsByAnalizeDataId(_selectedRowIndex);

      SeriesCollection series = new SeriesCollection();
      ChartValues<double> purchaseValue = new ChartValues<double>();
      ChartValues<double> sellingValue = new ChartValues<double>();
      List<string> dates = new List<string>();
      for (int i = 0; i < _AnalizePacketsList.Count; i++) {

purchaseValue.Add(_AnalizePacketsList[i].PacketsCount);

dates.Add(_AnalizePacketsList[i].PacketsDate.ToShortTimeString());
      }


      GraphicsCC.AxisX.Add(new LiveCharts.Wpf.Axis() {
        Title = "Дати",
        Labels = dates
      });

      LineSeries purchaseLine = new LineSeries();
      purchaseLine.Title = "Значення";
      purchaseLine.Values = purchaseValue;

      series.Add(purchaseLine);
      GraphicsCC.Series = series;
    }

  }
}
```

Лістинг 3. Код класу «LoginForm»

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using TCPIPAnalizer.AppCode;
using TCPIPAnalizer.Providers;

namespace TCPIPAnalizer.Forms.Systems {
  public partial class LoginForm : Form {
    public static Users CurrentUser = new Users();

    private UsersProvider _UserProvider = new UsersProvider();
    private ValidationMy _validation = new ValidationMy();
    private LogsProvider _LogsProvider = new LogsProvider();
    private List<Users> _UserList = new List<Users>();

    public LoginForm() {
      InitializeComponent();
      LoadAllDate();
    }

    private void SubmitBtn_Click(object sender, EventArgs e) {
      GetSubmitData();
    }

    private void DataLoad() {
      _LogsProvider.InsertLogs(CurrentUser.UsersId, "Користувач ввійшов в систему", DateTime.Now);
      this.Visible = false;
      if (CurrentUser.RoleId == 1) {
        (new AnalizerMDI()).ShowDialog();
      } else {
        (new AnalizerMDI()).ShowDialog();
      }
      _LogsProvider.InsertLogs(CurrentUser.UsersId, "Користувач вийшов із системи", DateTime.Now);
      this.Close();
    }

    private bool IsDataEnteringCorrect() {
      bool isCorrect = true;
      if (_validation.IsDataEntering(UserNameCBox.Text)) {
        UserNameValiadtionLbl.Text = NamesMy.ProgramButtons.RequiredValidation;
      } else {
        UserNameValiadtionLbl.Text = NamesMy.ProgramButtons.ErrorValidation;
        isCorrect = false;
      }
      if (_validation.IsDataEntering(UserPasswordTbx.Text)) {
        UserPasswordValidation.Text = NamesMy.ProgramButtons.RequiredValidation;
      } else {
        UserPasswordValidation.Text = NamesMy.ProgramButtons.ErrorValidation;
        isCorrect = false;
      }
      return isCorrect;
    }

    private void LoadAllDate() {
      _UserList = _UserProvider.GetAllUsersListForCBox();
      UserNameCBox.DataSource = _UserList;
      UserNameCBox.AutoCompleteMode = AutoCompleteMode.SuggestAppend;
      UserNameCBox.AutoCompleteSource = AutoCompleteSource.ListItems;
      UserNameCBox.ValueMember = "UsersId";
      UserNameCBox.DisplayMember = "UsersName";
    }

    private void GetSubmitData() {
      try {
        if (IsDataEnteringCorrect()) {
```

```
        List<Users> listUsers = new List<Users>();
        listUsers =
_UserProvider.SelectedUsersByUsersNameAndUsersPass
word(UserNameCBox.Text, UserPasswordTbx.Text);
        if (listUsers.Count > 0) {
          CurrentUser = listUsers[0];
          DataLoad();
        } else {

MessageBox.Show(NamesMy.MessageBoxExaption.ThisU
serLoginAndUserPasswordNotExistInSystem,
NamesMy.MessageBoxExaption.CaptionMessage);
        }
       }
     } catch {
       MessageBox.Show("Немає з'єднання!");
     }
   }

  }
}
```

Лістинг 4. Код класу «SystemLogForm»

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using TCPIPAnalizer.AppCode;
using TCPIPAnalizer.Providers;

namespace TCPIPAnalizer.Forms.Systems {
  public partial class SystemLogForm : Form {
    private int _selectedRowIndex = 0;
    private LogsProvider _LogsProvider = new
LogsProvider();
    private List<Logs> _LogsList = new List<Logs>();
    public SystemLogForm() {
      InitializeComponent();
      DataLoad();
    }

    private void DataLoad() {
      int firstRowIndex = 0;
      if (LogsGridView.FirstDisplayedScrollingRowIndex >
0) {
        firstRowIndex =
LogsGridView.FirstDisplayedScrollingRowIndex;
      }
      try {
        _LogsList = _LogsProvider.GetAllLogs();
        LoadDataInLogsGridView(_LogsList);
        if (_selectedRowIndex ==
LogsGridView.Rows.Count) {
          _selectedRowIndex = LogsGridView.Rows.Count -
1;
        }
        if (_selectedRowIndex >= 0) {
          LogsGridView.FirstDisplayedScrollingRowIndex =
firstRowIndex;
          LogsGridView.Rows[_selectedRowIndex].Selected
= true;
        }
      } catch { }
    }

    private void LoadDataInLogsGridView(List<Logs>
LogsList) {
      LogsGridView.DataSource = null;
      LogsGridView.Columns.Clear();
      LogsGridView.AutoGenerateColumns = false;
      LogsGridView.RowHeadersVisible = false;

      LogsGridView.DataSource = LogsList;

      if (LogsList.Count > 0) {
        if (LogsList[0].Message ==
NamesMy.NoDataNames.NoDataInLogs) {
          DataGridViewColumn messageColumn = new
DataGridViewTextBoxColumn();
          messageColumn.DataPropertyName = "Message";
          messageColumn.Width = LogsGridView.Width -
NamesMy.SizeOptins.MinusSizePanel;
          LogsGridView.Columns.Add(messageColumn);
        } else {
          DataGridViewColumn DetailIdColumn = new
DataGridViewTextBoxColumn();
          DetailIdColumn.DataPropertyName = "LogsId";
          LogsGridView.Columns.Add(DetailIdColumn);
          LogsGridView.Columns[0].Visible = false;

          DataGridViewColumn numberColumn = new
DataGridViewTextBoxColumn();
          numberColumn.HeaderText = "№ ";
          numberColumn.DataPropertyName = "Number";
          numberColumn.DefaultCellStyle.Alignment =
DataGridViewContentAlignment.MiddleRight;
          numberColumn.Width =
NamesMy.SizeOptins.NumberSize;
          LogsGridView.Columns.Add(numberColumn);

          DataGridViewColumn UsersNameColumn = new
DataGridViewTextBoxColumn();
          UsersNameColumn.HeaderText = "Користувач";
          UsersNameColumn.DataPropertyName =
"UsersName";
          UsersNameColumn.Width = 150;
          LogsGridView.Columns.Add(UsersNameColumn);

          DataGridViewColumn EventNameShowColumn =
new DataGridViewTextBoxColumn();
          EventNameShowColumn.HeaderText = "Подія";
          EventNameShowColumn.DataPropertyName =
"EventNameShow";
          EventNameShowColumn.Width = 500;

LogsGridView.Columns.Add(EventNameShowColumn);

          DataGridViewColumn EvendDateColumn = new
DataGridViewTextBoxColumn();
          EvendDateColumn.HeaderText = "Дата";
          EvendDateColumn.DataPropertyName =
"EvendDate";
          EvendDateColumn.Width =
NamesMy.SizeOptins.Date;
          LogsGridView.Columns.Add(EvendDateColumn);

        }
        for (int i = 0; i < LogsGridView.Columns.Count; i++)
{

LogsGridView.Columns[i].HeaderCell.Style.BackColor =
Color.LightGray;
        }
      }
    }
```

```
  }
}
```

Лістинг 5. Код класу «UpdateUsersForm»

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using TCPIPAnalizer.AppCode;
using TCPIPAnalizer.Providers;

namespace TCPIPAnalizer.Forms.Systems {
  public partial class UpdateUsersForm : Form {
    private int _UserId = 0;
    private Users _selectedUser = new Users();
    private UsersProvider _UserProvider = new
UsersProvider();
    private ValidationMy _validation = new ValidationMy();
    private RoleApp _RoleApp = new RoleApp();
    private List<Role> _RoleList = new List<Role>();

    public UpdateUsersForm(int UserId) {
      InitializeComponent();
      _UserId = UserId;
      LoadAllDate();
    }

    private void SaveBtn_Click(object sender, EventArgs e)
{
      if (IsDataEnteringCorrect()) {
        _UserProvider.UpdateUsers(FirstNameTBox.Text,
LastNameTBox.Text, UserLoginTbx.Text,
PasswordTbx.Text,
          Convert.ToInt32(RolesCBox.SelectedValue),
DescriptionTbx.Text, _UserId);
        this.Close();
      }
    }

    private void DeleteBtn_Click(object sender, EventArgs
e) {
      if (MessageBox.Show("Ви дійсно хочете видалити
цей елемент?", "Видалити", MessageBoxButtons.YesNo)
== DialogResult.Yes) {
        _UserProvider.DeleteUsersByUsersId(_UserId);
        this.Close();
      }
    }

    private void ExitBtn_Click(object sender, EventArgs e) {
      this.Close();
    }

    private void LoadAllDate() {
      _RoleList = _RoleApp.GetRoleList();
      RolesCBox.DataSource = _RoleList;
      RolesCBox.ValueMember = "RoleId";
      RolesCBox.DisplayMember = "RoleName";

      _selectedUser =
_UserProvider.SelectedUsersByUsersId(_UserId);
      FirstNameTBox.Text = _selectedUser.FirstName;
      LastNameTBox.Text = _selectedUser.LastName;
      UserLoginTbx.Text = _selectedUser.UsersName;
      RolesCBox.SelectedValue = _selectedUser.RoleId;
    }

    private bool IsDataEnteringCorrect() {
      bool isCorrect = true;
      if (_validation.IsDataEntering(FirstNameTBox.Text)) {
        FirstNameValiadtionLbl.Text =
NamesMy.ProgramButtons.RequiredValidation;
      } else {
        FirstNameValiadtionLbl.Text =
NamesMy.ProgramButtons.ErrorValidation;
        isCorrect = false;
      }
      if (_validation.IsDataEntering(LastNameTBox.Text)) {
        LastNameValiadtionLbl.Text =
NamesMy.ProgramButtons.RequiredValidation;
      } else {
        LastNameValiadtionLbl.Text =
NamesMy.ProgramButtons.ErrorValidation;
        isCorrect = false;
      }
      if (_validation.IsPasswordMatch(PasswordTbx.Text,
RePasswordTbx.Text)) {
        PasswordAndRePasswordDontMatchLbl.Visible =
false;
      } else {
        PasswordAndRePasswordDontMatchLbl.Visible =
true;
        isCorrect = false;
      }
      if (_validation.IsDataEntering(UserLoginTbx.Text)) {
        UserLoginValidationLbl.Text =
NamesMy.ProgramButtons.RequiredValidation;
      } else {
        UserLoginValidationLbl.Text =
NamesMy.ProgramButtons.ErrorValidation;
        isCorrect = false;
      }
      if (_validation.IsDataEntering(PasswordTbx.Text)) {
        PasswordValidationLbl.Text =
NamesMy.ProgramButtons.RequiredValidation;
      } else {
        PasswordValidationLbl.Text =
NamesMy.ProgramButtons.ErrorValidation;
        isCorrect = false;
      }
      if (_validation.IsDataEntering(RePasswordTbx.Text)) {
        RePasswordValidationLbl.Text =
NamesMy.ProgramButtons.RequiredValidation;
      } else {
        RePasswordValidationLbl.Text =
NamesMy.ProgramButtons.ErrorValidation;
        isCorrect = false;
      }
      return isCorrect;
    }
  }
}
```

Лістинг 6. Код класу «UsersForm»

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using TCPIPAnalizer.AppCode;
```

```
using TCPIPAnalizer.Providers;

namespace TCPIPAnalizer.Forms.Systems {
  public partial class UsersForm : Form {
    private int _selectedRowIndex = 0;
    ValidationMy _validation = new ValidationMy();
    private UsersProvider _UserProvider = new
UsersProvider();
    private List<Users> _UserList = new List<Users>();
    private RoleApp _RoleApp = new RoleApp();
    private List<Role> _RoleList = new List<Role>();
    public UsersForm() {
      InitializeComponent();
      LoadAllDate();
      DataLoad();
    }

    private void AddBtn_Click(object sender, EventArgs e)
{
      if (IsDataEnteringCorrect()) {
        _UserProvider.InsertUsers(FirstNameTBox.Text,
LastNameTBox.Text, UserLoginTbx.Text,
PasswordTbx.Text,
          Convert.ToInt32(RolesCBox.SelectedValue),
DescriptionTbx.Text);
        DataLoad();
        ClearAllControls();
      }
    }

    private void ClearBtn_Click(object sender, EventArgs e)
{
      ClearAllControls();
    }

    private void ExitBtn_Click(object sender, EventArgs e) {
      this.Close();
    }

    private void UsersGridView_CellClick(object sender,
DataGridViewCellEventArgs e) {
      if (e.RowIndex >= 0 && UsersGridView[0,
e.RowIndex].Value.ToString() != _UserList[0].Message) {
        _selectedRowIndex = e.RowIndex;
        UpdateUsersForm updateUsersForm = new
UpdateUsersForm(Convert.ToInt32(UsersGridView[0,
e.RowIndex].Value.ToString()));
        updateUsersForm.ShowDialog();
        DataLoad();
      }
    }

    private void DataLoad() {
      int firstRowIndex = 0;
      if (UsersGridView.FirstDisplayedScrollingRowIndex >
0) {
        firstRowIndex =
UsersGridView.FirstDisplayedScrollingRowIndex;
      }
      try {
        _UserList = _UserProvider.GetAllUsers();
        LoadDataInKlientGridView(_UserList);
        if (_selectedRowIndex ==
UsersGridView.Rows.Count) {
          _selectedRowIndex = UsersGridView.Rows.Count -
1;
        }
        if (_selectedRowIndex >= 0) {
          UsersGridView.FirstDisplayedScrollingRowIndex =
firstRowIndex;
          UsersGridView.Rows[_selectedRowIndex].Selected
= true;
        }
      } catch { }
    }

    private void LoadDataInKlientGridView(List<Users>
UserList) {
      UsersGridView.DataSource = null;
      UsersGridView.Columns.Clear();
      UsersGridView.AutoGenerateColumns = false;
      UsersGridView.RowHeadersVisible = false;

      UsersGridView.DataSource = UserList;

      if (UserList.Count > 0) {
        if (UserList[0].Message ==
NamesMy.NoDataNames.NoDataInUsers) {
          DataGridViewColumn messageColumn = new
DataGridViewTextBoxColumn();
          messageColumn.DataPropertyName = "Message";
          messageColumn.Width = UsersGridView.Width -
NamesMy.SizeOptins.MinusSizePanel;
          UsersGridView.Columns.Add(messageColumn);
        } else {
          DataGridViewColumn deviseIdColumn = new
DataGridViewTextBoxColumn();
          deviseIdColumn.DataPropertyName = "UsersId";
          UsersGridView.Columns.Add(deviseIdColumn);
          UsersGridView.Columns[0].Visible = false;

          DataGridViewColumn numberColumn = new
DataGridViewTextBoxColumn();
          numberColumn.HeaderText = "№ п/п";
          numberColumn.DataPropertyName = "Number";
          numberColumn.DefaultCellStyle.Alignment =
DataGridViewContentAlignment.MiddleRight;
          numberColumn.Width =
NamesMy.SizeOptins.NumberSize;
          UsersGridView.Columns.Add(numberColumn);

          DataGridViewColumn lastNameColumn = new
DataGridViewTextBoxColumn();
          lastNameColumn.HeaderText = "Фимилия";
          lastNameColumn.DataPropertyName = "LastName";
          lastNameColumn.Width =
NamesMy.SizeOptins.Name;
          UsersGridView.Columns.Add(lastNameColumn);

          DataGridViewColumn firstNameColumn = new
DataGridViewTextBoxColumn();
          firstNameColumn.HeaderText = "Имя";
          firstNameColumn.DataPropertyName =
"FirstName";
          firstNameColumn.Width =
NamesMy.SizeOptins.Name;
          UsersGridView.Columns.Add(firstNameColumn);

          DataGridViewColumn UserNameColumn = new
DataGridViewTextBoxColumn();
          UserNameColumn.HeaderText = "Логин";
          UserNameColumn.DataPropertyName =
"UsersName";
          UserNameColumn.Width =
NamesMy.SizeOptins.Name;
          UsersGridView.Columns.Add(UserNameColumn);

          DataGridViewColumn roleNameColumn = new
DataGridViewTextBoxColumn();
          roleNameColumn.HeaderText = "Роль";
```

```
        roleNameColumn.DataPropertyName =
"RoleName";
        roleNameColumn.Width =
NamesMy.SizeOptins.Name;
        UsersGridView.Columns.Add(roleNameColumn);
      }
      for (int i = 0; i < UsersGridView.Columns.Count; i++)
{

UsersGridView.Columns[i].HeaderCell.Style.BackColor =
Color.LightGray;
      }
    }
  }

  private void ClearAllControls() {
    FirstNameTBox.Text = String.Empty;
    LastNameTBox.Text = String.Empty;
    DescriptionTbx.Text = String.Empty;
    UserLoginTbx.Text = String.Empty;
    PasswordTbx.Text = String.Empty;
    RePasswordTbx.Text = String.Empty;
  }


  private bool IsDataEnteringCorrect() {
    bool isCorrect = true;
    if (_validation.IsDataEntering(FirstNameTBox.Text)) {
      FirstNameValiadtionLbl.Text =
NamesMy.ProgramButtons.RequiredValidation;
    } else {
      FirstNameValiadtionLbl.Text =
NamesMy.ProgramButtons.ErrorValidation;
      isCorrect = false;
    }
    if (_validation.IsDataEntering(LastNameTBox.Text)) {
      LastNameValiadtionLbl.Text =
NamesMy.ProgramButtons.RequiredValidation;
    } else {
      LastNameValiadtionLbl.Text =
NamesMy.ProgramButtons.ErrorValidation;
      isCorrect = false;
    }
    if (_validation.IsPasswordMatch(PasswordTbx.Text,
RePasswordTbx.Text)) {
      PasswordAndRePasswordDontMatchLbl.Visible =
false;
      PasswordAndRePasswordDontMatchLbl.Text =
NamesMy.ProgramButtons.RequiredValidation;
    } else {
      PasswordAndRePasswordDontMatchLbl.Visible =
true;
      PasswordAndRePasswordDontMatchLbl.Text =
NamesMy.ProgramButtons.ErrorValidation;
      isCorrect = false;
    }
    if (_validation.IsDataEntering(UserLoginTbx.Text)) {
      UserLoginValidationLbl.Text =
NamesMy.ProgramButtons.RequiredValidation;
    } else {
      UserLoginValidationLbl.Text =
NamesMy.ProgramButtons.ErrorValidation;
      isCorrect = false;
    }
    if (_validation.IsDataEntering(PasswordTbx.Text)) {
      PasswordValidationLbl.Text =
NamesMy.ProgramButtons.RequiredValidation;
    } else {
      PasswordValidationLbl.Text =
NamesMy.ProgramButtons.ErrorValidation;
      isCorrect = false;
    }
    if (_validation.IsDataEntering(RePasswordTbx.Text)) {
      RePasswordValidationLbl.Text =
NamesMy.ProgramButtons.RequiredValidation;
    } else {
      RePasswordValidationLbl.Text =
NamesMy.ProgramButtons.ErrorValidation;
      isCorrect = false;
    }
    return isCorrect;
  }

  private void LoadAllDate() {
    _RoleList = _RoleApp.GetRoleList();
    RolesCBox.DataSource = _RoleList;
    RolesCBox.ValueMember = "RoleId";
    RolesCBox.DisplayMember = "RoleName";
  }

 }
}
```

Лістинг 7. Код класу «AnalizeDataProvider»

```
using System;
using System.Collections.Generic;
using System.Data;
using System.Data.OleDb;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using TCPIPAnalizer.AppCode;

namespace TCPIPAnalizer.Providers {
 class AnalizeDataProvider {
  private string _ConnString =
System.Configuration.ConfigurationSettings.AppSettings["
CONNECT"];

  public void InsertAnalizeData(string AnalizeDataName)
{
    string SqlString = "INSERT INTO AnalizeData
(AnalizeDataName" +
      ") Values(?)";

    using (OleDbConnection conn = new
OleDbConnection(_ConnString)) {
      using (OleDbCommand cmd = new
OleDbCommand(SqlString, conn)) {
        cmd.CommandType = CommandType.Text;

cmd.Parameters.AddWithValue("AnalizeDataName",
AnalizeDataName);
        conn.Open();
        cmd.ExecuteNonQuery();
        conn.Close();
      }
    }
  }

  public List<AnalizeData> GetAllAnalizeData() {
    int i = 0;
    string SqlString = "SELECT * " +
      "FROM AnalizeData";

    List<AnalizeData> listAnalizeData = new
List<AnalizeData>();
    using (OleDbConnection conn = new
OleDbConnection(_ConnString)) {
      using (OleDbCommand cmd = new
```

```
OleDbCommand(SqlString, conn)) {
        conn.Open();
        using (OleDbDataReader reader =
cmd.ExecuteReader()) {
          while (reader.Read()) {
            AnalizeData oneAnalizeData = new
AnalizeData();
            oneAnalizeData.Number = ++i;
            oneAnalizeData.AnalizeDataId =
Convert.ToInt32(reader["AnalizeDataId"]);
            oneAnalizeData.AnalizeDataName =
reader["AnalizeDataName"].ToString();
            listAnalizeData.Add(oneAnalizeData);
          }
        }
        conn.Close();
      }
    }

    if (listAnalizeData.Count == 0) {
      AnalizeData noAnalizeData = new AnalizeData();
      noAnalizeData.AnalizeDataId = 0;
      noAnalizeData.Message =
NamesMy.NoDataNames.NoDataInAnalizeData;
      listAnalizeData.Add(noAnalizeData);
    }
    return listAnalizeData;
  }

  public AnalizeData SelectedAnalizeDataById(int
AnalizeDataId) {
    string SqlString = "SELECT * " +
      "FROM AnalizeData Where AnalizeDataId=" +
AnalizeDataId.ToString();

    AnalizeData oneAnalizeData = new AnalizeData();
    using (OleDbConnection conn = new
OleDbConnection(_ConnString)) {
      using (OleDbCommand cmd = new
OleDbCommand(SqlString, conn)) {
        conn.Open();
        using (OleDbDataReader reader =
cmd.ExecuteReader()) {
          while (reader.Read()) {
            oneAnalizeData.AnalizeDataId =
Convert.ToInt32(reader["AnalizeDataId"]);
            oneAnalizeData.AnalizeDataName =
reader["AnalizeDataName"].ToString();
          }
        }
      }
      conn.Close();
    }
    return oneAnalizeData;
  }

  public void UpdateAnalizeData(string
AnalizeDataName, string Description, int AnalizeDataId) {
    string SqlString = "UPDATE AnalizeData SET
AnalizeDataName=?  " +
 "WHERE AnalizeDataId=?";

    using (OleDbConnection conn = new
OleDbConnection(_ConnString)) {
      using (OleDbCommand cmd = new
OleDbCommand(SqlString, conn)) {
        cmd.CommandType = CommandType.Text;

cmd.Parameters.AddWithValue("AnalizeDataName",
AnalizeDataName);
        cmd.Parameters.AddWithValue("AnalizeDataId",
AnalizeDataId);
        conn.Open();
        cmd.ExecuteNonQuery();
        conn.Close();
      }
    }
  }

  public void DeleteAnalizeDataByAnalizeDataId(int
AnalizeDataId) {
    string SqlString = "DELETE FROM AnalizeData
WHERE AnalizeDataId=" + AnalizeDataId.ToString();
    using (OleDbConnection conn = new
OleDbConnection(_ConnString)) {
      using (OleDbCommand cmd = new
OleDbCommand(SqlString, conn)) {
        conn.Open();
        cmd.ExecuteNonQuery();
        conn.Close();
      }
    }
  }

  public int GetLastRecords() {
    int lastRecordNumber = 0;
    string SqlString = "Select LAST (AnalizeDataId) From
AnalizeData ";
    using (OleDbConnection conn = new
OleDbConnection(_ConnString)) {
      using (OleDbCommand cmd = new
OleDbCommand(SqlString, conn)) {
        conn.Open();
        using (OleDbDataReader reader =
cmd.ExecuteReader()) {
          while (reader.Read()) {
            lastRecordNumber =
Convert.ToInt32(reader.GetValue(0));
          }
        }
      }
      conn.Close();
    }
    return lastRecordNumber;
  }

 }
}


public class AnalizeData {
 private int _Number;
 private int _AnalizeDataId;
 private string _AnalizeDataName;
 private string _Message;

 public AnalizeData() {
  _Number = 0;
  _AnalizeDataId = 0;
  _AnalizeDataName = String.Empty;
  _Message = String.Empty;
 }

 public int Number {
  set { _Number = value; }
  get { return _Number; }
 }
 public int AnalizeDataId {
  set { _AnalizeDataId = value; }
  get { return _AnalizeDataId; }
```

```
  }
  public string AnalizeDataName {
    set { _AnalizeDataName = value; }
    get { return _AnalizeDataName; }
  }
  public string Message {
    set { _Message = value; }
    get { return _Message; }
  }
}
```

Лістинг 8. Код класу «AnalizePacketsPrivider»

```
using System;
using System.Collections.Generic;
using System.Data;
using System.Data.OleDb;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace TCPIPAnalizer.Providers {
  class AnalizePacketsPrivider {
    private string _ConnString =
System.Configuration.ConfigurationSettings.AppSettings["
CONNECT"];
    public void
InsertBatchAnalizePackets(List<AnalizePackets>
AnalizePackets) {
      string SqlString = "INSERT INTO AnalizePackets
(PacketsCount, PacketsDate, AnalizeDataId) Values(?, ?,
?)";
      using (OleDbConnection conn = new
OleDbConnection(_ConnString)) {
        using (OleDbCommand cmd = new
OleDbCommand(SqlString, conn)) {
          cmd.CommandType = CommandType.Text;
          conn.Open();
          for (int i = 0; i < AnalizePackets.Count; i++) {
            cmd.Parameters.AddWithValue("PacketsCount",
AnalizePackets[i].PacketsCount);
            cmd.Parameters.AddWithValue("PacketsDate",
AnalizePackets[i].PacketsDate.ToString());
            cmd.Parameters.AddWithValue("AnalizeDataId",
AnalizePackets[i].AnalizeDataId);
            cmd.ExecuteNonQuery();
            while (cmd.Parameters.Count > 0) {
              cmd.Parameters.RemoveAt(0);
            }
          }
          conn.Close();
        }
      }
    }

    public List<AnalizePackets>
GetAllAnalizePacketsByAnalizeDataId(int AnalizeDataId)
{
      int i = 0;
      string SqlString = "SELECT * FROM AnalizePackets
WHERE AnalizeDataId=" + AnalizeDataId;

      List<AnalizePackets> AnalizePackets = new
List<AnalizePackets>();
      using (OleDbConnection conn = new
OleDbConnection(_ConnString)) {
        using (OleDbCommand cmd = new
OleDbCommand(SqlString, conn)) {
          conn.Open();
          using (OleDbDataReader reader =
cmd.ExecuteReader()) {
            while (reader.Read()) {
              AnalizePackets oneAnalizePackets = new
AnalizePackets();
              oneAnalizePackets.Number = ++i;
              oneAnalizePackets.AnalizePacketsId =
Convert.ToInt32(reader["AnalizePacketsId"]);
              oneAnalizePackets.PacketsCount =
Convert.ToInt32(reader["PacketsCount"]);
              oneAnalizePackets.PacketsDate =
Convert.ToDateTime(reader["PacketsDate"]);
              oneAnalizePackets.AnalizeDataId =
Convert.ToInt32(reader["AnalizeDataId"]);
              AnalizePackets.Add(oneAnalizePackets);
            }
          }
          conn.Close();
        }
      }
      return AnalizePackets;
    }
  }
}

public class AnalizePackets {
  private int _Number;
  private int _AnalizePacketsId;
  private int _PacketsCount;
  private DateTime _PacketsDate;
  private int _AnalizeDataId;
  private string _Message;

  public AnalizePackets() {
    _Number = 0;
    _AnalizePacketsId = 0;
    _PacketsCount = 0;
    _PacketsDate = new DateTime();
    _AnalizeDataId = 0;
  }

  public int Number {
    set { _Number = value; }
    get { return _Number; }
  }
  public int AnalizePacketsId {
    set { _AnalizePacketsId = value; }
    get { return _AnalizePacketsId; }
  }
  public int PacketsCount {
    set { _PacketsCount = value; }
    get { return _PacketsCount; }
  }
  public DateTime PacketsDate {
    set { _PacketsDate = value; }
    get { return _PacketsDate; }
  }
  public int AnalizeDataId {
    set { _AnalizeDataId = value; }
    get { return _AnalizeDataId; }
  }
  public string Message {
    set { _Message = value; }
    get { return _Message; }
  }
}
```

Лістинг 9. Код класу «LogsProvider»

```
using System;
using System.Collections.Generic;
using System.Data;
using System.Data.OleDb;
using System.Linq;
```

```
using System.Text;
using System.Threading.Tasks;
using TCPIPAnalizer.AppCode;

namespace TCPIPAnalizer.Providers {
  class LogsProvider {
    private string _ConnString =
System.Configuration.ConfigurationSettings.AppSettings["
CONNECT"];
    public void InsertLogs(int UsersId, string
EventNameShow, DateTime EvendDate) {
      string SqlString = "INSERT INTO Logs (UsersId,
EventNameShow, EvendDate) Values(?, ?, ?)";
      using (OleDbConnection conn = new
OleDbConnection(_ConnString)) {
        using (OleDbCommand cmd = new
OleDbCommand(SqlString, conn)) {
          cmd.CommandType = CommandType.Text;
          cmd.Parameters.AddWithValue("UsersId", UsersId);
          cmd.Parameters.AddWithValue("EventNameShow",
EventNameShow);
          cmd.Parameters.AddWithValue("EvendDate",
EvendDate.ToString());
          conn.Open();
          cmd.ExecuteNonQuery();
          conn.Close();
        }
      }
    }

    public List<Logs> GetAllLogs() {
      int i = 0;
      string SqlString = "SELECT Logs.LogsId,
Logs.UsersId, Logs.EventNameShow, Logs.EvendDate,
Users.UsersName " +
        "FROM Logs INNER JOIN Users ON Users.UsersId =
Logs.UsersId  ORDER BY Logs.EvendDate DESC";
      List<Logs> listAllLogs = new List<Logs>();
      using (OleDbConnection conn = new
OleDbConnection(_ConnString)) {
        using (OleDbCommand cmd = new
OleDbCommand(SqlString, conn)) {
          conn.Open();
          using (OleDbDataReader reader =
cmd.ExecuteReader()) {
            while (reader.Read()) {
              Logs oneLogs = new Logs();
              oneLogs.Number = ++i;
              oneLogs.LogsId =
Convert.ToInt32(reader["LogsId"]);
              oneLogs.UsersId =
Convert.ToInt32(reader["UsersId"]);
              oneLogs.EventNameShow =
reader["EventNameShow"].ToString();
              oneLogs.EvendDate =
Convert.ToDateTime(reader["EvendDate"]);
              oneLogs.UsersName =
reader["UsersName"].ToString();
              listAllLogs.Add(oneLogs);
            }
          }
          conn.Close();
        }
      }

      if (listAllLogs.Count == 0) {
        Logs noLogs = new Logs();
        noLogs.LogsId = 0;
        noLogs.Message =
NamesMy.NoDataNames.NoDataInLogs;
        listAllLogs.Add(noLogs);
      }
      return listAllLogs;
    }

  }
}

public class Logs {
  private int _Number;
  private int _LogsId;
  private int _UsersId;
  private string _UsersName;
  private string _EventNameShow;
  private DateTime _EvendDate;
  private string _Message;

  public Logs() {
    _Number = 0;
    _LogsId = 0;
    _UsersId = 0;
    _UsersName = String.Empty;
    _EventNameShow = String.Empty;
    _EvendDate = new DateTime();
    _Message = String.Empty;
  }

  public int Number {
    set { _Number = value; }
    get { return _Number; }
  }
  public int LogsId {
    set { _LogsId = value; }
    get { return _LogsId; }
  }
  public int UsersId {
    set { _UsersId = value; }
    get { return _UsersId; }
  }
  public string UsersName {
    set { _UsersName = value; }
    get { return _UsersName; }
  }
  public string EventNameShow {
    set { _EventNameShow = value; }
    get { return _EventNameShow; }
  }
  public DateTime EvendDate {
    set { _EvendDate = value; }
    get { return _EvendDate; }
  }
  public string Message {
    set { _Message = value; }
    get { return _Message; }
  }
}
```

Лістинг 10. Код класу «UsersProvider»

```
using System;
using System.Collections.Generic;
using System.Data;
using System.Data.OleDb;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using TCPIPAnalizer.AppCode;

namespace TCPIPAnalizer.Providers {
  class UsersProvider {
    private EncryptData _encryptData = new EncryptData();
```

```
    private string _ConnString =
System.Configuration.ConfigurationSettings.AppSettings["
CONNECT"];

    public void InsertUsers(string FirstName, string
LastName, string UsersName, string UsersPassword,
   int RoleId, string Description) {
      string SqlString = "INSERT INTO Users (FirstName,
LastName, UsersName, UsersPassword, " +
       "RoleId, Description) Values(?, ?, ?, ?, ?, ?)";

      using (OleDbConnection conn = new
OleDbConnection(_ConnString)) {
        using (OleDbCommand cmd = new
OleDbCommand(SqlString, conn)) {
          cmd.CommandType = CommandType.Text;
          cmd.Parameters.AddWithValue("FirstName",
FirstName);
          cmd.Parameters.AddWithValue("LastName",
LastName);
          cmd.Parameters.AddWithValue("UsersName",
UsersName);
          cmd.Parameters.AddWithValue("UsersPassword",
_encryptData.Encrypt(UsersPassword));
          cmd.Parameters.AddWithValue("RoleId", RoleId);
          cmd.Parameters.AddWithValue("Description",
Description);
          conn.Open();
          cmd.ExecuteNonQuery();
          conn.Close();
        }
      }
    }

    public List<Users> GetAllUsers() {
      int i = 0;
      string SqlString = "SELECT * FROM Users ORDER
BY LastName ASC";
      List<Users> listAllUsers = new List<Users>();
      using (OleDbConnection conn = new
OleDbConnection(_ConnString)) {
        using (OleDbCommand cmd = new
OleDbCommand(SqlString, conn)) {
          conn.Open();
          using (OleDbDataReader reader =
cmd.ExecuteReader()) {
            while (reader.Read()) {
              Users oneUsers = new Users();
              oneUsers.Number = ++i;
              oneUsers.UsersId =
Convert.ToInt32(reader["UsersId"]);
              oneUsers.FirstName =
reader["FirstName"].ToString();
              oneUsers.LastName =
reader["LastName"].ToString();
              oneUsers.FIO = oneUsers.LastName + " " +
oneUsers.FirstName;
              oneUsers.UsersName =
reader["UsersName"].ToString();
              oneUsers.UsersPassword =
_encryptData.Decrypt(reader["UsersPassword"].ToString()
);
              oneUsers.RoleId =
Convert.ToInt32(reader["RoleId"]);
              oneUsers.RoleName =
GetRoleName(oneUsers.RoleId);
              oneUsers.Description =
reader["Description"].ToString();
              listAllUsers.Add(oneUsers);
            }
          }
          conn.Close();
        }
      }

      if (listAllUsers.Count == 0) {
        Users noUsers = new Users();
        noUsers.UsersId = 0;
        noUsers.Message =
NamesMy.NoDataNames.NoDataInUsers;
        listAllUsers.Add(noUsers);
      }
      return listAllUsers;
    }

    private string GetRoleName(int RoleId) {
      RoleApp roleApp = new RoleApp();
      for (int i = 0; i < roleApp.GetRoleList().Count(); i++) {
        if (RoleId == roleApp.GetRoleList()[i].RoleId) {
          return roleApp.GetRoleList()[i].RoleName;
        }
      }
      return "";
    }

    public Users SelectedUsersByUsersId(int UsersId) {
      string SqlString = "SELECT * FROM Users WHERE
UsersId=" + UsersId.ToString();

      Users oneUsers = new Users();
      using (OleDbConnection conn = new
OleDbConnection(_ConnString)) {
        using (OleDbCommand cmd = new
OleDbCommand(SqlString, conn)) {
          conn.Open();
          using (OleDbDataReader reader =
cmd.ExecuteReader()) {
            while (reader.Read()) {
              oneUsers.UsersId =
Convert.ToInt32(reader["UsersId"]);
              oneUsers.FirstName =
reader["FirstName"].ToString();
              oneUsers.LastName =
reader["LastName"].ToString();
              oneUsers.UsersName =
reader["UsersName"].ToString();
              oneUsers.FIO = oneUsers.LastName + " " +
oneUsers.FirstName;
              oneUsers.UsersPassword =
_encryptData.Decrypt(reader["UsersPassword"].ToString()
);
              oneUsers.RoleId =
Convert.ToInt32(reader["RoleId"]);
              oneUsers.Description =
reader["Description"].ToString();
            }
          }
        }
        conn.Close();
      }
      return oneUsers;
    }

    public List<Users> GetAllUsersListForCBox() {
      string SqlString = "SELECT UsersId, UsersName,
UsersPassword FROM Users ORDER BY UsersName
ASC";
      List<Users> listAllUsers = new List<Users>();

      using (OleDbConnection conn = new
```

```
OleDbConnection(_ConnString)) {
       using (OleDbCommand cmd = new
OleDbCommand(SqlString, conn)) {
         conn.Open();
         using (OleDbDataReader reader =
cmd.ExecuteReader()) {
           while (reader.Read()) {
             Users oneUsers = new Users();
             oneUsers.UsersId =
Convert.ToInt32(reader["UsersId"].ToString());
             oneUsers.UsersName =
reader["UsersName"].ToString();
             oneUsers.UsersPassword =
_encryptData.Decrypt(reader["UsersPassword"].ToString()
);
             listAllUsers.Add(oneUsers);
           }
         }
         conn.Close();
       }
     }

     if (listAllUsers.Count == 0) {
       Users noUsers = new Users();
       noUsers.UsersId = 0;
       noUsers.Message =
NamesMy.NoDataNames.NoDataInUsers;
       listAllUsers.Add(noUsers);
     }
     return listAllUsers;
   }

   public List<Users>
SelectedUsersByUsersNameAndUsersPassword(string
UsersName, string UsersPassword) {
     string SqlString = "SELECT * FROM Users WHERE
UsersName='" + UsersName + "' AND UsersPassword='" +
_encryptData.Encrypt(UsersPassword) + "'";
     List<Users> UsersList = new List<Users>();

     using (OleDbConnection conn = new
OleDbConnection(_ConnString)) {
       using (OleDbCommand cmd = new
OleDbCommand(SqlString, conn)) {
         conn.Open();
         using (OleDbDataReader reader =
cmd.ExecuteReader()) {
           while (reader.Read()) {
             Users oneUsers = new Users();
             oneUsers.UsersId =
Convert.ToInt32(reader["UsersId"]);
             oneUsers.FirstName =
reader["FirstName"].ToString();
             oneUsers.LastName =
reader["LastName"].ToString();
             oneUsers.UsersName =
reader["UsersName"].ToString();
             oneUsers.FIO = oneUsers.LastName + " " +
oneUsers.FirstName;
             oneUsers.UsersPassword =
_encryptData.Decrypt(reader["UsersPassword"].ToString()
);
             oneUsers.RoleId =
Convert.ToInt32(reader["RoleId"]);
             oneUsers.Description =
reader["Description"].ToString();
             UsersList.Add(oneUsers);
           }
         }
       }
       conn.Close();
     }
     return UsersList;
   }

   public void UpdateUsers(string FirstName, string
LastName, string UsersName, string UsersPassword,
  int RoleId, string Description, int UsersId) {
     string SqlString = "UPDATE Users SET FirstName=?,
LastName=?, " +
  "UsersName=?, UsersPassword=?, RoleId=?,
Description=? WHERE UsersId=?";

     using (OleDbConnection conn = new
OleDbConnection(_ConnString)) {
       using (OleDbCommand cmd = new
OleDbCommand(SqlString, conn)) {
         cmd.CommandType = CommandType.Text;
         cmd.Parameters.AddWithValue("FirstName",
FirstName);
         cmd.Parameters.AddWithValue("LastName",
LastName);
         cmd.Parameters.AddWithValue("UsersName",
UsersName);
         cmd.Parameters.AddWithValue("UsersPassword",
_encryptData.Encrypt(UsersPassword));
         cmd.Parameters.AddWithValue("RoleId", RoleId);
         cmd.Parameters.AddWithValue("Description",
Description);
         cmd.Parameters.AddWithValue("UsersId", UsersId);
         conn.Open();
         cmd.ExecuteNonQuery();
         conn.Close();
       }
     }
   }

   public void DeleteUsersByUsersId(int UsersId) {
     string SqlString = "DELETE FROM Users WHERE
UsersId=" + UsersId.ToString();
     using (OleDbConnection conn = new
OleDbConnection(_ConnString)) {
       using (OleDbCommand cmd = new
OleDbCommand(SqlString, conn)) {
         conn.Open();
         cmd.ExecuteNonQuery();
         conn.Close();
       }
     }
   }

  }
}

public class Users {
  private int _Number;
  private int _UsersId;
  private string _FirstName;
  private string _LastName;
  private string _UsersName;
  private string _FIO;
  private string _UsersPassword;
  private int _RoleId;
  private string _RoleName;
  private string _Description;
  private string _Message;

  public Users() {
    _UsersId = 0;
    _FirstName = String.Empty;
```

```
    _LastName = String.Empty;
    _UsersName = String.Empty;
    _FIO = String.Empty;
    _UsersPassword = String.Empty;
    _RoleId = 0;
    _Description = String.Empty;
  }

  public int Number {
    set { _Number = value; }
    get { return _Number; }
  }
  public int UsersId {
    set { _UsersId = value; }
    get { return _UsersId; }
  }
  public string FirstName {
    set { _FirstName = value; }
    get { return _FirstName; }
  }
  public string LastName {
    set { _LastName = value; }
    get { return _LastName; }
  }
  public string FIO {
    set { _FIO = value; }
    get { return _FIO; }
  }
  public string UsersName {
    set { _UsersName = value; }
    get { return _UsersName; }
  }
  public string UsersPassword {
    set { _UsersPassword = value; }
    get { return _UsersPassword; }
  }
  public int RoleId {
    set { _RoleId = value; }
    get { return _RoleId; }
  }
  public string RoleName {
    set { _RoleName = value; }
    get { return _RoleName; }
  }
  public string Description {
    set { _Description = value; }
    get { return _Description; }
  }
  public string Message {
    set { _Message = value; }
    get { return _Message; }
  }
}
```

Лістинг 11. Код класу «DNSHeader»

```
using System.Net;
using System.Text;
using System;
using System.IO;
using System.Windows.Forms;
using System.Collections.Specialized;
using System.Collections;
using System.Collections.Generic;

namespace TCPIPAnalizer {
  public class DNSHeader {
    //DNS header fields
    private ushort _UsIdentification;       //Шістнадцять біт для ідентифікації
    private ushort _UsFlags;                //Шістнадцять біт для прапорів DNS
    private ushort _UsTotalQuestions;       //Шістнадцять бітів, що вказують кількість записів у списку запитів
    private ushort _UsTotalAnswerRRs;       //Шістнадцять бітів, що вказують кількість записів записів у списку записів ресурсу відповідей
    private ushort _UsTotalAuthorityRRs;    //Шістнадцять бітів, що вказують кількість записів записи в списку авторитетних ресурсних записів
    private ushort _UsTotalAdditionalRRs;   //Шістнадцять бітів, що вказують кількість записів записів у списку записів додаткових ресурсів кінцеві поля заголовка DNS

    public DNSHeader(byte[] byBuffer, int nReceived) {
      MemoryStream memoryStream = new MemoryStream(byBuffer, 0, nReceived);
      BinaryReader binaryReader = new BinaryReader(memoryStream);

      //Перші шістнадцять бітів призначені для ідентифікації
      _UsIdentification = (ushort)IPAddress.NetworkToHostOrder(binaryReader.ReadInt16());

      //Наступні шістнадцять містять прапори
      _UsFlags = (ushort)IPAddress.NetworkToHostOrder(binaryReader.ReadInt16());

      //Прочитати загальну кількість запитань у списку запитань
      _UsTotalQuestions = (ushort)IPAddress.NetworkToHostOrder(binaryReader.ReadInt16());

      //Прочитати загальну кількість відповідей у списку відповідей
      _UsTotalAnswerRRs = (ushort)IPAddress.NetworkToHostOrder(binaryReader.ReadInt16());

      //Прочитати загальну кількість записів у списку повноважень
      _UsTotalAuthorityRRs = (ushort)IPAddress.NetworkToHostOrder(binaryReader.ReadInt16());

      //Загальна кількість записів у списку додаткових ресурсів
      _UsTotalAdditionalRRs = (ushort)IPAddress.NetworkToHostOrder(binaryReader.ReadInt16());
    }

    public string Identification {
      get {
        return string.Format("0x{0:x2}", _UsIdentification);
      }
    }

    public string Flags {
      get {
        return string.Format("0x{0:x2}", _UsFlags);
      }
    }

    public string TotalQuestions {
      get {
```

```
      return _UsTotalQuestions.ToString();
    }
  }

  public string TotalAnswerRRs {
    get {
      return _UsTotalAnswerRRs.ToString();
    }
  }

  public string TotalAuthorityRRs {
    get {
      return _UsTotalAuthorityRRs.ToString();
    }
  }

  public string TotalAdditionalRRs {
    get {
      return _UsTotalAdditionalRRs.ToString();
    }
  }
 }
}
```