**Форма № 24**

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
**Національний університет «Запорізька політехніка»**

Факультет комп'ютерних наук і технологій
(повне найменування інституту, назва факультету)

Кафедра програмних засобів
(повне найменування кафедри )

# Пояснювальна записка

до дипломного проєкту (роботи)
бакалавр
(ступінь вищої освіти)

на тему ВИЯВЛЕННЯ МЕРЕЖЕВИХ АТАК ЗА ДОПОМОГОЮ ЕНТРОПІЇ
NETWORK ATTACK DETECTION USING ENTROPY


Виконав: студент(ка) 4 курсу, групи КНТ-219сп
Спеціальності 122 Комп'ютерні науки
(код і найменування спеціальності)
Освітня програма (спеціалізація)
Комп'ютерні науки
Федонюк Р.П.
(прізвище та ініціали)
Керівник Зайко Т.А.
(прізвище та ініціали)
Рецензент Коцур М.І.
(прізвище та ініціали)


2022

**Форма № 25**

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
**Національний університет «Запорізька політехніка»**
( повне найменування закладу вищої освіти)

Інститут, факультет ФКНТ
Кафедра програмних засобів
Ступінь вищої освіти бакалавр
Спеціальність 122 Комп'ютерні науки
(код і найменування)
Освітня програма (спеціалізація) Комп'ютерні науки
(назва освітньої програми (спеціалізації)

**ЗАТВЕРДЖУЮ**

**Завідувач кафедри ПЗ, д.т.н, проф.**
**_____С.О. Субботін**
“____” _____2022 року

# З А В Д А Н Н Я

## НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

Федонюк Роман Павлович
(прізвище, ім’я, по батькові)

1. Тема проєкту (роботи) Виявлення мережевих атак за допомогою ентропії. Network Attack Detection Using Entropy
керівник проєкту (роботи) Зайко Тетяна Анатоліївна, к.т.н., доцент,
( прізвище, ім’я, по батькові, науковий ступінь, вчене звання)
затверджені наказом закладу вищої освіти від “ 27 ” квітня 2022 року № 102
2. Строк подання студентом проєкту (роботи) 30 травня 2022 року
3. Вихідні дані до проєкту (роботи) рекомендована література, технічне завдання

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Аналіз предметної області. 2. Вибір і обґрунтування структури проєктованої системи. 3. Основні рішення щодо реалізації компонентів системи. 4. Керівництво програміста. 5. Керівництво користувача.

5. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень) Слайди презентації

6. Консультанти розділів проєкту (роботи)

| Розділ | Прізвище, ініціали та посада консультанта | Підпис, дата | |
|---|---|---|---|
| | | завдання видав | прийняв виконане завдання |
| 1-5 Основна частина | Зайко Т.В., доцент | | |
| Нормоконтроль | Липовець М.В., асистент | | |

7. Дата видачі завдання “ 04 ” квітня 2022 року.

## КАЛЕНДАРНИЙ ПЛАН

| № з/п | Назва етапів дипломного проєкту (роботи) | Строк виконання етапів проєкту ( роботи ) | Примітка |
|---|---|---|---|
| 1 | Постановка завдання роботи. | 1 тиждень | Завдання, ТЗ |
| 2 | Аналіз предметної області. | 1 тиждень | Розділ 1 |
| 3 | Вибір мови програмування та інших технологій | 2 тиждень | Розділ 2 |
| | розробки. | | |
| 4 | Розробка архітектури програми. | 2 тиждень | Розділ 3 |
| 5 | Розробка програми. | 3-4 тижні | Розділи 3, 4 |
| 6 | Тестування та експериментальне дослідження | 5 тиждень | Розділ 4 |
| | програмного забезпечення. | | |
| 7 | Оформлення пояснювальної записки та документів | 6 тиждень | Додатки |
| | до неї. | | |
| 8 | Нормоконтроль та рецензування. | 7 тиждень | |
| 9 | Захист роботи. | 8 тиждень | |

**Студент(ка)** _____ Федонюк Р.П.
( підпис ) (прізвище та ініціали)

**Керівник проєкту (роботи)** _____ Зайко Т.А.
( підпис ) (прізвище та ініціали)

# РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи бакалавра: 110 с., 32 рис., 2 табл., 3 дод., 14 джерел.

МЕРЕЖЕВІ ТЕХНОЛОГІЇ, МЕРЕЖЕВИЙ ТРАФІК, МЕРЕЖЕВІ ПРОТОКОЛИ, МЕРЕЖЕВА АТАКА, DDOS АТАКА, СИСТЕМА МОНІТОРИНГУ, ПАКЕТИ ДАНИХ.

Об’єкт дослідження – алгоритми та комп’ютерні програми виявлення мережевих атак.

Предмет дослідження – програмні засоби та алгоритми виявлення мережевих атак за допомогою ентропії мережі.

Мета роботи – створення програмної системи для автоматизації виявлення мережевих атак за допомогою ентропії.

Матеріали, методи та технічні засоби: об’єктно-орієнтоване програмування, маршрутизатор з підтримкою протоколу NetFlow, мова програмування C#, персональний комп’ютер під керуванням операційної системи Microsoft Windows 7 або вище, з процесором Intel Core 2 Duo або вище.

Результати. Розроблено програмне забезпечення для ОС Windows, яке в режимі реального часу аналізує мережевий трафік за допомогою налаштованого мережевого обладнання та виводить на екран графік шкідливої активності в мережі.

Висновки. Розроблено програмне забезпечення для виявлення мережевих атак за допомогою ентропії, що дозволяє своєчасно виявляти та припиняти шкідливу діяльність.

Галузь використання – аналіз мережі персональних користувачів, спеціалістами невеликих приватних компаній та підрозділами обслуговування серверів.

# ABSTRACT

Explanatory note to the diploma qualifying work of the bachelor: 110 pages, 32 figures, 2 tables, 3 appendixes, 14 sources.

NETWORK TECHNOLOGIES, NETWORK TRAFFIC, NETWORK PROTOCOLS, NETWORK ATTACK, DDOS ATTACK, MONITORING SYSTEM, DATA PACKAGES.

The object of research - algorithms and computer programs to detect network attacks.

The subject of research - software and algorithms for detecting network attacks using network entropy.

The purpose of the work is to create a software system to automate the detection of network attacks using entropy.

Materials, methods and techniques: object-oriented programming, router with NetFlow support, C # programming language, personal computer running Microsoft Windows 7 or higher, with Intel Core 2 Duo processor or higher.

Results. Developed software for Windows that real-time analyzes network traffic using configured network equipment and displays a graph of malicious network activity.

Conclusions. Software has been developed to detect network attacks using entropy, which allows you to detect and stop malicious activity in a timely manner.

Scope - analysis of the network of personal users, specialists of small private companies and server service departments.

# ЗМІСТ

С.

# ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

ARP – Address Resolution Protocol;

CA – Computer Associates;

DDoS – Denial-of-Service Attack;

DNS – Domain Name System;

DNSSEC – Domain Name System Security Extensions;

ICMP – Internet Control Message Protocol;

IP – Internet Protocol Address;

IPFIX – Internet Protocol Flow Information Export;

ISO – International Organization for Standardization;

ISS – International Service System;

NTA – NetFlow Traffic Analyzer;

OSI – Open Systems Interconnection Basic Reference Model;

PTT – Push-to-Talk;

TCP – Transmission Control Protocol;

UDP – User Datagram Protocol;

WAAS – Wide Area Application Services;

МС – мережеві системи;

ОС – операційна система;

ПВВ – пасивний віддалений вплив;

ПЗ – програмний засіб;

ПК – персональній комп’ютер;

РОС – розподілена обчислювальна система.

# ВСТУП

На сьогоднішній день обчислювальна техніка міцно закріпилась у житті багатьох людей. Стрімкий технічний прогрес, прагнення до автоматизації розв’язання усіх задач у життєвому циклі підприємств та компаній спонукають до використання все більшої кількості технологічних рішень їх розв’язання.

За останні роки зовнішні фактори стали одним із найбільших чинників збільшення використання електронно обчислювальної техніки. Процес глибокого переходу компаній у поле цифрових технології значно пришвидшився, що дуже сильно позначилось як на самих компаніях так і на цифрових системах що були застосовані при цьому переході. Одним із побічних явищ стало значне збільшення використання мережі інтернет, майже для усіх процесів. Таке навантаження на засоби мережевого зв’язку призвело до зростання ризику компрометації інформації.

Постійне зростання використання Інтернету в останні роки призвело до збільшення ризику розкриття інформації. Техніки вторгнення розвиваються і стають все більш і більш досконалими.

Таким чином, класичні системи виявлення вторгнень демонструють зниження продуктивності, коли мова йде про виявлення нових атак. Системи виявлення вторгнень класифікуються відповідно до сигнатурних, статистичних (аномальних) та гібридних методів аналізу [1].

Основною проблемою аномальних методів є складність конфігурації та велика кількість помилкових спрацьовувань у разі неправильно налаштованих правил. Ця робота присвячена системам виявлення вторгнень на основі аномалій, зокрема виявлення мережевих атак. Методи виявлення статистичних аномалій створюють статистичну модель, яка описує нормальний мережевий трафік і поведінку мережі, а також визначає будь-яку ненормальну поведінку, яка відхиляється від моделі [2].

В дипломній роботі було зосереджено увагу на системах на основі статистичного виявлення аномалій, а саме виявлення DDoS-атак.

# 1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

## 1.1 Актуальність проблеми

Останнім часом спостерігається збільшення поширених атак на глобальні комп’ютерні мережі. Багато з них пов’язані з доступністю або «розподіленою відмовою в обслуговуванні» (DDoS) і можуть вплинути на будь-які пов’язані з Інтернетом ресурси, включаючи DNS-сервери, електронну пошту. поштові служби, онлайн-проєкти та інтерфейси маршрутизації.

Сучасні МС розвиваються стрімко і набирають потужність. Компанії і організації, що використовують МС які налічують велику кількість комп'ютерів, першочерговим завданням вважають управління безліччю різноманітних захисних механізмів. Складність інфраструктури МС призводять до того, що при реалізації системи інформаційної безпеки за межами уваги адміністратора безпеки можуть залишитися багато вторгнень. Число інцидентів, пов’язаних з інформаційною безпекою, за даними провідних аналітичних агентств, постійно зростає [3].

Кібератака - спрямовані (навмисні) дії в кіберпросторі, які здійснюються за допомогою засобів електронних комунікацій (включаючи інформаційно-комунікаційні технології, програмні, програмно-апаратні засоби, інші технічні та технологічні засоби і обладнання) та спрямовані на досягнення однієї або сукупності таких цілей: порушення конфіденційності, цілісності, доступності електронних інформаційних ресурсів, що обробляються (передаються, зберігаються) в комунікаційних та/або технологічних системах, отримання несанкціонованого доступу до таких ресурсів; порушення безпеки, сталого, надійного та штатного режиму функціонування комунікаційних та/або технологічних систем; використання комунікаційної системи, її ресурсів та засобів електронних комунікацій для здійснення кібератак на інші об’єкти кіберзахисту [4].

В літературі існує чітка віра в те, що виявити атаки DDOS на ранніх стадіях можливо використовуючи аналіз даних мережевого трафіку з

перевагами в реальному часі та можливими відповідями є ефективним індикатором аномальної поведінки трафіку. Виявити аномалії мережі за допомогою відомих математичних методів обробки, аналізу та моделювання сигналів. Класичний статистичний аналіз, факторний та кластерний аналіз, спектральний аналіз, хвильовий аналіз, цифрова обробка сигналів, моделювання часових рядів, аналіз даних.

Практичні методи виявлення мережевих аномалій розробляють як університетські дослідницькі центри (Університет Огайо, Колумбійський університет, МДУ та ін.), так і великі корпорації (Cisco, CA, ISS, Symantec). У цих дослідженнях виявлення шкідливої діяльності в мережі засноване на обчисленні ентропії мережевого трафіку. У 1994 році В. Ліланд виявив, що DDOS-атака створює агрегований мережевий трафік що схожий на сам на себе [1]. У 2003 роцік Фестайн і Шнахенберг [2] виявили, що статистичні показники мережевого трафіку, такі як середнє вибіркове, вибіркова дисперсія та критерії консенсусу Пірсона $\chi^2$ або ентропія трафіку, теоретична міра інформації, можна розглядати як індикатори атаки. Останнє пов’язано з тим, що у випадку звичайної комп’ютерної мережі діапазон IP-атрибутів мережевих пакетів дуже широкий, а трафік непередбачуваний і хаотичний. При DDOS-атаці «хаотичність» в мережі поступово втрачається, оскільки вона «затоплена» багатьма подібними пакетами. Кількісно такі процеси традиційно описуються ентропією розподілу ймовірностей Шеннона. Тому DDOS-атаки неминуче мають зменшити ентропію мережевого трафіку.

## 1.2 Віддалена мережева атака

Віддалена мережева атака – це спроба вплинути на віддалений комп’ютер за допомогою програмних методів. Як правило, метою мережевих атак є порушення конфіденційності даних, тобто крадіжка інформації.

Набори TCP/IP використовуються для організації зв'язку в гетерогенних мережевих середовищах. Це забезпечує сумісність між різними

типами комп’ютерів. Цей набір протоколів TCP/IP популярний завдяки сумісності та доступу до ресурсів світової мережі Інтернет. Поширення протоколу TCP/IP показало його недоліки. Розподілені системи вразливі до віддалених атак, оскільки їх компоненти зазвичай використовують відкриті канали даних, що дозволяє зловмисникам пасивно прослуховувати інформацію, що надсилається, а також змінювати вихідний трафік.

Складність виявлення дистанційних атак і відносно легке їх виконання (через надмірну функціональність сучасних систем) виводять цей вид протиправної діяльності на одне з перших місць за небезпекою та заважає вчасно реагувати на загрозу, як наслідок, збільшує шанси на успішне виконання порушення.

### 1.2.1 Класифікація мережевих атак

За характером впливу мережеві атаки поділяють на:

- пасивні впливи на розподілені комп’ютерні системи – це ті, які безпосередньо не впливають на поведінку системи, але водночас можуть порушувати її політику безпеки. Пасивні дистанційні ефекти важко виявити, оскільки вони не впливають безпосередньо на продуктивність розподілених комп’ютерних систем. Можливим прикладом типового пасивного дистанційного ефекту є прослуховування каналу зв'язку в мережі;
- активний вплив на РОС - ефекти, які безпосередньо заважають роботі системи (перебої, зміни в конфігурації РОС тощо). Це порушує політику безпеки, прийняту системою. Активними ефектами є майже всі види дистанційних атак. Це пояснюється тим, що активна складова є головною частиною подібних інформаційних атак. Очевидна різниця між активними та пасивними ефектами – це основна можливість їх виявлення, оскільки впровадження системи призводить до деяких змін. Пасивні дії не залишають сліду ,бо коли зловмисник переглядає інше повідомлення в системі, насправді нічого не змінюється.

Класифікація мережевих атак за цілями впливу виділяються три типи:

- атаки розвідки;
- атаки отримання доступу;
- атаки відмови в обслуговуванні [ТВС].

Ці класифікатори, по суті, є прямим описом трьох основних типів загроз: відмови в наданні послуг, розкриття та невідповідності. Основна мета майже всіх атак – отримати несанкціонований доступ до інформації. Існує два основних способи отримання інформації: спотворення та перехоплення. Можливість перехопити інформацію означає доступ без можливості змінити інформацію. Перехоплення інформації порушує її конфіденційність. Прикладом перехоплення інформації є прослуховування каналів мережі. У цьому випадку є незаконний доступ до інформації і неможливо її змінити. Зрозуміло, що порушення конфіденційності відповідають характеристиці пасивних впливів.

Під здатністю змінювати інформацію слід розуміти або здатність повністю контролювати потік інформації між об’єктами системи, або можливість надсилати різні повідомлення від імені інших об’єктів. Тому зрозуміло, що зміна інформації порушує її цілісність. Наслідки такої деструктивної інформаційної атаки є класичним прикладом активного впливу. Прикладом віддаленої атаки з метою порушення цілісності інформації є віддалена атака на «підроблений об’єкт у РОС».

Якщо є зворотний зв’язок від атакованого об’єкта:

- зі зворотним зв'язком;
- немає зворотного зв'язку (однонаправлена атака).

Зловмисник надсилає кілька запитів атакованому об’єкту та очікує відповіді. В результаті між зловмисником і метою атаки створюється зворотний зв’язок, і зловмисник може належним чином реагувати на будь-які зміни цілі. У цьому полягає суть дистанційної атаки яка здійснюється при отриманні зворотного зв'язку від об'єкта атаки. Такі атаки найчастіше трапляються на РОС.

Атаки без зворотного зв'язку характеризуються тим, що вони не повинні реагувати на зміни в об'єкті, який атакується. Такі атаки зазвичай здійснюються шляхом відправки єдиного запиту до цілі атаки. Відповіді на ці запитання зловмисникам не потрібні. Такі інформаційні атаки також відомі як односторонні. Прикладом односторонньої атаки є типова DoS-атака.

За умовою початку здійснення впливу.

Як і в інших випадках, дистанційний вплив можна розпочати лише за певних умов. У РОС існують три типи таких умовних атак:

- атака за запитом від атакованого ПК;
- атака на об'єкт за настання очікуваної події;
- атака без попередніх умов.

Якщо потенційна мета атаки надсилає запит певного типу, починається атака на стороні зловмисника. Таку атаку можна назвати атакою на вимогу від цілі атаки. Цей тип інформаційної атаки найчастіше зустрічається в розподілених обчислювальних системах. Прикладами таких онлайн-запитів є запити ARP і DNS, а Novell NetWare — запит SAP.

Тип атаки, коли в атакованому об'єкті відбувається очікувана подія. Зловмисник постійно контролює стан операційної системи віддаленого об'єкта атаки і починає впливати на нього, коли в цій системі відбуваються певні події. Ініціатором атаки є сам об'єкт, на який нападають. Прикладом такої події є те, що наприклад Novell NetWare припиняє сеанс користувача із сервером без виконання команди LOGOUT.

Безумовні атаки відбуваються негайно, незалежно від стану операційної системи чи мети атаки. Тому в даному випадку зловмисник є ініціатором атаки.

Атаки, які перешкоджають нормальній роботі системи, служать іншим цілям, і не очікується, що зловмисник отримає несанкціонований доступ до даних. Його мета – відключити ОС атакуваного об’єкта та запобігти доступу інших системних об’єктів до ресурсів об’єкта. Прикладом такої атаки є DoS-атака.

По місцерозташуванню суб'єкта атаки до атакованого об'єкта:

- міжсегментне;
- внутрішньосегментне.

Джерело атаки (суб'єкт атаки) — програма, оператор або інфікований пристрій, що виконує атаку і здійснює безпосередній шкідливий вплив.

Хост (host) — комп'ютер, що є елементом мережі.

Роутер (маршрутизатор) — пристрій, який забезпечує маршрутизацію пакетів у мережі.

Підмережі — це групи комп’ютерів, які є частиною глобальної мережі та характеризуються маршрутизаторами, які призначають один і той же номер підмережі хост-комп’ютерам. Можна також сказати, що підмережі є логічною комбінацією хостів, які використовують маршрутизатор. Хост-комп'ютери в одній підмережі можуть безпосередньо спілкуватися один з одним без використання маршрутизатора.

Сегмент мережі — об'єднання хостів на фізичному рівні.

Взаємне положення об'єкта атаки дуже важливе для дистанційних атак. Тобто вони знаходяться в різних або однакових сегментах. При виконанні внутрішньо-сегментованої атаки суб'єкт атаки і ціль знаходяться в одному сегменті. Під час міжсегментної атаки ціль і мета атаки знаходяться на різних сегментах мережі. Ця класифікаційна ознака дозволяє визначити так звану «дистанцію» атаки.

На практиці, більшість внутрішньо-сегментних атак набагато легше реалізувати, ніж міжсегментні атаки. Майте на увазі, що віддалені атаки між сегментами набагато небезпечніші, ніж всередині сегментів. Це пов'язано з тим, що в разі міжсегментної атаки об'єкт і безпосередній нападаючий можуть знаходитися на відстані тисячі кілометрів один від одного, що може істотно ускладнити способи відбиття атаки.

За впливом на якому здійснюється атака відповідно до рівня моделі ISO/OSI:

- фізичний;

- канальний;
- мережевий;
- транспортний;
- сеансовий;
- представницький;
- прикладний.

Міжнародна організація ISO прийняла ISO 7498, який описує відносини між відкритими системами (OSI), включаючи розподілені обчислювальні системи. Кожен протокол мережевого обміну і кожна мережева програма можуть бути розроблені будь-яким чином, використовуючи еталонну 7-рівневу модель OSI. Ця багатокрокова проекція дозволяє вам описати модель OSI, яка використовується в мережевих протоколах або функціях програмування. Remote Attack — це мережева програма, яку має сенс розглянути під час проєктування на стандартну модель ISO / OSI.

### 1.2.2 Основні типи мережевих атак

Фрагментація даних.

При відправці пакета даних до мережевого протоколу IP цей пакет можна розділити на кілька фрагментів. Потім, коли він досягає місця призначення, пакет відновлюється з цих фрагментів. Зловмисник може почати надсилати багато фрагментів, які переповнять буфер програмного забезпечення хоста і, можливо, навіть пошкодять систему.

Атака Ping Flooding.

Програма ping надсилає пакет ICMP типу ICHO REQUEST і встановлює час та його ідентифікатор. Ядро машини-отримувача відповідає на подібні запити за допомогою пакета ICMPECHOREPLY. При отриманні ping вказує швидкість пакету. У стандартному режимі пакети відправляються через певні інтервали, що не дуже ускладнює роботу мережі. Проте в «агресивному»

режимі потік пакетів запиту/відповіді ICMP може перевантажити невелику чергу і втратити можливість надсилати корисну інформацію.

Нестандартні протоколи, інкапсульовані в IP.

IP-пакети містять поля, які визначають протокол для інкапсульованих пакетів (TCP, UDP, ICMP). Зловмисники можуть використовувати нестандартні значення в цьому полі для надсилання даних, які не фіксуються стандартними інструментами керування інформаційними потоками.

Атака Smurf.

Зловмисник надсилає запит до широкомовної мережі ICMP від імені комп’ютера жертви. В результаті комп'ютер, який отримує такий широкомовний пакет, збігається з комп'ютером жертви, значно зменшуючи пропускну здатність каналу зв'язку, а в деяких випадках повністю ізолюючи атакувану мережу.

Атаки смурфів дуже ефективні і поширені.

Контрзаходи: щоб дізнатися про цю атаку, вам потрібно проаналізувати навантаження каналу та визначити причину зменшення пропускної здатності.

Атака DNS Spoofing.

Ця атака викликає вимушену відповідь між IP-адресою та іменем домену в кеші DNS сервера. В результаті таких атак усі користувачі DNS-серверів отримують невірну інформацію про доменні імена та IP-адреси. Ця атака включає багато DNS-пакетів з однаковим доменним іменем. Це тому, що потрібно вибрати деякі параметри обміну DNS.

Рішення: щоб виявити таку атаку, ви повинні проаналізувати вміст потоку DNS або використовувати DNSSEC.

Атака IP Spoofing.

Багато атак в Інтернеті передбачають зміну вихідної IP-адреси. Такі атаки включають втручання в системний журнал, який надсилає повідомлення на комп’ютер жертви від імені іншого комп’ютера у внутрішній мережі. Syslog використовується для керування системними журналами і може надсилати

помилкові повідомлення на комп'ютер жертви, щоб нав'язати інформацію або змінити сліди несанкціонованого доступу.

Рішення: Атака, яка передбачає зміну IP-адреси під час моніторингу отримання пакету з такою ж адресою джерела інтерфейсу, можна виявити на одному з інтерфейсів внутрішньої мережі або під час моніторингу отримання пакету з IP-адресою на зовнішньому інтерфейсі.

Нав'язування пакетів.

Зловмисник надсилає в мережу пакет з недійсною адресою повернення. Ця атака дозволяє зловмиснику перемикати з’єднання між іншими комп’ютерами на свій комп’ютер. У цьому випадку права доступу зловмисника дорівнюють правам користувача при перемиканні підключення до сервера на комп’ютер зловмисника.

Sniffing — прослуховування каналу.

Можливо виконати в сегменті локальної мережі.

Майже всі мережеві адаптери підтримують можливість перехоплення пакетів, надісланих через загальний канал локальної мережі. Робоча станція може отримувати пакети для інших комп'ютерів у тому самому сегменті мережі. Таким чином, зловмисник матиме доступ до всього обміну інформацією в сегменті мережі. Щоб ця атака була успішною, комп’ютер зловмисника повинен знаходитися в тому самому сегменті локальної мережі, що і комп’ютер атакує.

Перехоплення пакетів на маршрутизаторі.

Мережеве програмне забезпечення маршрутизатора може отримати доступ до всіх мережевих пакетів, надісланих через цей маршрутизатор, тому його можна перехопити. Для здійснення цієї атаки зловмисник повинен мати привілейований доступ принаймні до одного мережевого маршрутизатора. Зазвичай через маршрутизатор надсилаються багато пакетів, що робить їх повне перехоплення майже неможливим.

Однак окремі пакети можуть бути перехоплені та збережені для зловмисника для подальшого аналізу. Найефективніші пакети FTP з паролями користувачів та електронною поштою.

Нав'язування хосту помилкового маршруту з допомогою протоколу ICMP.

В Інтернеті існує спеціальний протокол керування повідомленнями Інтернету (ICMP), одна з його функцій — сповіщати хоста про зміни в існуючому маршрутизаторі. Це контрольне повідомлення називається перенаправленням. Будь-який комп’ютер у сегменті мережі може відправити помилкове повідомлення про пересилання на атакуваний комп’ютер від імені маршрутизатора.

В результаті хост замінює існуючу таблицю маршрутизації, і в майбутньому весь мережевий трафік на цьому хості буде перенаправлено, наприклад, на хост, який надіслав неправильне повідомлення пересилання. В результаті неправильний маршрут може бути агресивно нав’язаний одному сегменту Інтернету.

WinNuke.

На додаток до звичайних даних, що надсилаються через TCP-з'єднання, стандарт також визначає передачу аварійних (поза смугових) даних. На рівні формату пакету TCP це вказується ненульовим аварійним покажчиком. Більшість комп’ютерів під керуванням Windows мають мережевий протокол NetBIOS, який за потреби використовує три IP-порти (137, 138, 139). Якщо підключитися до пристрою Windows з портом 139 і надіслати на нього кілька байтів даних OutOfBand, програма NetBIOS зупинить або перезапустить ПК, тому що не знає, що робити з цими даними. Якщо ви використовуєте Windows 95, це зазвичай виглядає як помилка в драйвері TCP/IP і синє текстове повідомлення про те, що мережею не можна керувати до перезавантаження ОС. NT4.0 без пакета оновлень буде перезапущено, а NT4.0 з пакетом оновлень 2 матиме синій екран. Судячи з даних мережі, Windows NT 3.51 і Windows 3.11 для робочих груп також стикаються з такими атаками.

Під час надсилання даних на порт 139 NT 4.0 перезапускається або «синій екран» вимикається, коли інсталюється пакет оновлень 2. Для Windows NT WorkStation це значно сповільнює та ефективно вимикає Windows NT Server.

Підміна довіреного хоста.

Якщо цей тип віддаленої атаки буде успішним, зловмисник може увійти на сервер від імені довіреного хоста. Довірений хост - станція, яка законно підключена до сервера. Здійснення цього типу атаки зазвичай передбачає надсилання пакетів зі станції зловмисника від імені довіреної станції, якою керує зловмисник.

## 1.3 Огляд систем і програмних продуктів для аналізу та виявлення мережевих атак

Виявлення мережевих атак є дуже складною проблемою, і вона все ще досліджується. Особливо це стосується корпоративних мереж, які часто стикаються з проблемами безпеки та конфіденційності мережі. У той же час вирішення проблем безпеки мережі може допомогти спростити життя багатьох компаній.

### 1.3.1 Програмний продукт " Plixer Scrutinizer "

Програмне рішення Plixer Scrutinizer — це повна система реагування на інциденти, яку можна використовувати як для аналізу мережевого трафіку, так і для звітування про інциденти безпеки. Plixer Scrutinizer може використовувати NetFlow, J-Flow, NetStream і IPFIX для збору та аналізу даних з різних типів трафіку. Одним словом, це рішення чудово працює на різних пристроях різних виробників.

Plixer Scrutinizer забезпечує видимість як у фізичному, так і у віртуальному середовищах. Він також має високу масштабованість, оскільки

він швидкий, розширений звіт, підтримує командну роботу та побудований на розподіленій архітектурі.

Головне вікно наведено на рисунку 1.1.

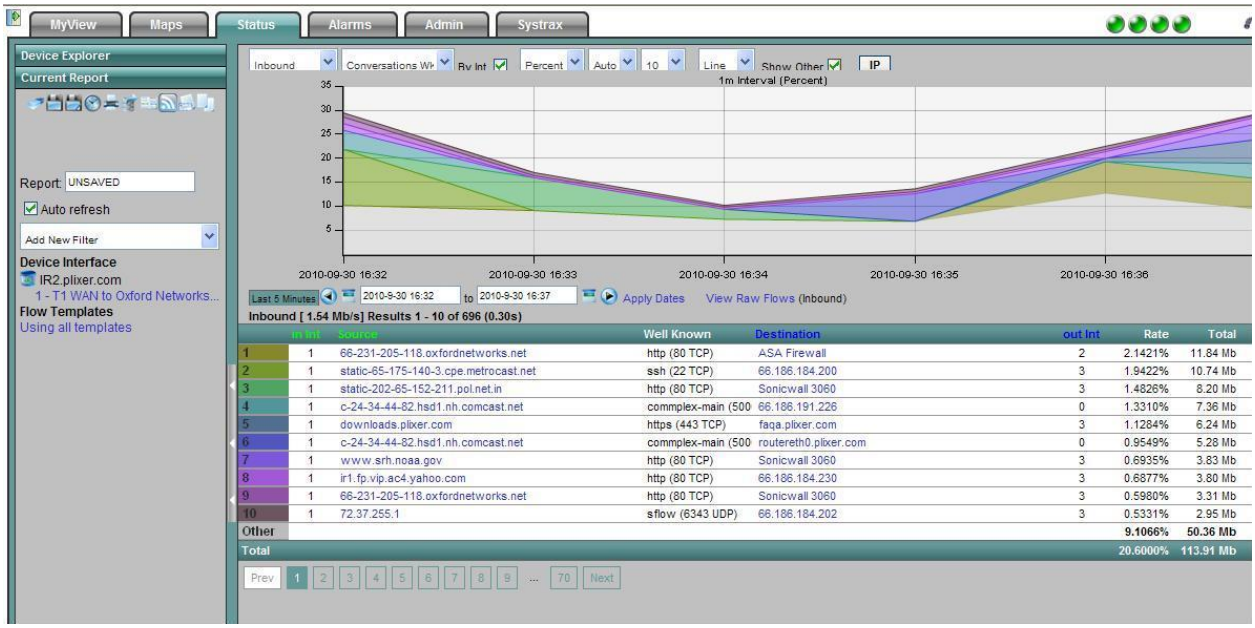


Рисунок 1.1 – Головне вікно Plixer Scrutinizer

### 1.3.2 Програмне забезпечення "PRTG Network Monitor

PRTG Network Monitor — це універсальне програмне рішення для моніторингу мережі, яке можна аналізувати за допомогою різних версій стандарту NetFlow (v5 і v9), відкритого універсального стандарту IPFIX та інших технологій обліку мережевого трафіку, таких як sFlow і J-Flow.

Приклад головного вікна наведено на рисунку 1.2.

Виконує такі функції:

- моніторинг діяльності;
- контроль обмеження можливостей;
- контролюйте продуктивність серверів ЦП, програм, вебсайтів і хмарних сервісів.

Рисунок 1.2 – Головне вікно PRTG Network Monitor

### 1.3.3 Програма «Solarwinds NetFlow Traffic Analyzer»

Solarwinds NetFlow Traffic Analyzer (NTA) — це програмне забезпечення для аналізу мережевого трафіку та моніторингу пропускної здатності, яке підтримує різноманітні протоколи, такі як NetFlow, J-Flow, sFlow, IPFIX та NetStream.

NetFlow Traffic Analyzer збирає дані про трафік, адаптує їх до використовуваного формату та відображає інформацію користувачам через вебінтерфейс для моніторингу мережевого трафіку. Ви можете аналізувати мережевий трафік за певний період часу (хвилини, години, дні або місяці).

Приклад головного вікна програми наведено на рисунку 1.3.

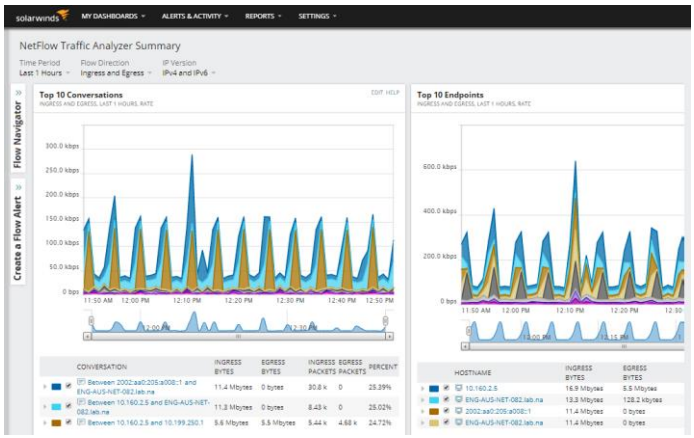


Рисунок 1.3 – Приклад головного вікна програми «Solarwinds NetFlow Traffic Analyzer»

### 1.3.4 Програмне забезпечення «ManageEngine NetFlow Analyzer»

ManageEngine — це потужний програмний продукт з усіма функціями для збору та аналізу даних мережевого трафіку. ManageEngine NetFlow Analyzer підтримує різноманітні потокові інформаційні технології, такі як NetFlow, J-Flow і NetStream, і призначений для аналізу мережевого трафіку в реальному часі та моніторингу пропускної здатності.

Приклад головного вікна програми наведено на рисунку 1.4.

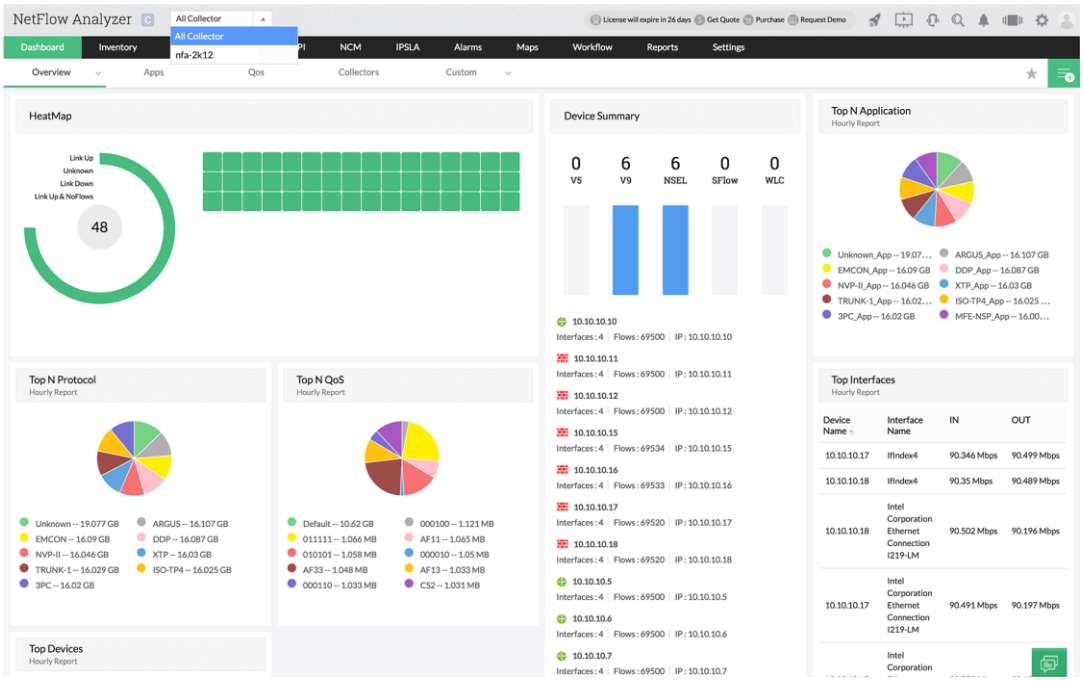


Рисунок 1.4 – Приклад головного вікна програми «ManageEngine NetFlow Analyzer»

### 1.3.5 Програмне забезпечення «nProbe and ntopng»

Ntopng — вебінструмент з відкритим кодом для моніторингу та аналізу мережевого трафіку. Ця програма була розроблена для використання на мобільних пристроях під керуванням однієї з найпопулярніших платформ, таких як Unix, MacOS та Windows. Дані в переданому пакеті отримуються

через зашифрований інтерфейс. Програма аналізує їх і надає різноманітну корисну інформацію про можливості мережі.

Приклад головного вікна програми наведено на рисунку 1.5.

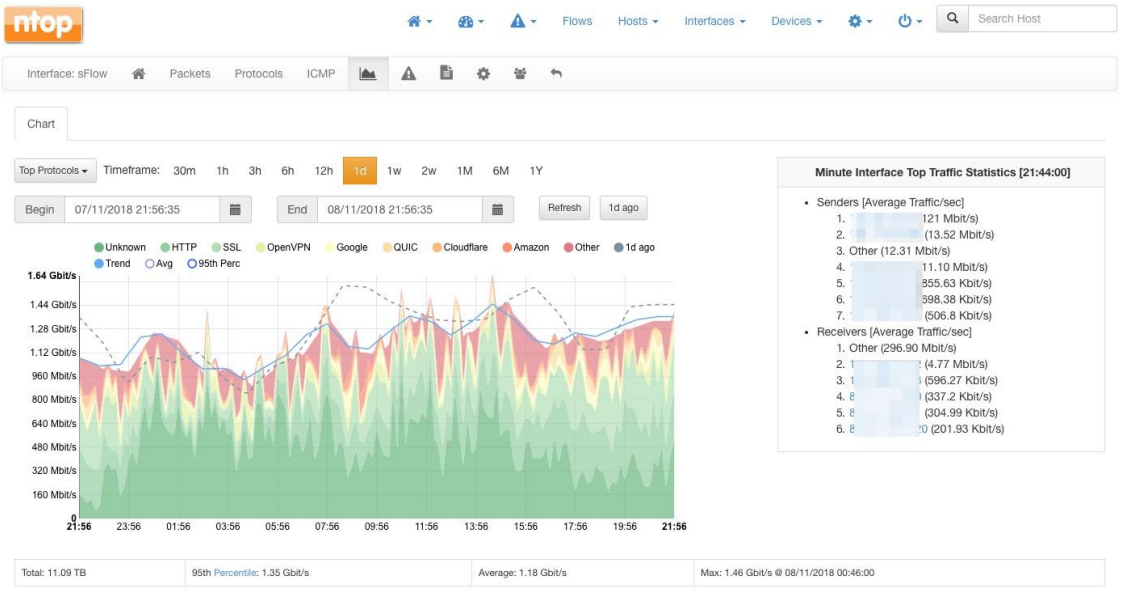


Рисунок 1.5 – Приклад головної сторінки «nProbe and ntopng»

Наприклад, за допомогою ntopng ви можете:

- сортувати інформацію про мережевий трафік за багатьма параметрами, такими як IP-адреса, порт, протокол, пропускна здатність тощо;
- моніторинг у режимі реального часу найактивніших комп'ютерів і програм, які потребують найбільшої пропускної здатності;
- відстежуйте кількість надісланих байтів і кількість надісланих, отриманих і втрачених пакетів;
- збережіть історію мережевого трафіку для подальшого аналізу;
- визначте географічні розташування та створіть діаграми підключення хостів на карті розташування.

Ntopng може підключатися до колектора NetFlow / IPFIX, nProbe. Ролі в цьому тандемі такі: nProbe збирає дані про трафік і надсилає цю інформацію до ntopng. Потім Ntopng аналізує його та представляє у зручному форматі.

Ntopng — безкоштовна версія (є більш розширені платні версії продукту), але для використання nProbe потрібна ліцензія (крім некомерційних або навчальних закладів). Існує дві версії nProbe, Standard і Pro з плагінами.

## 1.4 Висновки за розділом

Мета даного проєкту – створення простого та швидкого програмного забезпечення для можливості виявлення мережевих атак. Завдяки цьому програмному продукту планується зменшити ризики компрометування інформації, виведення з ладу компонентів мережі та ефективно виявляти мережеві атаки для швидкого реагування на загрозу.

Алгоритми та програмне забезпечення є основою для розробки автоматизованих систем для аналізу мережі інтернет і виявлення мережевих атак.

На основі огляду існуючих систем і програмного забезпечення для автоматизованого аналізу ми визначили ключові функції, необхідні для програми для аналізу мережевих атак.

- знаходження аномалій мережі на графіку мережевого трафіку;
- розрахунок основних ентропійних показників;
- здатність аналізувати мережу за допомогою атрибутів мережевого трафіку;
- перегляд графіку аналізу.

# 2 ВИБІР І ОБГРУНТУВАННЯ СТРУКТУРИ ПРОЄКТОВАНОЇ СИСТЕМИ

## 2.1 Дослідження методів та алгоритмів розпізнавання мережевих атак

Загальноприйнята класифікація систем виявлення атак за способами виявлення атак включає системи виявлення аномалій та системи виявлення зловживань. Однією з класичних праць у галузі виявлення зловживань є робота Кумар С., «Модель відповідності шаблону для виявлення неправомірного вторгнення», матеріали 17-ї національної конференції з комп'ютерної безпеки 1994 рік. На рисунку 2.1 представлена схема виявлення аномалій мережі на основі показників мережевого трафіку [5].


Рисунок 2.1 – Схема виявлення аномалій мережі

Загальний алгоритм виявлення аномалій мережі може бути описаний наступним чином. Даними для аналізу є мережевий трафік, представлений як набір мережевих пакетів, у випадку фрагментованих лише на рівні IP. Зібрані сирі дані надалі будуть джерелом для формування необхідної інформації для подальшого аналізу. Так, отримані дані можуть бути агреговані за певний часовий інтервал та нормалізовані з метою завдання ознакових атрибутів

загального виду, які будуть потрібні при побудові поточного профілю активності. Створений набір ознак порівнюється з набором характеристик нормальної діяльності об'єкта (користувача або системи) - шаблоном нормальної поведінки. Якщо спостерігається істотна розбіжність порівнюваних параметрів, то фіксується аномалія мережі. В іншому випадку відбувається уточнення шаблону нормальної поведінки за допомогою зміни параметрів його налаштування з урахуванням поточного профілю активності, що спостерігається [5].

Описаний вище алгоритм може включати кілька варіантів виконання реалізації підсистеми перевірки на відповідність шаблону нормальної поведінки. Найпростішим є процедура порівняння з порогової величиною, коли накопичені результати, що описують поточну мережну активність, порівнюються з експертно заданою числової планкою. У цьому вся підході випадок перевищення значень аналізованих параметрів зазначеної межі є ознакою мережевої аномалії [5].

Класифікація методів виявлення атак, запропонована у цій роботі, схематично показано рисунку 2.2. Для її побудови було проведено аналіз великого переліку робіт, який дозволив уточнити запропоновані у них таксономії та схеми відомих методів виявлення мережевих атак [5].

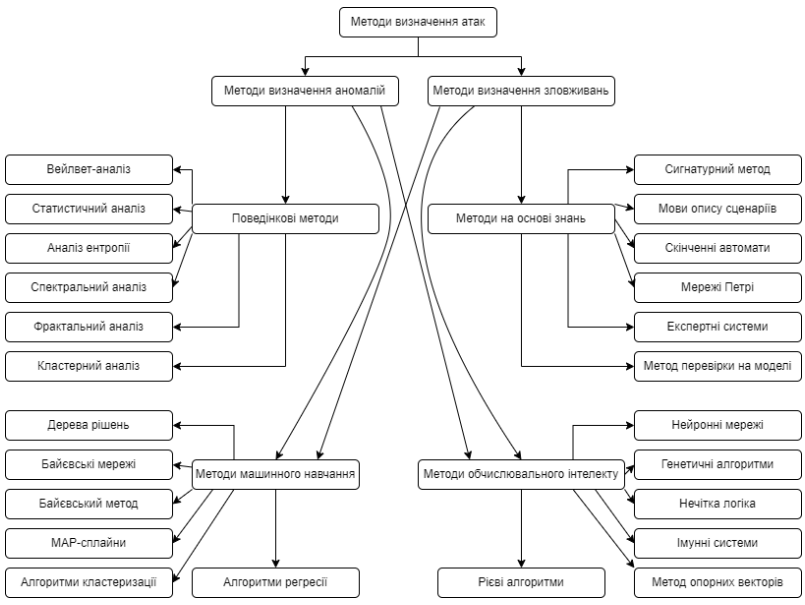


Рисунок 2.2 – Схема відомих методів виявлення мережевих атак

До методів поведінки належать алгоритми, які використовують інформацію про нормальну поведінку системи. Цю інформацію можна порівняти з отриманою під час спостереження.

Цей метод має такі недоліки:

- помилкові тривоги, пов’язані зі складністю опису моделей поведінки користувачів;
- багатокрокові попередні налаштування можуть зайняти тижні або місяці.

До цієї групи відноситься і метод мережевої ентропії, заснований на ентропії Шеннона.

Методи, засновані на знаннях, виконують дії, які виявляють атаки на основі відомих фактів і представлені у форматі на основі знань, який включає опис відомих атак. У більшості випадків ці методи не виявляють атаки, які не мають бази знань. Це найсерйозніший недолік. Також ефективність цих методів безпосередньо залежить від точності та актуальності пояснення ознак нападу.

Алгоритми машинного навчання для вирішення проблем виявлення вторгнень автоматично створюють моделі на основі навчального набору даних, представленого набором ознак (якісним або кількісним). Алгоритми машинного навчання, так як і методи обчислювального інтелекту, використовують як навчальні приклади, так і шаблони звичайної та нелегітимної поведінки мережі. Його можна використовувати для виявлення як зловживання, так і аномалій.

Методи обчислювального інтелекту поєднують елементи навчання, адаптації, еволюції та нечіткої логіки для прийняття розумних рішень. Такі алгоритми можуть реалізувати функції самонавчання в системах безпеки та виявляти загрози на льоту.

Методи машинного навчання та обчислювального інтелекту забезпечують високий рівень гарантії виявлення мережевих атак. Критерії оцінки методів виявлення вторгнень.

Для оцінки методів машинного навчання та обчислювального інтелекту були обрані наступні загальні критерії.

- точність;
- чесність;
- F-значення.

Точність визначається відношенням правильно виявлених аномалій до суми правильно виявлених аномалій і помилок другого типу. Точність (Precision) розраховується за формулою (2.1):

$$Precision = \frac{(TP)}{(TP+FP)}, \tag{2.1}$$

де TP - дійсно позитивним рішенням (кількість правильно виявлених аномалій);

FP - помилка другого типу (хибнопозитивна робота, тобто хибне спрацювання на нешкідливий об’єкт).

Recal – це відношення аномалії, правильно виявленої алгоритмом, до суми аномалій, виявлених правильно, і помилки 1 типу. За визначенням він розраховується за такою формулою (2.2):

$$Recal = \frac{(TP)}{(TP+FN)}, \tag{2.2}$$

де FN - помилка першого типу (помилково негативна операція, тобто якщо дійсно шкідливий об’єкт вважається нешкідливим).

Міра F є добутком оцінок повноти та точності, отриманих із суми оцінок повноти та точності, та визначається за формулою (2.3):

$$F = 2 \times \frac{(Precision \times Recal)}{(Precision+Recal)}. \tag{2.3}$$

Критерії оцінки методів.

Для проведення оцінки методів машинного навчання та обчислювального інтелекту необхідно використовувати уніфіковані набори, що містять як зразки нормального трафіку, і ознаки мережевих атак. У цьому дослідженні набори ознак беруться зі стиснутої бази [6].

NSL-KDD, кожен запис якої є образ мережевого з'єднання. Набір даних для навчання містить 125973 зразка, набір тестових даних містить 22544 зразка. Навчання класифікаторів здійснюється на вибірці KDDTrain, тестування – на вибірці KDDTest, дані в яких не перетинаються [6].

Для реалізації методів МО та BI використовувалися інтерпретована мова програмування Python 3.7 та бібліотеки для машинного навчання та штучного інтелекту TesorFlow, ScikitLearn та PyBrain [6].

Оцінки точності, Rical і F-мажор були розраховані як середнє значення результатів тесту для таких методів виявлення атак: smulf, neptune, guess_passwd, buffer_overflow, satan, warezmaster, back, nmap, pod, ipsweep, teardrop, portsweep, land [6].

Оцінка методів машинного навчання.

Методи машинного навчання включають дерева рішень, байєсівські мережі та алгоритми кластеризації.

Дерево рішень використовує «корінь» і «лист» для класифікації нових елементів і отримання відповідного класу. Цей шлях являє собою набір правил класифікації на основі значення атрибутів об’єкта.

Байєсівська мережа є моделлю кодування для стохастичних зв'язків між подіями, що розглядаються. Ця модель забезпечує механізм розрахунку умовних ймовірностей.

Існує багато типів алгоритмів кластеризації. Найпоширенішим алгоритмом є метод k-ближнього сусіда (k-NN).

Метод K-ближнього сусіда — це метричний алгоритм автоматичної класифікації об’єктів. Основний принцип методу найближчого сусіда полягає в тому, що об’єкт належить до найпоширенішого класу серед своїх сусідів.

Сусіди отримують з набору об’єктів, клас яких уже відомий, і більшість їхніх класів розраховується з базового значення k цього методу. Кожен об’єкт має скінченну кількість атрибутів (розмірів). Передбачається, що у вас є певний набір об’єктів з поточною класифікацією. Приклад цієї класифікації зображено на рисунку 2.3.

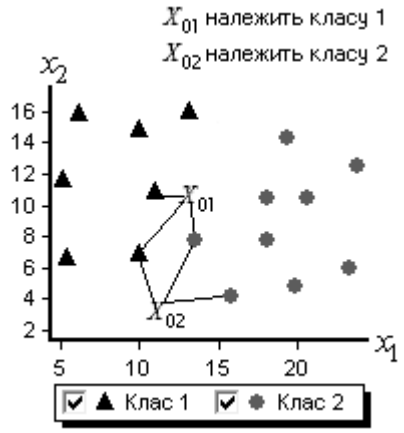


Рисунок 2.3 – Приклад класифікації

Проблема розмірності.

Цей класифікатор передбачає, що подібні точки мають подібні мітки. На жаль, у великому просторі точки, вибрані з розподілу ймовірностей, рідко наближаються одна до одної.

Цей алгоритм класифікації заснований на принципі віднесення найбільш поширених об’єктів серед сусідів класу.

Опис методів обчислюваного інтелекту

Методи обчислювального інтелекту включають ройовий алгоритм, методи опорних векторів, нейронні мережі, штучні імунні системи, генетичні алгоритми, інформаційні та ентропійні методи.

Алгоритми рою засновані на моделюванні поведінки та взаємодії біологічних особин у «колоніях» або «стадах».

Машина опорних векторів (SVM), використовуючи лінійні комбінації кількох опорних векторів із навчального прикладу, створює оптимальну гіперплощину. Приклад алгоритму наведено на рисунку 2.4.

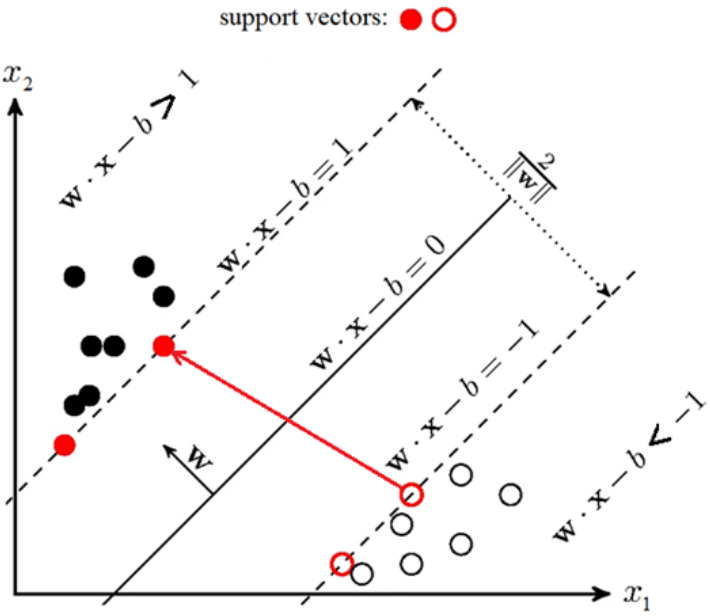


Рисунок 2.4 – Приклад алгоритму опорних векторів

Залежно від розташування елемента стосовно площині приймається рішення про належність елемента до того чи іншого класу [7].

В імунних системах за основу взято модель імунної системи людини. Такі системи включають механізми створення, навчання, імунних детекторів, знищення детекторів, що викликають помилкові спрацьовування, і реакцію у відповідь на чужорідні патогени [7].

Генетичний алгоритм базується на моделі біологічних принципів природного відбору.

Ці алгоритми використовуються в задачах оптимізації. Спочатку створюється початкова популяція особин, потім за допомогою операторів схрещування та мутації створюються нове потомство. Значення придатності, що описують ефективність вирішення проблеми, розраховуються для кожного нового потомства. Приклад цього алгоритму наведено на рисунку 2.5.

Нейронні мережі засновані на наборі пов’язаних вузлів, які називаються штучними нейронами. Кожне з’єднання штучних нейронів (подібно до синапсів) може посилати сигнали один одному. Після отримання сигналу штучний нейрон може обробити сигнал і відправити його на підключений до нього штучний нейрон.

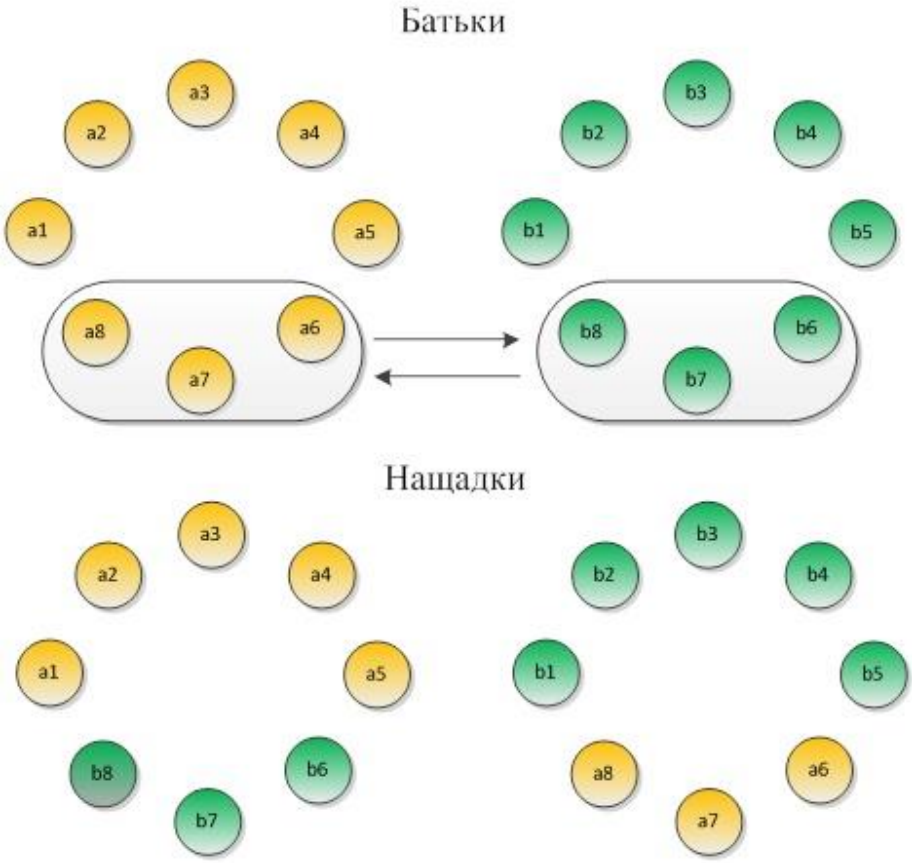


Рисунок 2.5 – Приклад генетичного алгоритму

Нейронні мережі навчаються, аналізуючи приклади атак у кожному класі. Після навчання вони використовуються для визначення правильної поведінки одного з класів атаки. Нейронні мережі використовуються на рівні мережі та вузла. Крім того, вони адаптовані і мають значно меншу обчислювальну складність.

Отримані дані.

У таблиці 2.1 наведено результати розрахунку оцінок для кожного методу випробування відповідно до критеріїв Recal, Precision і F-вимірювання. На основі порівняння цих результатів можна виділити наступні методи:

- метод опорних векторів, нейронна мережа;
- алгоритм кластеризації (k-найближчий сусід);
- генетичні і роєві алгоритми;
- метод інформаційної ентропії.

Таблиця 2.1 – Порівняльний аналіз алгоритмів

| Метод | Recal | Precision | F, % |
|---|---|---|---|
| Алгоритм кластеризації k-NN | 81,32 | 86,56 | 79,05 |
| Метод Баєвської мережі | 53,36 | 91,83 | 67,50 |
| Дерева рішень | 72,54 | 71,08 | 65,75 |
| Роєві алгоритми | 84,83 | 85,85 | 85,36 |
| Генетичні алгоритми | 84,53 | 85,02 | 84,78 |
| Імунні системи | 31,09 | 94,22 | 93,50 |
| Метод опорних веторів | 83,62 | 83,41 | 80,26 |
| Нейронні мережі | 86,03 | 87,02 | 83,40 |
| Інформаційно-ентропійний метод | 88,08 | 95,27 | 94,50 |

Порівняння методів виявлення мережевих атак показало, що найефективнішими методами за цими критеріями є опорні вектори, нейронні мережі та алгоритми кластеризації. (K-найближчий сусід), стадо та генетичний алгоритм. Однак жоден з цих методів не гарантує 100% виявлення вторгнень, а різні методи можуть успішно боротися з різними типами атак. На основі цього розвитку найефективніший метод залишається однією з найперспективніших задач при розробці систем виявлення вторгнення.

## 2.2 Вибір алгоритму на основі Ентропії

Наведені методи зазвичай використовуються для виявлення мережевих аномалій на основі статистичних методів. Алгоритм кластеризації (зазвичай метод K-середніх). модель Маркова; вейлвет аналіз, нейронна мережа, штучна імунна система.

Статистичні характеристики, такі як середнє значення вибірки, дисперсія вибірки, критерій відповідності Пірсона X та теоретична ентропія інформації, також можуть використовуватися як параметри мережевого трафіку для виявлення мережевих атак. Кількісна ентропія характеризується ентропією розподілу ймовірностей К. Шеннона.

Метою є підвищення ефективності IDS (intrusion detection systems), ADS (anomaly detection system) та систем управління інформаційною безпекою, виконання теоретичних та експериментальних досліджень із вивчення можливості використання значень обчисленої в режимі реального часу інформаційної ентропії в якості базового індикатора атаки на мережеві сервіси [8].

Використання значень ентропії мережевого трафіку для виявлення атак DDoS — це алгоритм що побудований на короткостроковій середній ентропії трафіку (локальна міра невизначеності) і довгостроковій відповідній вимірювання, яка розрахована без атаки (глобальна) на мережевий сервіс. Вимірювання невизначеності базуються на порівнянні. Якщо локальне вимірювання сильно відрізняється від відповідного глобального вимірювання, ймовірність мережевої атаки значно збільшується.

Відповідно до формули Шеннона (2.4), ентропія трафіку залежить від ймовірності того, що пакет буде надіслано:

$$H(x) = -\sum_{i=1}^{n} p_i \times \log_2 p_i, \tag{2.4}$$

де мірою ймовірності появи $P_i$ пакету і- того типу може виступати його частота, що визначається за формулою (2.5):

$$f_i = \frac{n_i}{N}, \tag{2.5}$$

де $n_i$ – кількість пакетів і-го типу;

N – загальна кількість пакетів в аналізованому трафіку.

Якщо рахувати за цим класичним алгоритмом, вам доведеться перерахувати частоту всіх пакетів, коли ви отримаєте нові пакети. Велика кількість пакетів значно сповільнить обчислення ентропії, що неприпустимо під час DDOS-атак. Для прискорення обчислення ентропії мережевого трафіку використовувався метод рухомого вікна.

Розрахунки ентропії за цим методом виконуються за таким алгоритмом: вибирається вікно розміру пакету W, обчислюється і зберігається частота кожного пакета типу $f_i$ і визначається основне значення ентропії $H_0$ для першого W пакету трафіку. Потім вікно переміщується вправо на ΔW відповідно до послідовності пакетів, що надходять у мережевий інтерфейс. Записуються частоти пакетів $f_{before.input}$ і $f_{now.input}$ всередині вікна, а також частоти пакетів $f_{before.output}$ і $f_{now.output}$, які залишаються поза вікном після зсуву. За цих умов поточне значення ентропії було розраховано з використанням формули (2.6):

$$H_i = H_{i-1} + \Delta H, \tag{2.6}$$

де $\Delta H$ розраховується за формулою (2.7):

$$\Delta H = -f_{before,input} \times \log f_{before.input} - f_{before.output} \times \log f_{before.output} + f_{now.input} \times \log f_{now.input} + f_{now.output} \times \log f_{now.output}, \tag{2.7}$$

де ΔH - показує зміну ентропії при зміщенні вікна, що представлено на рисунку 2.6.

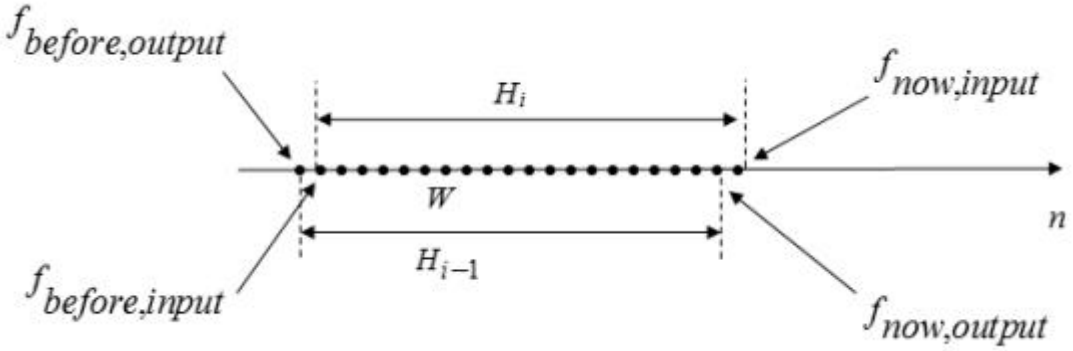


Рисунок 2.6 – Приклад методу рухомого вікна

Висновки. Аналіз отриманих експериментальних результатів дозволяє зробити висновок, що інформаційний аналіз трафіку в реальному масштабі часу може бути використано як ефективний індикатор аномального стану мережі. Зокрема, значення ентропії трафіку є чутливими до DDoS-атак, із розвитком атаки її значення зменшуються майже на 70%. При цьому слід відзначити, що на результати обчислень не впливає розмір пакету. У запропонованій постановці задачі на ентропію впливає сам факт надходження пакету з визначеними параметрами. Відповідно, обчислення ентропії трафіку можна застосовувати в системах виявлення вторгнень IDS, системах управління інформаційною безпекою підприємства, системах підтримки прийняття рішень та ін. Безпосередньо обчисленням повинні передувати декілька підготовчих етапів, зокрема визначення розміру рухомого [9].

## 2.3 Вибір середовища розробки

### 2.3.1 Середовище розробки Microsoft Visual Studio

Для вирішення цієї проблеми було обрано Microsoft Visual Studio 2019, інтерактивне середовище програмування та візуалізації.

Microsoft Visual Studio - серія продуктів фірми Майкрософт, які містять інтегроване середовище розробки програмного забезпечення та низку інших інструментальних засобів. Ці продукти дають змогу розробляти як консольні програми, так і програми з графічним інтерфейсом, включно з підтримкою технології Windows Forms, а також вебсайти, вебзастосунки, вебслужби як в рідному, так і в керованому кодах для всіх платформ, що підтримуються Microsoft Windows, Windows Mobile, Windows Phone, Windows CE, .NET Framework, .NET Compact Framework та Microsoft Silverlight [10].

Середовище Visual Studio доступне для Windows та Mac. Функції Visual Studio для Mac багато в чому аналогічні можливостям Visual Studio для Windows та оптимізовані для розробки крос-платформних та мобільних програм [10].

Інтегроване середовище розробки (IDE) – це багатофункціональна програма, яка підтримує багато аспектів розробки програмного забезпечення. Інтегроване середовище розробки Visual Studio – це стартовий майданчик для написання, налагодження та складання коду, а також подальшої публікації програм. Крім стандартного редактора і відладчика, які є в більшості середовищ IDE, Visual Studio включає компілятори, засоби автозавершення коду, графічні конструктори і багато інших функцій для поліпшення процесу розробки [10].

### 2.3.2 Переваги Microsoft Visual Studio

Visual Studio містить редактор коду, який підтримує IntelliSense (компонент завершення коду) і модифікацію коду. Інтегрований тюнер діє як на рівні джерела, так і як тюнер на рівні машини. Інші інтегровані інструменти включають інструмент профілювання коду, інструмент розробки додатків із графічним інтерфейсом користувача, вебдизайнер, конструктор класів і розробник схеми бази даних. Додатки, які розширюють функціональність майже на кожному рівні, включаючи підтримку систем керування кодом (таких як Subversion і Git), нові набори інструментів, такі як редактори та візуальні дизайнери для мов, пов’язаних із доменом, та інші набори інструментів. Я прийняв це. Аспекти розробки програмного забезпечення. життєвий цикл (наприклад, клієнт Azure DevOps: Team Explorer).

Visual Studio підтримує 36 різних мов програмування, тому редактори та коригувальники коду можуть підтримувати майже будь-яку мову програмування (різною мірою), якщо є спеціальна мова. Вбудовані мови включають C, C ++, C ++ / CLI, Visual Basic .NET, C #, F #, JavaScript, TypeScript, XML, XSLT, HTML і CSS. Підтримка інших мов, таких як Python, Ruby, Node.js і M, доступна через плагіни. Java (і J #), що підтримувалася раніше.

Версія для спільноти, найпростіша версія Visual Studio, доступна безкоштовно. Девіз спільноти Visual Studio — «безкоштовна, всеосяжна IDE для студентів, відкритих кодів і розробників на замовлення».

Visual Studio, як правило, не підтримує мови програмування, рішення чи інструменти. Замість цього ви можете підключити функції, закодовані як пакет VS. Після встановлення цю функцію можна використовувати як послугу. IDE надає три послуги. SVsSolution, який дозволяє перераховувати проєкти та рішення. SVsUIShell. Надає функції віконного та користувацького інтерфейсу, включаючи вкладки, панелі інструментів та панелі інструментів. SVsShell для реєстрації пакетів VS. Крім того, IDE відповідає за координацію послуг і зв'язок. Усі редактори, конструктори, типи проєктів та інші інструменти встановлюються як пакети VS. Visual Studio використовує COM для доступу до пакетів VS. Пакет SDK Visual Studio також включає платформу керованих пакетів (MPF). Це набір керованих пакетів інтерфейсу COM, який дозволяє писати пакунки на мові, сумісній з CLI. Однак MPF не надає всіх функцій інтерфейсу COMVisualStudio. Потім ви можете використовувати службу для створення інших пакетів, які доповнюють функції Visual Studio IDE.

Підтримка мови програмування додається через спеціальний пакет VSP під назвою Language Services. Мовні служби визначають різні інтерфейси, які інсталяція пакета VS може реалізувати для підтримки різноманітних функцій. компілювати. Якщо інтерфейс встановлено, функції будуть доступні цією мовою. Мовні послуги надаються на мовній основі. Інсталяції можуть повторно використовувати код з аналізатора або компілятора мови. Мовні служби можуть бути встановлені з локальним або керованим кодом. Ви можете використовувати свій інтерфейс COM або Babel Framework (частина пакета SDK Visual Studio) для локального коду. Для керованого коду MPF є обгорткою для розробки керованих мовних служб.

Visual Studio не включає вбудовану підтримку керування джерелами, але перелічує два альтернативні способи інтеграції системи керування кодом

із IDE. Source Control VS Package може надати власний налаштований інтерфейс користувача. На відміну від цього, надбудови для керування кодом, які використовують MSSCCI (інтерфейс керування вихідним кодом Microsoft), надають набір функцій, які використовуються для реалізації різних функцій керування кодом у стандартному інтерфейсі користувача Visual Studio. MSSCCI вперше використовувався для інтеграції Visual SourceSafe з Visual Studio 6.0, але пізніше був використаний за допомогою Visual Studio SDK. Visual Studio .NET 2002 використовувала MSSCCI 1.1, а Visual Studio .NET 2003 використовувала MSSCCI 1.2. Visual Studio 2005, 2008 і 2010 використовують MSSCCI версії 1.3. Це на додаток до підтримки перейменування та видалення та асинхронного відкриття.

Visual Studio підтримує запуск кількох екземплярів вашого середовища (кожен із власним набором пакетів VS). Екземпляри використовують різні кульки реєстру (див. визначення MSDN для терміну "кущ реєстру", як він використовується тут), щоб зберегти статус конфігурації та мати різні AppIds (ідентифікатори програми). Копіювання виконується спеціальним AppId.exe, який вибирає AppId, встановлює кореневий вулик і запускає IDE. Пакети VSP, зареєстровані в одному AppId, будуть інтегровані з іншими пакетами VI в цьому AppId. Різні версії продуктів Visual Studio створюються за допомогою різних AppIds. Продукти Visual Studio Express мають власний AppIdID, але стандартні, професійні продукти та продукти TeamSuite мають однаковий AppId. Тому, на відміну від інших випусків, які оновлюють ту саму інсталяцію, Express Edition можна інсталювати разом з іншими випусками. Професійне видання містить додатковий набір пакетів VS для стандартної версії, а набір команд містить додатковий набір пакетів VS для обох інших випусків. Система AppId використовується оболонкою Visual Studio у Visual Studio 2008.

IntelliSense - це набір можливостей, що відображають відомості про код безпосередньо в редакторі і в деяких випадках автоматично створюють невеликі уривки коду. По суті, це вбудована редактор базова документація, яка

позбавляє необхідності шукати інформацію в інших джерелах. Приклад цієї технології зображено на рисунку 2.7 [10].

CodeLens допомагає знаходити посилання на код, зміни коду, пов'язані з кодом помилки, робочі елементи, перевірки коду та модульні тести – не виходячи з редактора. Приклад використання цього інструменту зображено на рисунку 2.8 [10].

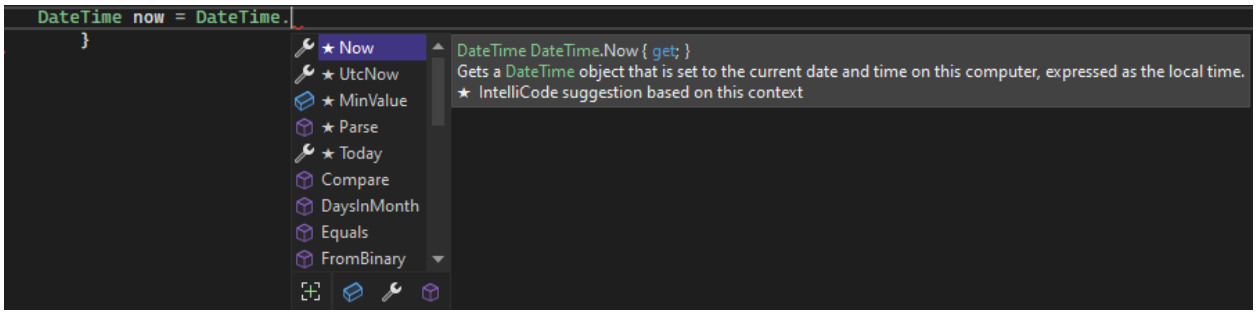


Рисунок 2.7 – Використання IntelliSense

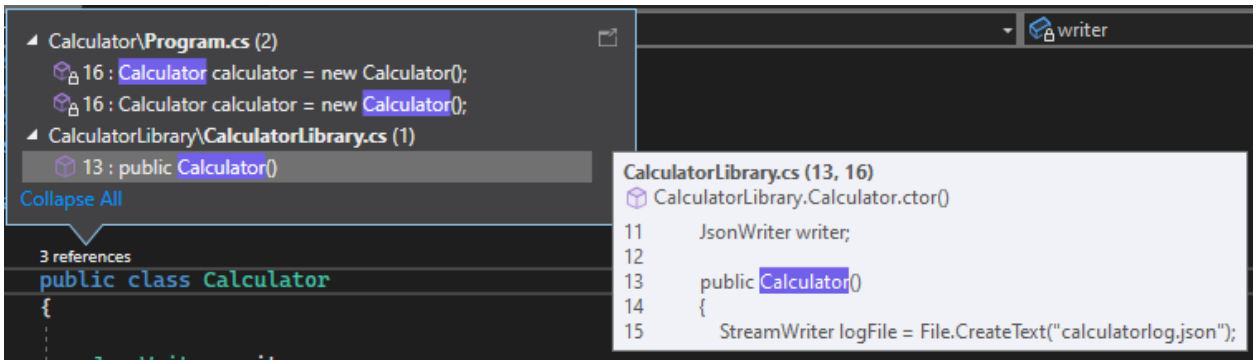


Рисунок 2.8 – Використання CodeLens

Live Share - Надає можливості спільного редагування та налагодження в реальному часі незалежно від типу програми або мови. Ви можете миттєво надавати спільний доступ до свого проєкту за допомогою високого рівня безпеки. Крім того, ви можете надавати спільний доступ до сеансів, екземплярів терміналу, вебдодатків на локальному комп'ютері, голосових дзвінків тощо [10].

Модульний інсталятор Visual Studio дозволяє вибирати та встановлювати потрібні робочі навантаження. Робочі навантаження – це групи

функцій, які мають працювати у мовах програмування чи платформах. Така модульна стратегія скорочує обсяг установки Visual Studio, прискорюючи встановлення та оновлення середовища [11].

### 2.3.3 Вибір мови програмування

C# (вимовляється як "сі шарп") - сучасна об'єктно-орієнтована і типобезпечна мова програмування. C# дозволяє розробникам створювати різні типи безпечних та надійних програм, що виконуються в .NET. C# відноситься до широко відомого сімейства мов C, і здасться добре знайомим будь-кому, хто працював з C, C++, Java або JavaScript [12].

C# — об'єктно-орієнтована, орієнтована на компоненти мова програмування. C# надає мовні конструкції безпосередньої підтримки такої концепції роботи. Завдяки цьому C# підходить для створення та застосування програмних компонентів. З моменту створення мова C# збагатилася функціями для підтримки нових робочих навантажень та сучасними рекомендаціями щодо розробки ПЗ. В основному C# - об'єктно-орієнтована мова. Ви визначаєте типи та їх поведінку [12].

### 2.3.4 Архітектура .Net

Програми C# виконуються в .NET, віртуальній системі виконання, що викликає загальномовне середовище виконання (CLR) та набір бібліотек класів. Середовище CLR – це реалізація загальномовної інфраструктури мови (CLI), що є міжнародним стандартом від корпорації Майкрософт. CLI є основою для створення середовищ виконання та розробки, в яких мови та бібліотеки прозоро працюють одна з одною [12].

Вихідний код, написаний мовою C#, компілюється в проміжну мову (IL), який відповідає специфікаціям CLI. Код мовою IL та ресурси, у тому числі растрові зображення та рядки, зберігаються у збірці, зазвичай з розширенням

.dll. Складання містить маніфест з інформацією про типи, версії, мову та регіональні параметри для цієї збірки [12].

Під час виконання програми C# збірка завантажується у середовище CLR. Середовище CLR виконує JIT-компіляцію з коду мовою IL в інструкції машинної мови. Середовище CLR також виконує інші операції, наприклад, автоматичне складання сміття, обробку виключень та управління ресурсами. Код, який виконується середовищем CLR, іноді називають "керованим кодом". "Некерований код" компілюється на машинну мову, призначену для конкретної платформи [12].

Забезпечення взаємодії між мовами є основною особливістю .NET. Код IL, створений компілятором C# відповідає специфікації загальних типів (CTS). Код IL, створений з коду C#, може взаємодіяти з кодом, створеним з версій .NET для мов F#, Visual Basic, C++. Існує більше 20 інших мов, сумісних із CTS. Одна збірка може містити кілька модулів, написаних різними мовами .NET, і всі типи можуть посилатися один на одного, якби вони були написані однією мовою [12].

Крім служб часу виконання .NET також включає розширені бібліотеки. Ці бібліотеки підтримують багато різних робочих навантажень. Вони впорядковані за просторами імен, які надають різні корисні можливості: від операцій файлового введення та виведення до керування рядками та синтаксичного аналізу XML, від платформ вебдодатків до елементів керування Windows Forms. Зазвичай програму C# активно використовують бібліотеку класів .NET для вирішення типових завдань [12].

### 2.3.5 Проєктування графіки за допомогою середовища Visual Studio

GDI + — це інтерфейс прикладного програмування (API), який є підсистемою операційної системи Microsoft Windows. GDI + відповідає за відображення інформації на екрані та принтері. Як випливає з назви, GDI + є

наступником GDI, інтерфейсу графічного пристрою, який був включений у попередні версії Windows.

GDI + API надається через набір класів, встановлених як керований код. Цей набір класів називається інтерфейсом керованого класу GDI +. Інтерфейс керованого класу складається з таких імен:

- System.Drawing;
- System.Drawing.Drawing2D;
- System.Drawing.Imaging;
- System.Drawing.Text;
- System.Drawing.Printing.

Графічні пристрої, такі як GDI +, можна використовувати для відображення інформації на екрані або принтері, не турбуючись про деталі конкретного пристрою відображення. Програміст виконує виклики методів, наданих класом GDI +. Ці методи викликають драйвер для певного пристрою. GDI + відокремлює програму від графічного обладнання. Саме цей поділ дозволяє програмістам створювати незалежні від пристрою програми.

### 2.3.6 NetFlow

Netflow — це мережевий протокол, розроблений компанією Cisco Systems на основі мережевого трафіку. Це справді промисловий стандарт і підтримується не тільки апаратним забезпеченням Cisco, але й багатьма іншими пристроями (включаючи Juniper і Enterasys). Установка систем типу UNIX також існує.

Існує кілька версій протоколу, найпоширенішою з яких на 2011 рік є версії 5 і 9. На основі версії 9 також був розроблений відкритий стандарт під назвою Internet Protocol Flow Information eXport (IPFIX).

Для збору інформації про трафік NetFlow необхідні наступні компоненти:

- датчик. Збирає статистику про трафік на ньому. Зазвичай це комутатор або маршрутизатор L3, але ви також можете використовувати окремий датчик, завдяки якому ви отримаєте дані, відображаючи порти комутатора;
- колектор. Збирає та зберігає дані, отримані від датчика;
- аналізатор. Він аналізує дані, зібрані виборцем, і готує читабельний звіт (часто у вигляді графіка).

## 2.4 Висновок за розділом 2

У цьому розділі ми розглянули та проаналізували різні види алгоритмів розпізнавання мережевих аномалій. На основі отриманих результатів зроблено висновок, що найбільш ефективний алгоритм заснований на ентропії Шеннона.

Також було розглянуто та продемонстровано переваги вибору середовища розробки Visual Studio 2019 та симулятора колектора NetFlow для програмного забезпечення «Виявлення мережевих атак за допомогою ентропії».

# 3 ОСНОВНІ РІШЕННЯ ЩОДО РЕАЛІЗАЦІЇ КОМПОНЕНТІВ СИСТЕМИ

## 3.1 Проєктування програмної реалізації ентропії

Програма призначена для аналізу та інтерпретації мережевих даних для виявлення та класифікації мережевих аномалій.

Цей програмний продукт надає користувачам такі функції:

- можливість вибору ширини вікна для перегляду та навігації по розкладу мережевого трафіку;
- можливість вибору порту для отримання мережевих даних;
- відображення аномалії мережі на графіку мережевого трафіку;
- розрахунок основних параметрів ентропії Шеннона;
- здатність аналізувати мережу за допомогою атрибутів мережевого трафіку;
- перегляд графічного розкладу аналізу.

Час виконання програми залежить від конфігурації комп'ютера, на якому виконується програма, та конфігурації мережевого обладнання.

Для експлуатації програмного продукту були визначені такі рекомендовані характеристики програмно технічних засобів:

- персональний комп’ютер з процесором Intel Pentium D з тактовою частотою 2 GHz або вище;
- операційна система Microsoft Windows 7 або вище;
- обсяг оперативно запам’ятовуючого пристрою не менше 1 Гб;
- жорсткий або твердотілий накопичувач з вільним обсягом не менше 2 Гб;
- програмне забезпечення Microsoft .Net Framework 4.5;
- екран з розширенням 1024x768 або вище;
- комп’ютерна миша та клавіатура.

## 3.2 Розробка програмних модулів

У ході розробки програми були використано таку мову програмування як C#, для аналізу мережевих даних було обрано протокол NetFlow, для збору інформації використовується колектор мережевого трафіку NetFlow, користувацький інтерфейс було реалізовано за допомогою вбудованого класу Visual Studio, WinForms, за побудову графіків зміни ентропії у реальному часі відповідає бібліотека для мови C# - LiveGrid. Також для оптимізації роботи програми у реальному часі та покращення користувацького досвіду були використані можливості багатопоточності мови C#, що представлені класом Thread.

Також для реалізації функцій з передачі аналізованого трафіку була обрана база даних Microsoft SQL Server 2019. Мова запитів що використовується в базі даних – SQL.

Програма складається з модульних блоків, кожен з яких виконує свої функції. Взаємодія між ними відбувається за допомогою головної форми програми. Модульна схема програми представлена на рисунку 3.1.

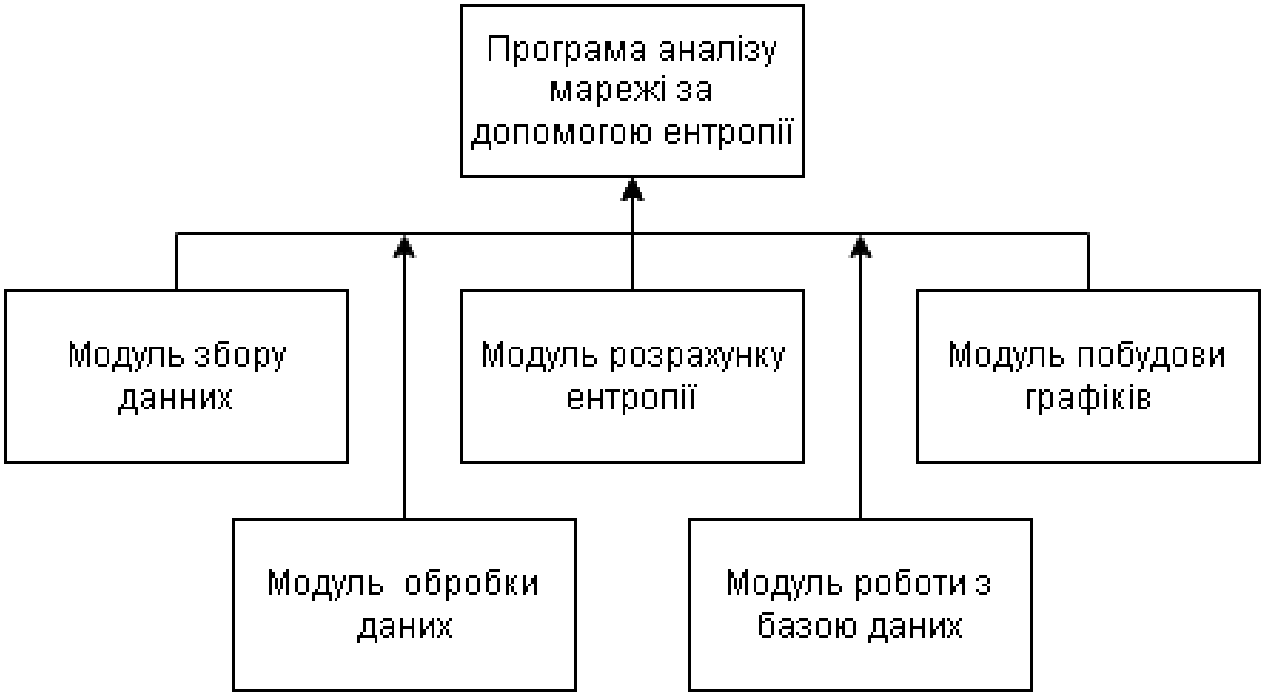


Рисунок 3.1 – Модульна схема програми

Програма має ієрархічну структуру і складається з таких п'яти функціональних модулів:

- модуль збору даних;
- модуль розрахунку ентропії;
- модуль малювання графіків;
- модуль обробки даних;
- модуль для роботи з базами даних.

Програма контролює точність даних і відповідність формату. Коли вона отримує неформатовані дані, програма розпізнає їх і форматує відповідно до необхідного формату.

Роботу програми гарантує послідовна обробка мережевих сигналів за своєю функцією, послідовність якої виконується за схемою, зазначеною розробником.

Кожен модуль вирішує окрему задачу. Розглянемо призначення основного програмного модуля докладніше.

Модуль збору даних дозволяє підключатися і слухати на вибраному порту, а також завантажувати мережеві дані, надіслані колектором.

Модуль обчислення ентропії використовується для обчислення значення ентропії мережевого трафіку за певний період часу.

Модуль графіків призначений для створення графіків на основі даних, обчислених іншими модулями, та показу їх користувачеві.

Модуль обробки даних призначений для обробки вхідних даних і форматування їх у формат, необхідний для подальшого розрахунку.

Модулі роботи з базами даних призначені для використання баз даних для взаємодії з програмними продуктами. Ви можете виконувати різні операції з базою даних, наприклад, додавати, оновлювати, видаляти та отримувати дані відповідно до різних критеріїв вибору. База даних у даному проєкті використовується виключно в цілях збереження розшифрованих та згрупованих даних.

Програма реалізована на мові C#, та використовує багато внутрішніх функцій. Приклади їх використання були наведені у таблиці 3.1.

Таблиця 3.1 – Приклади використаних при розробці функцій мови C#

| Назва функції | Синтаксис | Опис |
|---|---|---|
| 1 | 2 | 3 |
| InitializeComponent | private void InitializeComponent()<br>{<br>тіло функції ініціювання головного вікна<br>} | Запускає початкову ініціалізацію головного вікна програми. |
| SqlConnection | SqlConnection.SqlConnection(строка підключення до бази даних) | Створить новий екземпляр класу SqlConnection, використовуючи рядок підключення до бази даних. |
| SqlConnection.Open | SqlConnection.Open(); | Відкриває підключення до бази даних, використовуючи значення властивості, визначене SqlConnection.ConnectionString. |
| SqlConnection.Close | SqlConnection.Close(); | Ініціює закриття з’єднання з базою даних. |
| SqlCommand | SqlCommand(string command, SqlConnection connection) | Ініціює новий екземпляр класу SqlCommand, використовуючи вміст рядку запиту та підключення до бази даних SQLConnection. |
| ExecuteScalar | SqlCommand. ExecuteScalar() | Виконує запит і повертає перший рядок першого стовбця, що був отриманий з набору даних. |
| ExecuteReader | SqlCommand.ExecuteReader() | Надсилає SqlCommand для запуску та створює SqlDataReader, який є контейнером для отримання. |
| ExecuteNonQuery | SqlCommand.ExecuteNonQuery() | Виконує оператор Transact-SQL і повертає кількість рядків у інструкції. |
| ToInt32 | Convert.ToInt32(Object) | Перетворює вказаний об’єкт у відповідне 32-розрядне ціле число. |
| Socket | Socket(AddressFamily.InterNetwork, SocketType.Dgram, ProtocolType.Udp); | Ініціює екземпляр класу Socket, використовуючи вказане сімейство адрес, тип сокета та протокол зв’язку. |
| IPEndPoint | IPEndPoint(IPAddress, Int32) | Створює новий екземпляр класу IPEndPoint із зазначеною адресою та номером порту. |
| Bind | Socket.Bind(EndPoint); | Пов’язує об’єкт Socket з локальною кінцевою точкою. |
| ReceiveFrom | Socket.ReceiveFrom(byte, EndPoint) | Прийміть дейтаграму в буфер даних і зберігає кінцеву точку. |

Продовження таблиці 3.1

| 1 | 2 | 3 |
|---|---|---|
| Packet | Packet.Packet(bytes, _templates); | Створює екземпляр класу Packet за допомогою набору отриманих даних та шаблону розшифровки даних |
| ToString | Packet.ToString(); | Перетворює екземпляр класу Packet в рядок, що можливий для читання |
| ScrollToCaret | RichTextBox.ScrollToCaret() | Прокручує вміст контролера до поточної позиції курсору. |
| ToTransport() | Packet.ToTransport() | Перетворіть елемент упаковки в транспортний клас для подальшого використання. |
| InsertInto | DbWorker.InsertInto(Transport) | Перетворює екземпляр класу у формат, необхідний для запису в базу даних, і надає запити на запис даних. |
| CalculationsOfEntropy() | Form1.Entropy(); | Обчислює ентропію, використовуючи інформацію з бази даних. |
| SelectUnicIpSource | DbWorker. SelectUnicIpSource (); | Сформуйте запит до бази даних, щоб отримати унікальний запис. |
| Read | SqlDataReader.Read(); | Зчитує дані в буфер і переміщує курсор SqlDataReader в інше місце. Повертає значення true, якщо є будь які символи, інакше false. |
| Add | List<T>.Add(T) | Додайте об’єкт T-типу в кінець послідовності списку T-типу. |
| Close | SqlDataReader.Close(); | Закриває екземпляр класу SqlDataReader |
| SelectCountIpSource | DbWorker. SelectCountIpSource (string) | Створіть запит до бази даних і запустіть його за допомогою пов’язаної бази даних. Повертає кількість записів, які відповідають умові рядка. |
| SelectCount | DbWorker.SelectCount() | Створіть запит до бази даних і запустіть його за допомогою пов’язаної бази даних. Повертає кількість записів у базі даних. |
| Log | Math.Log(p[i], 2) | Повертає логарифм числа p [i], заданого у вказаному радикалі 2. |
| DateTime.Now | DateTime.Now | Повертає екземпляр класу DateTime, який за місцевим часом означає поточний рік, місяць, день, хвилину, |
| AddRange | List.AddRange(object) | Додає елемент до вказаного набору об’єктів у кінець списку. |

Продовження таблиці 3.1

| 1 | 2 | 3 |
|---|---|---|
| ToLongTimeString | DateTime.ToLongTimeString(); | Перетворює значення вказаного об’єкта DateTime в еквівалентний довгостроковий перегляд дати та часу. |
| AxisX.Clear() | AxisX.Clear() | Ця функція видаляє всі елементи з колекції. |
| LineSeries() | LineSeries.LineSeries() | Конструктор класу LineSeries, створює новий екземпляр класу |
| Reverse() | Reverse() | Змінює на зворотній порядок об’єктів в послідовності |

## 3.3 Розробка користувацького інтерфейсу

Основна проблема у створенні інтерфейсу користувача полягає в тому, щоб зробити інтерфейс максимально зручним для користувача та прибрати зайву інформацію, щоб зробити його простіше у використанні.

Інтерфейс програми було розроблено за допомогою вбудованих інструментів Visual Studio, що представлені у вигляді класу Forms.

Form — це представлення будь-якого вікна, яке відображається в програмі. Клас Form можна використовувати для створення стандартних, інструментів, безмежних та плаваючих вікон. Клас також можна використовувати для створення модальних вікон, таких як діалогове вікно. Форма з декількома документами (MDI) може містити інші форми, які називаються дочірніми формами MDI. Форма MDI створюється шляхом надання властивості IsMdiContainer значення true. Дочірні форми MDI створюються шляхом встановлення MdiParent властивості батьківської форми MDI, яка міститиме дочірню форму [13].

Використовуючи властивості, доступні Form у класі, можна визначити зовнішній вигляд, розмір, колір та функції керування вікнами вікна або діалогового вікна. Властивість Text дозволяє вказати заголовок вікна у рядку

заголовка. Властивості Size дозволяють DesktopLocation визначити розмір та положення вікна під час його відображення. Властивість кольору можна використовувати для зміни кольору переднього плану за промовчанням для всіх елементів керування, розміщених у формі. MinimizeBoxВластивості FormBorderStylei MaximizeBox властивості дозволяють керувати тим, чи може бути згорнута, розгорнута або змінена під час виконання [13].

Крім властивостей, можна використовувати методи класу для керування формою. Наприклад, можна використовувати ShowDialog метод для відображення форми у вигляді модального діалогового вікна. Метод можна використовувати для SetDesktopLocation розміщення форми на робочому столі [13].

Події Form класу дозволяють реагувати на дії, які виконуються у формі. Подія можна використовувати для Activated виконання таких операцій, як оновлення даних, що відображаються в елементах керування форми під час активації форми [13].

Форму можна використовувати як початковий клас у додатку, помістивши метод, що викликається Main у класі. У методі Main додайте код для створення та відображення форми. Крім того, для запуску форми необхідно додати STAThread атрибут Main в метод. При закритті початкової форми програма також закривається [13].

Якщо для властивості false встановлено значення Enabled перед Form видимим (наприклад, якщо Enabled встановлено значення false у конструкторі Microsoft Visual Studio), згортання, розгортання, закриття та системні кнопки залишаються включеними. Якщо встановлено значення Enabled false після Form відображення (наприклад, при виникненні події завантаження), кнопки будуть відключені [13].

Розглянемо інтерфейс програми. Огляд головного вікна програми показано на рисунку 3.2.

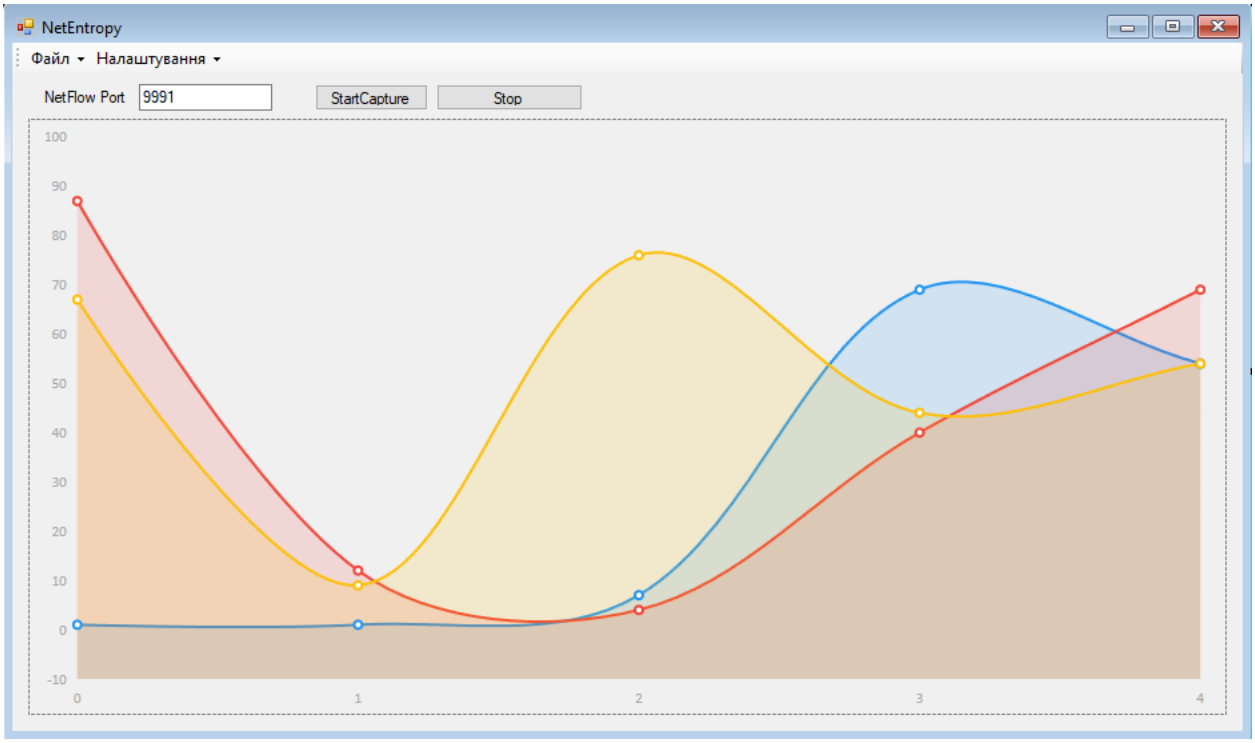


Рисунок 3.2 – Головне вікно програми

### 3.4 Принцип роботи програми

Розглянемо алгоритм роботи програми. Аналіз мережі виконується наступним чином: спочатку необхідно налаштувати мережеве обладнання для збору даних в колектор NetFlow, якщо подібного обладнання немає для перевірки програми можна скористатись різноманітними програмним забезпеченням що може генерувати NetFlow потік, в цій роботі було використане програмне забезпечення з симуляції потоку NetFlow, Netflow Simulator by Jupin Jin.

Програма працює за наступним алгоритмом: після підключення до порту прослуховування потоку NetFlow, програма починає отримувати пакети.

Спочатку виконується перевірка пакету на відповідність шаблону протоколу NetFlow, щоб виключити можливість отримання невідповідних даних, що можуть викликати виключення. Для цього використовується клас «Template».

Вміст класу «Template»:

```
[Serializable]
 public class Template
 {
 private UInt16 counter;
 private UInt16 ident1;
 private List<Field> contextList1;

 private Byte[] context1;


 public UInt16 Counter1
 {
 get
 {
 return this.counter;
 }
 }

 public UInt16 Ident1
 {
 get
 {
 return this.ident1;
 }
 }

 public List<Field> ContextList1
 {
 get
```

```
{
return this.contextList1;
}
}

public Template(Byte[] bytes)
{
this.context1 = bytes;
this.Parse();
}

public UInt16 FieldLength
{
get
{
UInt16 tmplen = 0;
foreach (Field fields in this.contextList1)
{
 tmplen += fields.Length;
}
return tmplen;
}
}

private void Parse()
{
byte[] reverse = this.context1.Reverse().ToArray();
this.contextList1 = new List<Field>();
```

```
this.ident1 = BitConverter.ToUInt16(reverse, this.context1.Length - sizeof(Int16) - 0);
this.counter = BitConverter.ToUInt16(reverse, this.context1.Length - sizeof(Int16) - 2);

if (this.context1.Length == ((this.counter * 4) + 4))
{
for (int i = 0, j = 4; i < this.counter; i++, j += 4)
{
 Byte[] bfield = new Byte[4];
 Array.Copy(this.context1, j, bfield, 0, 4);
 Field field = new Field(bfield);
 this.contextList1.Add(field);
}
}
}
}
```

Далі двійкові дані пакету повністю конвертуються в користувацький клас «Packet», для розшифровки вмісту пакету та отримання повної інформації.

Процес конвертації був реалізований таким чином:

```
private void Parse(Templates templates)
{
this.flowlist = new List<FlowSet>();

Int32 length = bytes.Length - 20;

Byte[] header = new Byte[20];
Byte[] flowset = new Byte[length];
```

```
Array.Copy(bytes, 0, header, 0, 20);
Array.Copy(bytes, 20, flowset, 0, length);

this.headPack = new Header(header);
byte[] reverse = flowset.Reverse().ToArray();

int templengh = 0;

while ((templengh + 2) < flowset.Length)
{
UInt16 lengths = BitConverter.ToUInt16(reverse, flowset.Length - sizeof(Int16) - (templengh + 2));
Byte[] bflowsets = new Byte[lengths];
Array.Copy(flowset, templengh, bflowsets, 0, lengths);

FlowSet flowsets = new FlowSet(bflowsets, templates);
this.flowlist.Add(flowsets);

templengh += lengths;
}
}
```

Після конвертації пакет у вигляді класу «Packet» зберігає усю інформацію у вигляді що може прочитати людина або для потреб логування.

Код функції що виводить рядок розшифрованого потоку:

```
public override string ToString()
{
String ret = "NetFlow Packet " + this.bytes.Length + "b.\r\n"
+ "Header " + "20b.:\r\n"
+ "Version: " + headPack.Version + "\r\n"
```

```
 + "Count: " + headPack.Count + "\r\n"
 + "UpTime: " + headPack.UpTime + "\r\n"
 + "Secs: " + headPack.Secs + "\r\n"
 + "Sequence: " + headPack.Sequence + "\r\n"
 + "ID: " + headPack.ID + "\r\n\n"
 + "FlowSet`s " + (this.bytes.Length - 20) + "b.:\r\n\n";
int a = 0;

 foreach (FlowSet flows in this.flowlist)
 {
 a++;

 ret += "FlowSet [" + a + "]:" + "\r\n"
 + "ID: " + flows.ID + "\r\n"
 + "Length: " + flows.Length + "\r\n";

 foreach (Byte bt in flows.ValueByte)
 {
  ret += "0x" + bt.ToString("X") + " ";
 }

 int i = 1;
 foreach (Template templ in flows.Template)
 {
  ret += i + "\tTemplate:" + "\r\n"
  + "\tID: " + templ.Ident1 + "\r\n"
  + "\tCount: " + templ.Counter1 + "\r\n";
  foreach (Field fields in templ.ContextList1)
  {
  ret += "\t\t" + fields.Type + ": ";
```

```
if (fields.Value.Count == 0) ret += "\r\n";

if ((fields.GetTypes() == (UInt16)FieldType.IPV4_DST_ADDR) ||
(fields.GetTypes() == (UInt16)FieldType.IPV4_SRC_ADDR) ||
(fields.GetTypes() == (UInt16)FieldType.IPV4_NEXT_HOP) ||
(fields.GetTypes() == (UInt16)FieldType.IPV6_DST_ADDR) ||
(fields.GetTypes() == (UInt16)FieldType.IPV6_SRC_ADDR) ||
(fields.GetTypes() == (UInt16)FieldType.IPV6_NEXT_HOP))
{
if (fields.Value.Count != 0) ret += new IPAddress(fields.Value.ToArray()).ToString();
}
else if ((fields.GetTypes() == (UInt16)FieldType.L4_DST_PORT) ||
(fields.GetTypes() == (UInt16)FieldType.L4_SRC_PORT))
{
if (fields.Value.Count != 0) ret += BitConverter.ToUInt16(fields.Value.ToArray().Reverse().ToArray(), 0);
}
else
{

foreach (Byte bt in fields.Value)
{
ret += "0x" + bt.ToString("X") + " ";
}
}

if (fields.Value.Count != 0) ret += "\r\n";
}
```

```
 i++;
 }
 }

 return ret;
 }
```

Наступним кроком ці дані записуються у базу даних. Це необхідно для можливості передачі цих даних на інший комп’ютер та можливості відтворення графіку з раніше записаних даних.

Далі виконується алгоритм обчислення ентропії Шенона, за допомогою якого визначаються мережеві аномалії.

Інформаці́йна ентропі́я (англ. information entropy) — це усереднена швидкість, із якою продукує інформацію стохастичне джерело даних [14].

Мірою інформаційної ентропії, пов'язаною з кожним можливим значенням даних, є від'ємний логарифм функції маси ймовірності для цього значення. Таким чином, коли джерело даних має менш імовірне значення (наприклад, коли стається низькоймовірна подія), то ця подія несе більше «інформації» («неочікуваності»), ніж коли джерело даних має більш імовірне значення. Визначена таким чином кількість інформації, що передається кожною подією, стає випадковою змінною, чиє математичне сподівання є інформаційною ентропією. Взагалі, ентропію позначають безлад або невизначеність, і визначення ентропії, що застосовують в теорії інформації, є прямим аналогом визначення, що застосовують у статистичній термодинаміці. Поняття інформаційної ентропії було введено Клодом Шенноном у його праці 1948 року «Математична теорія зв'язку» [14].

Базова модель системи передавання даних складається з трьох елементів: джерела даних, каналу зв'язку та приймача, і, за виразом Шеннона, «фундаментальною задачею зв'язку» є те, щоби отримувач був здатен

встановлювати, які дані було породжено джерелом, на основі сигналу, що він отримує каналом. Ентропія забезпечує абсолютну межу найкоротшої можливої усередненої довжини безвтратного стиснювального кодування даних, продукованих джерелом, і якщо ентропія джерела є меншою за пропускну спроможність каналу зв'язку, то дані, породжувані цим джерелом, можливо надійно передавати приймачеві [14].

Код обчислення значення ентропії:

```
public void CalculationsOfEntropy()
 {
 Action action = () =>
 {
 List<string> unicIpSrc = new List<string>();
 List<int> counters = new List<int>();
 int sumCount;
 _DataBaseWorker.SelectUnicIpSource();
 while (_DataBaseWorker.sqlDataReader.Read())
 {
  unicIpSrc.Add(_DataBaseWorker.sqlDataReader["IPv4Src"].ToString());
 }
 _DataBaseWorker.sqlDataReader.Close();
 foreach (var item in unicIpSrc)
 {
  counters.Add(_DataBaseWorker.SelectCountIpSource(item));
 }
 _DataBaseWorker.sqlDataReader.Close();
 sumCount = _DataBaseWorker.SelectCount();
 _DataBaseWorker.sqlDataReader.Close();

 double h1;
```

```
double H0variable;
double Htmp = 0;

double[] p = new double[unicIpSrc.Count];

for (int i = 0; i < unicIpSrc.Count; i++)
{
 p[i] = counters[i] / (double)sumCount;
 Htmp += p[i] * Math.Log(p[i], 2);
}
h1 = -Htmp;
H0variable = h1 / Math.Log(unicIpSrc.Count, 2);
ListofEntropy1.Add(H0variable);
};

if (InvokeRequired)
Invoke(action);
else
action();
}
```

Останнім кроком роботи алгоритму буде побудова графіку. Для побудови графіків в програмі використовується потужна графічна бібліотека LiveCharts. Ця бібліотека дозволяє швидко та зручно будувати різноманітні графіки та діаграми майже у реальному часі. Вона написана на мові C# що гарантує її правильну роботу у середовищі .Net.

Код що відповідає за створення, доповнення графіку та відмальвування графіку на формі:

```
void DrawGraph()
```

```
{
Action action = () =>
{
DateTime timeFN = DateTime.Now;

SeriesCollection seriesCol = new SeriesCollection();

ChartValues<double> chartsListHo = new ChartValues<double>();

List<string> dates = new List<string>();

cartesianChart1.LegendLocation = LegendLocation.Bottom;
chartsListHo.AddRange(ListofEntropy1);
ListofEntrophyTime.Add(timeFN.ToLongTimeString());
dates.AddRange(ListofEntrophyTime);

cartesianChart1.AxisX.Clear();

cartesianChart1.AxisX.Add(new Axis()
{
 Title = "Time",
 Labels = dates
});

LineSeries line = new LineSeries();
line.Title = "ip source entropy";
line.Values = chartsListHo;

seriesCol.Add(line);
```

```
cartesianChart1.Series = seriesCol;
};

if (InvokeRequired)
Invoke(action);
else
action();
}
```

### 3.5 Висновки за розділом 3

В цьому розділі описана головна інформація про розробку застосунку виявлення мережевих аномалій.

Було проведено ознайомлення з модулями розроблюваної системи. Розроблено основні алгоритми, які застосовуються в роботі проєкту – а саме алгоритм опрацювання вхідного сигналу з колектору NetFlow, алгоритм вирахування ентропії мережі на основі даних колектору.

Зроблено загальний опис створеного програмного продукту та розглянуто основні модулі та функції роботи.

# 4 КЕРІВНИЦТВО ПРОГРАМІСТА

## 4.1 Призначення й умови використання програми

Програма призначена для автоматизації процесу виявлення мережевих атак за допомогою ентропії.

Програма призначена для аналізу та інтерпретації мережевих даних та послідуючого виявлення та класифікації мережевих аномалій використовуючи ентропію.

Програмний продукт був розроблений за допомогою мови програмування C# та з використанням протоколу NetFlow.

Головною умовою використання програми є наявність визначеного обладнання, що підтримує функції та виконує моніторинг мережі як колектор NetFlow. Також для роботи необхідно встановити фреймворк виконання Microsoft .Net 4.6. Це програмне забезпечення необхідне для коректної роботи програми.

Для роботи програми необхідно – комп’ютер з пристроями вводу і виводу (комп’ютерна миша та клавіатура), необхідна підтримка мережевих взаємодій за допомого мережевої карти з роз’ємом RJ-45, або за допомогою мережевих карт з підтримкою безпровідного з’єднання WiFi, або аналогічних апаратних можливостей вбудованих всередину материнської плати персонального комп’ютера.

## 4.2 Характеристики розробленої системи

При розробці програмного продукту були створені такі логічні блоки, які реалізують певну бізнес логіку програми. Основними такими логічними блоками ж такі класи:

- Form1.cs – основний логічний блок, що реалізує логіку головної форми, включає функцій взаємодії з користувачем;

- Packet.cs – блок, що реалізує збереження та роботу з отриманими даними;
- DbWorker.cs – блок, що працює з базою даних;
- Header.cs – блок призначений для розпізнавання заголовків пакетів та їх фільтрації.

Далі будуть наведені певні функції що присутні у блоках.

Функція запису нових даних у базу даних, ця функція необхідна для додавання інформації у підключену базу даних, код функції представлений нижче.

```
public void InsertIntoAn(Temporary pack)
 {
 if (pack.ipSource is null)
 {
 return;
 }
 comSql = new SqlCommand();
 string sql = "Insert into netdb (IPv4Src ,IPv4Dest, PortSrc, PortDest, DateTime)
";
 sql += "values (@IPv4Src ,@IPv4Dest, @PortSrc, @PortDest, @DateTime)";
 comSql.CommandText = sql;
 comSql.Connection = sqlCon1;
 comSql.Parameters.AddWithValue("@IPv4Src", pack.ipSource);
 comSql.Parameters.AddWithValue("@IPv4Dest", pack.ipDestenation);
 comSql.Parameters.AddWithValue("@PortSrc", pack.portSource);
 comSql.Parameters.AddWithValue("@PortDest", pack.portDestenation);
 comSql.Parameters.AddWithValue("@DateTime", pack.dateTime);
 comSql.ExecuteNonQuery();
                }
```

При запиті на додавання інформації в базу даних спочатку необхідно перевірити наявність находженої інформації, щоб виключити хибне спрацювання функції, та послідуючі виключення для подібного прецеденту. Після успішного проходження перевірки функція почне формувати строку запиту та з'єднання з екземпляром бази даних, що представлено полем Connection.

Функція розбору потоку на складові частини потрібна для можливості аналізу внутрішніх даних. Протокол NetFlow при обробці мережевого трафіку проводить збір інформації, сортування за попередньо встановленими оператором критеріями та запаковує її у так звані потоки (flow). Для розбору цих потоків і необхідна дана функція. Код функції представлений нижче.

```
private void Parse(Templates templates)
        {
        byte[] reverse = this._bytes.Reverse().ToArray();
        this._template = new List<Template>();
        this._valuebyte = new List<byte>();

        this._id = BitConverter.ToUInt16(reverse, this._bytes.Length - sizeof(Int16) - 0);
        this._length = BitConverter.ToUInt16(reverse, this._bytes.Length - sizeof(Int16) - 2);

        if ((this._id == 0) || (this._id == 1))
        {
        if (this._id == 0)
        {
         int cout, pastaddress, address = 6;

         while (address < this._bytes.Length)
         {
```

```
cout = BitConverter.ToUInt16(reverse, this._bytes.Length - sizeof(Int16) - address);

int length = cout * 4 + 4;
Byte[] btemplate = new Byte[length];
Array.Copy(this._bytes, address - 2, btemplate, 0, length);

Template template = new Template(btemplate);

this._template.Add(template);

Boolean flag = false;
Template[] templs = templates.Templats.ToArray();

for (int i = 0; i < templs.Length; i++)
{
if (template.Ident1 == templs[i].Ident1)
{
flag = true;
templs[i] = template;
}
}

if (flag)
{
templates.Templats = templs.ToList();
}
else
{
templates.Templats.Add(template);
```

```
 }

 pastaddress = address;
 address = cout * 4 + pastaddress + 4;
 }
}
}
else if (this._id > 255)
{
Boolean flag = false;
Template templs = null;

foreach (Template templ in templates.Templats)
{
 if (templ.Ident1 == this._id)
 {
 templs = DeepClone(templ) as Template;
 flag = true;
 }
}

int j = 4, z;

if (flag)
{

 z = (this._length - 4) / templs.FieldLength;

 for (int y = 0; y < z; y++)
 {
```

```
foreach (Field fields in templs.ContextList1)
{
for (int i = 0; i < fields.Length; i++, j++)
{
fields.Value.Add(this._bytes[j]);
}
}

this._template.Add(DeepClone(templs) as Template);

foreach (Field filds in templs.ContextList1)
{
filds.Value.Clear();
}
}
}

if (!flag)
{
for (int i = 4; i < this._bytes.Length; i++)
{
this._valuebyte.Add(this._bytes[i]);
}
}
}
}
```

Функція на вхід отримує список шаблонів за допомогою якого буде виконуватись розпізнавання типу інформації що міститься всередині потоку та зворотне сортування їх по необхідним даним всередині функції.

Функція що виконує розбір шапки пакетів отриманих з розбору потоку NetFlow. Необхідна для отримання даних про тип пакету, час надходження, версію, послідовність та поверхневого опису вмісту пакету. Код функції представлений нижче.

```
private void Parse()
        {
            if (this._bytes.Length == 20)
            {
                byte[] reverse = this._bytes.Reverse().ToArray();

                this._version = BitConverter.ToUInt16(reverse,
this._bytes.Length - sizeof(Int16) - 0);
                this._count = BitConverter.ToUInt16(reverse,
this._bytes.Length - sizeof(Int16) - 2);

                this._uptime = BitConverter.ToUInt32(reverse,
this._bytes.Length - sizeof(UInt32) - 4);
                this._secs = BitConverter.ToUInt32(reverse,
this._bytes.Length - sizeof(Int32) - 8);
                this._sequence = BitConverter.ToUInt32(reverse,
this._bytes.Length - sizeof(Int32) - 12);
                this._id = BitConverter.ToUInt32(reverse,
this._bytes.Length - sizeof(Int32) - 16);
            }
    }
```

### 4.3 Вхідні та вихідні дані

Вхідними даними для програми є мережевий потік оброблений, згрупований та зашифрований колектором мережевого трафіку NetFlow.

Вихідними даними для програми є графік обробленого мережевого трафіку, за допомогою ентропії, а також за необхідності база даних з отриманими та розподіленими пакетами сеансу.

## 4.4 Структура компонентів програмного продукту

Якщо розглядати програмний продукт з боку структурної схеми то можна поділити його на три основні частини. Частина що відповідає за роботу з мережевим обладнанням, отриманням пакетів, розшифровкою пакетів, визначення їх типу та визначення відповідності пакету певному шаблону отримуваних даних. Частина що відповідає за опрацювання інформації, запис розшифрованих пакетів до бази даних, зчитування даних з бази даних, вирахування основних показників мережевого трафіку та отримання значення ентропії. Частина що відповідає за графічне відображення графіку отриманої ентропії, виконує побудову графіків, діаграм та оновлення їх у реальному часі.

## 4.5 Висновки за розділом 4

У цьому розділі було розглянуті основні характеристики програмного продукту, надано схематичне визначення основним компонентам системи та описане головне призначення програми.

Були розглянуті більш детально основні логічні блоки ба наведені приклади функцій що використовуються в них.

# 5 КЕРІВНИЦТВО КОРИСТУВАЧА

## 5.1 Призначення програми

Програма призначена для користувачів що мають знання у комп’ютерних науках та перед якими стоїть завдання відстежування мережевих аномалій.

Програма була розроблена за допомогою об’єктно орієнтованої мови програмування C#, використовує мережевий протокол призначений для обліку мережевого трафіку NetFlow, та відкриту бібліотеку для побудови графіків LiveCharts також написану на мові C#.

Головною умовою використання програми є наявність визначеного обладнання, що підтримує функції та виконує моніторинг мережі як колектор NetFlow. Також для роботи необхідно встановити фреймворк виконання Microsoft .Net 4.6. Це програмне забезпечення необхідне для коректної роботи програми.

Для роботи програми необхідно – комп’ютер з пристроями вводу і виводу (комп’ютерна миша та клавіатура), необхідна підтримка мережевих взаємодій за допомого мережевої карти з роз’ємом RJ-45, або за допомогою мережевих карт з підтримкою безпровідного з’єднання WiFi, або аналогічних апаратних можливостей вбудованих всередину материнської плати персонального комп’ютера.

## 5.2 Початок роботи з програмою

Для початку роботи з програмою необхідно виконати підготовчі дії. Першочергово потрібно виконати налаштування мережевого обладнання, маршрутизатора, для того щоб він почав виконувати облік мережевого трафіку та його відправку у вигляді потоків за допомогою протоколу NetFlow. Варто попередити що не всі маршрутизатори підтримують дану технологію. Для тестування програми також можливо використовувати різноманітне

програмне забезпечення що виконує ті самі функції, та симулює поведінку справжнього мережевого обладнання, що дозволяє отримати згенерований потік NetFlow. У цій роботі було використано програмне забезпечення з симуляції мережевого потоку NetFlow, «Netflow Simulator by Jupin Jin» що розповсюджується за ліцензією «AS IS». Далі розглянемо налаштування програми симуляції потоку. Якщо ви маєте сумісний роутер то вам необхідно налаштувати його відповідно до інструкцій фірми виробника або постачальника послуг. Почнемо з головного вікна програми, що зображене на рисунку 5.1.

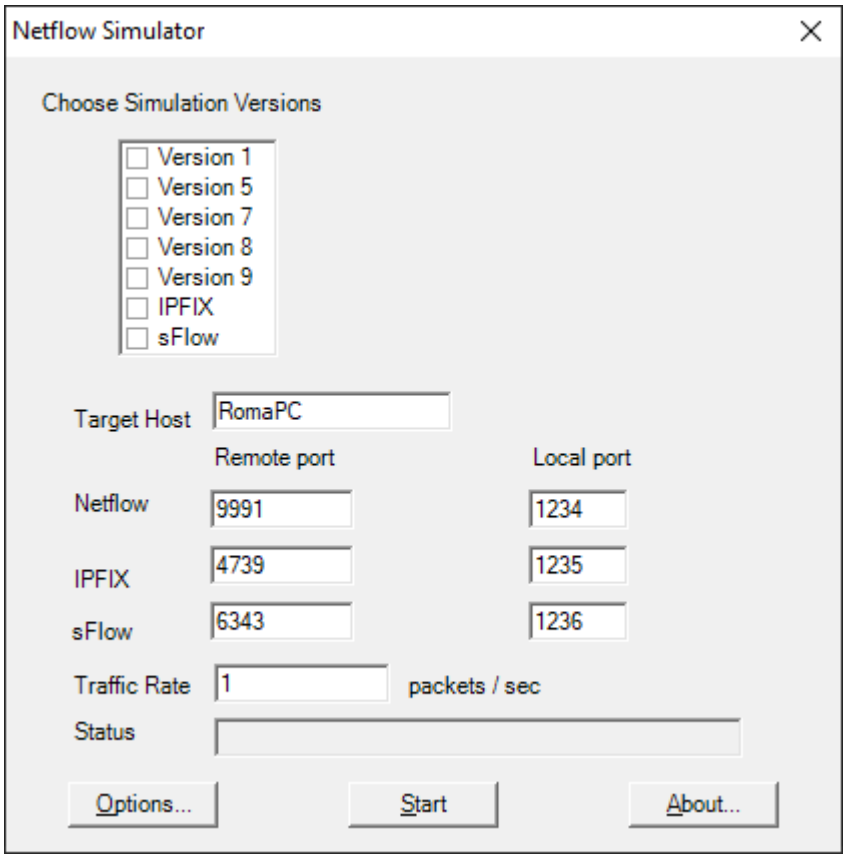


Рисунок 5.1 – Вигляд головного вікна програми симуляції NetFlow

На початку використання необхідно обрати тип потоку що буде симулювати програма. Зі списку шо знаходиться під написом Choose Simulation Version, нам необхідно обрати Version 9. Нам буде видано

додаткове вікно з налаштуваннями шаблону потоку. Представлене на рисунку 5.2.

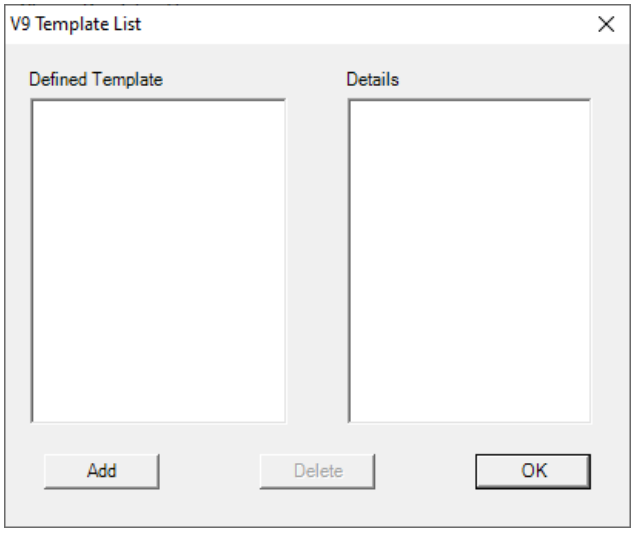


Рисунок 5.2 – Вікно конфігурування шаблону потоку

У цьому вікні потрібно створити нову конфігурацію шаблону потоку, для цього потрібно натиснути кнопку «Add». Далі з’явиться нове вікно, яке призначене для створення або корегування шаблонів, воно зображене на рисунку 5.3.

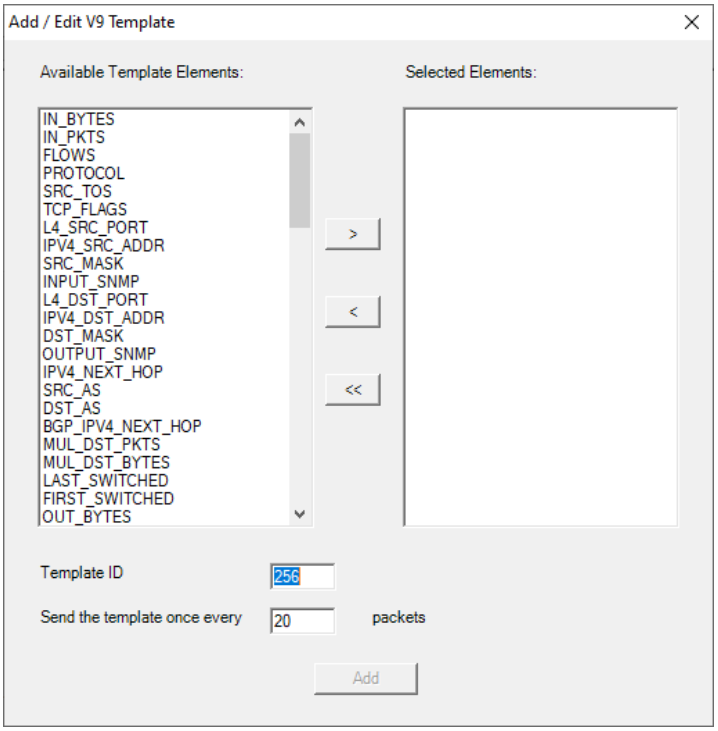


Рисунок 5.3 – Вікно корегування та створення нового шаблону потоку

У цьому вікні нам потрібно обрати такі варіанти як «L4_SRC_PORT», IPV4_SRC_ADDR», «IPV4_DEST_ADDR» та «L4_DST_PORT». Порядок у якому вибираються ці значення може бути будь яким, розроблена програма знайде необхідні поля на етапі розпізнавання шаблонів. Після того як ми перемістимо необхідні нам конфігурації у вікно «Selected Elements», кнопка «Add» внизу форми стане активною, натиснувши на неї ми зможемо перейти до наступного кроку.

Наступним кроком нас поверне на вікно зображене на рисунку 5.2. У списку «Defined Template» буде відображена єдиний варіант вибору. Необхідно натиснути на нього та потім натиснути кнопку «ОК».

Після цього програма знову покаже нам головне вікно, що зображене на рисунку 5.1. У чекбоксі напроти Version 9 тепер буде маркер, це покажчик того що симулятор налаштований на генерування потоку NetFlow версії 9. На цьому вікні нам необхідно запам’ятати значення що знаходиться у рядку «NetFlow» і у стовбці «Remote port». Останнім кроком з налаштування симулятору буде необхідно натиснути кнопку «Start», що знаходиться у нижній частині форми, посередині.

З цього моменту програма починає постійно генерувати мережевий потік NetFlow на той порт, що ми запам’ятали.

Після цього налаштування ми можемо переходити до розробленого програмного продукту.

Для доступу до програмного продукту необхідно скопіювати його з носія, на якому його було придбано, на жорсткий диск персонального комп’ютера. Та встановити на цей комп’ютер необхідне програмне забезпечення що вказане в інструкції.

Для старту програми необхідно запустити до виконання файл з розширенням «exe».

Відкриваємо програму, з’явиться головне вікно програми, що зображено на рисунку 5.4.


Рисунок 5.4 – Головне вікно програми

Першим кроком необхідно ввести номер порту NetFlow на який раніше ми налаштували симулятор, цю інформацію необхідно було запам’ятати. Ці дані необхідно ввести у поле «NetFlow Port».

Далі слід натиснути на кнопку «StartCapture». Програма виконає підключення до заданого порту та майже одразу почне вираховувати значення ентропії та відображати її графік. Приклад зображено на рисунку 5.5.

Завдяки використаним можливостям мови програмування C# процес отримання даних, вирахування значення ентропії та побудова графіку виконується в паралельних потоках, що значно покращує характеристики швидкодії програми, дозволяє виконувати інші дії поки сеанс моніторингу мережі виконує своє завдання по виявленню мережевих аномалій.

Графік ентропії буде змінюватися кожен раз з надходженнями нових даних. Аномальна поведінка на даному графіку буде відображена у вигляді різкої зміни значення ентропії та відповідно зображено на графіку як різка зміна показників.

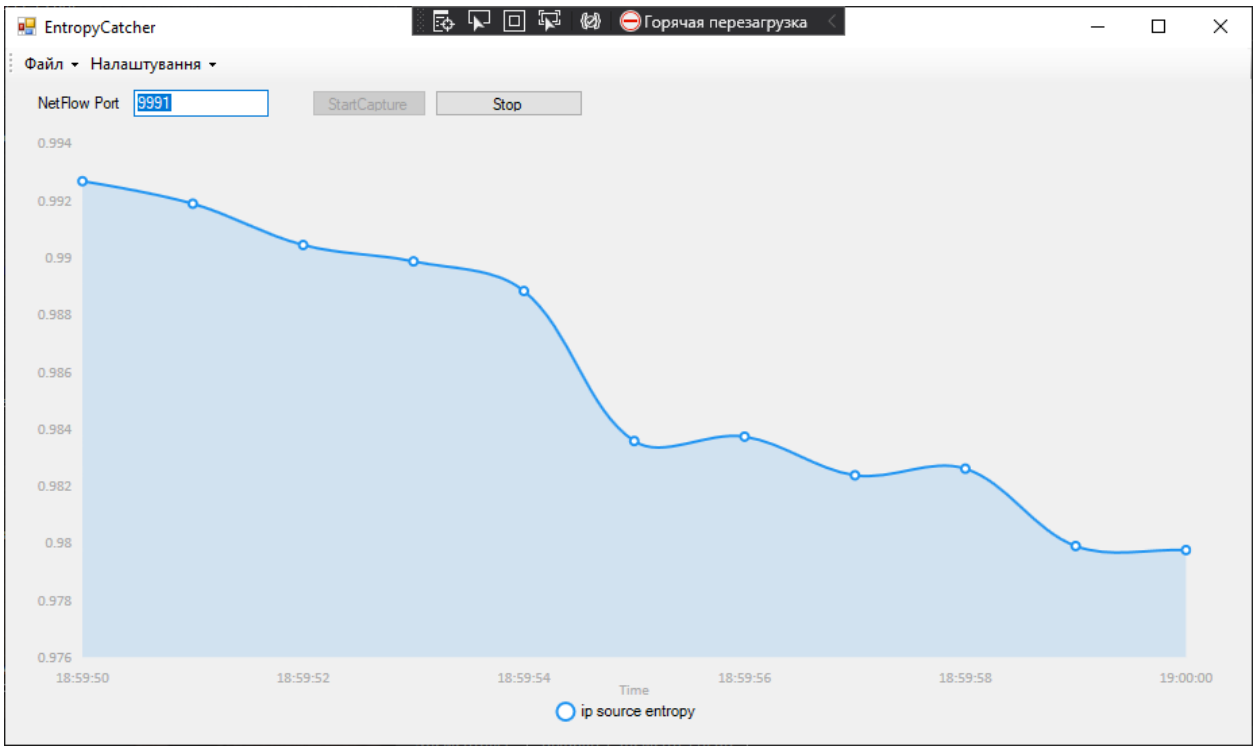


Рисунок 5.5 – Головне вікно програми після початку визначення аномалій

У разі виникнення помилкових прецедентів в програмі передбачено виведення сповіщень про помилки у консолі середовища.

Для виходу з програми необхідно натиснути кнопку «Закрити» у правому верхньому куті програми, або скористатись меню програми, обравши підменю «Файл» та вибрати варіант «Закрити».

## 5.3 Висновки за розділом 5

В розділі було описано процес налаштування симулятору потоку NetFlow, описано процес взаємодії користувача з програмним продуктом.

Інтерфейс розробленого додатку є максимально спрощеним для користувача, щоб користування було максимально приємним та збільшити площу відображення графіку мережі.

# ВИСНОВКИ

У процесі виконання дипломної кваліфікаційної роботи було проведено аналіз проблеми мережевих атак, проведено дослідження ефективності різних алгоритмів виявлення мережевих атак. Було розроблено програмне забезпечення, яке виконує аналіз мережевого трафіку для виявлення мережевих аномалій за допомогою ентропії Шенона.

Після тестування та перевірки справності відпрацювання методу виявлення мережевих атак можливо зробити такі висновки:

– функціонал програмного продукту відповідає вимогам поставленого технічного завдання в повній мірі;

– програма виконує поставлені завдання у багатопотоковому режимі;

– програма працює стабільно;

– програма має зручний та компактний інтерфейс користувача;

– програма забезпечує коректне розпізнавання різних типів пакетів що надсилає колектор мережевого трафіку;

– було наведено детальну інструкцію з використання програми.

Для розробки проєкту було обрано об’єктно орієнтовану мову програмування C#, був обраний мережевий протокол NetFlow для коректного обліку мережевого трафіку, для побудови графіків було використано бібліотеку побудови діаграм та графіків LiveCharts.

Для збереження даних було обрано базу даних Microsoft SQL Server 2019. Мовою запитів у цій базі даних виступає мова SQL.

Обране середовище розробки Microsoft Visual Studio повністю відповідає критеріям вибору зазначеним у постанові завдання, та має необхідний набір інструментів для розробки проєктованого додатку.

Обґрунтовано вибір середовища розробки і функціонування системи, та вибір мови програмування.

В результаті випробувань розроблена програма дозволяє виявляти мережеві атаки за допомогою ентропії мережі.

# ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Leland W. On the Self-Similar Nature of Ethernet Traffic. IEEE Transactions on Networking, [Текст] / W. Leland, M. Taqqu, W. Willenger, D. Wilson // 1994, №. 2. – С. 2 – 15.

2. Bulakh V. Classification of Multifractal Time Series by Decision Tree Methods. 14th International Conference ICTERI 2018 ICT in Education, Research, and Industrial Applications [Текст] / V. Bulakh , L. Kirichenko, T. Radivilova // 2018. - С.1-4.

3. Дубровін В.І. ВИЯВЛЕННЯ DOS-АТАК В МЕРЕЖЕВОМУ ТРАФІКУ МЕТОДОМ ВЕЙВЛЕТ-ПЕРЕТВОРЕННЯ [Текст] / В.І. Дубровін, Б. В. Петрик, Г.В. Неласа // Сучасний захист інформації, ISSN 2409-7292. - 2020. - №2(42). - С. 37-46.

4. Про основні засади забезпечення кібербезпеки України: Закон України від 05.10.2017 № 2163-VIII // Відомості Верховної Ради України. – 2017. – № 45. – С. 403.

5. Браницкий А. А. Анализ и классификация методов обнаружения сетевых атак, [Текст] / А. А. Браницкий, И. В. Котенко // СПИИРАН, 2016, выпуск 45, С. 207–244.

6. Hackeling G. Mastering machine learning with scikit-learn. [Текст] / G. Hackeling // Packt Publishing, 2014, - С. 238.

7. Кожевнікова І. С. Контрольовані методи машинного навчання як засіб детектування мережевих вторгнень [Текст] / І. С. Кожевнікова, Є. В. Ананьїн, А. В. Лисенко, А. В. Нікішова // Молодий учений. 2016. №27 (131). С. 24-29.

8. Бурлаков М. Є. Алгоритм виявлення вторгнень в інформаційних мережах на основі штучної імунної системи [Текст] / М. Є. Бурлаков канд. техн. наук.// Уфа, 2017 С. 36.

9. Dubey R. Empirical study of intrusion detection system using feature reduction based on evolutionary algoritms and swarm intelligence methods [Текст]

/ R. Dubey, D. Rathore // International Journal of Applied Engineering Research. 2017. Vol. 12. No. 19. C. 8884-8889.

10. Добро пожаловать в интегрированную среду разработки Visual Studio/ [Электрон. ресурс]. – Режим доступа: https://docs.microsoft.com/ru-ru/visualstudio/.

11. Возможности Visual Studio [Электрон. ресурс]. – Режим доступа: https://docs.microsoft.com/ru-ru/visualstudio/ide/.

12. Краткий обзор языка C# [Электрон. ресурс]. – Режим доступа: https://docs.microsoft.com/ru-ru/dotnet/csharp/tour-of-csharp/.

13. Form Класс [Электрон. ресурс]. – Режим доступа: https://docs.microsoft.com/ru/dotnet/api/system.windows.forms.form?view=windowsdesktop-6.0/.

14. Інформаційна ентропія [Електрон. ресурс]. – Режим доступу: https://uk.wikipedia.org/wiki/.

# ДОДАТОК А

## Технічне завдання

## А.1 Підстави для розробки

Підставою для розробки програмного продукту є завдання на дипломну кваліфікаційну роботу на тему «Виявлення мережевих атак за допомогою ентропії» і затверджене наказом Національного університету «Запорізька політехніка» № 102 від 27 квітня 2022 р.

## А.2 Призначення розробки

Мета роботи – розробка програмного забезпечення, яке виконує аніліз мережевого трафіку для виявлення мережевих аномалій за допомогою ентропії Шенона.

## А.3 Основні вимоги до програми, що розробляється

### А.3.1 Вимоги до функціональних характеристик

Програма повинна аналізувати мережевий трафік, та за допомогою даних отриманих з колектору проводити розрахунок ентропії.

Графік ентропії перераховується та виводиться у реальному часі що забезпечує велику швидкість визначення аномалій.

### А.3.2 Вимоги до надійності

Програма повинна проводити коректну обробку виключень у таких ситуаціях:

– при виборі зайвих критеріїв формування потоку NetFlow;

– при спробі доступу до інтерфейсу програми або значень графіку при виконанні аналізу мережі.

### А.3.3 Умови експлуатації

Під час виконання програмою обчислень не рекомендується вимикати живлення або перезавантажувати персональний комп’ютер.

Для використання програми клієнтом необхідно мати персональний комп’ютер та маршрутизатор з підтримкою протоколу NetFlow, і мати постійний доступом до мережі інтернет.

### А.3.4 Вимоги до складу та параметрів технічних засобів

Мінімально необхідні системні вимоги та програмне забезпечення, що дозволять експлуатувати програму визначені на такому рівні:

– процесор – не менше 2 GHz;
– оперативна пам'ять – не менше 2Гб;
– операційна система Microsoft Windows 7 або вище;
– жорсткий диск об’ємом не менше – 40Гб;
– маршрутизатор з підтримкою технології NetFlow;
– програмна технологія .Net Framework 4.6.

### А.3.5 Вимоги до маркування та упакування

Програма може розповсюджуватись за допомогою як Flash-накопичувачів, так і на компакт-дисках. Програмний продукт повинен мати позначку товарного знаку компанії розробника, номера версії. На пакуванні повинна бути зазначена назва програми – «Програма для виявлення мережевих аномалій за допомогою ентропії».

### А.3.6 Вимоги до транспортування та збереження

Вимоги до транспортування та зберігання програмного виробу мають бути відповідні вимогам, що висуваються до носія інформації, на якому розповсюджується програма.

### А.3.7 Вимоги до програмної документації

До складу програмної документації повинно входити «Керівництво користувача».

# ДОДАТОК Б

## Текст програми

DbWorker.cs

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

namespace EntropySolution
{
 class DbWorker
 {
 private string connString = ConfigurationManager.ConnectionStrings["NetEntropy.Properties.Settings.netdbConnectionString"].ConnectionString;

 SqlCommand comSql = null;

 private SqlConnection sqlCon1 = null;

 public SqlDataReader sqlDataReader = null;

 public void SelectFirst()
 {
 string commandTmp = $"Select top 1 From netdb";
 comSql = new SqlCommand(commandTmp, sqlCon1);
 comSql.ExecuteNonQuery();
```

```
        }

        public int SelectCount()
        {
        DateTime timeNow = DateTime.Now;
        DateTime pastTime = DateTime.Now.AddSeconds(-30);

        string commandTmp = $"Select count(*) From netdb where DateTime BETWEEN '{pastTime.ToString("yyyy-MM-dd HH:mm:ss")}' and '{timeNow.ToString("yyyy-MM-dd HH:mm:ss")}' ";
        comSql = new SqlCommand(commandTmp, sqlCon1);
        return Convert.ToInt32(comSql.ExecuteScalar());
        }

        public int SelectCountIpSource(string ipv4)
        {
        DateTime timeNow = DateTime.Now;
        DateTime pastTime = DateTime.Now.AddSeconds(-30);

        string commandTmp = $"Select count(*) From netdb where IPv4Src = '{ipv4}' and DateTime BETWEEN '{pastTime.ToString("yyyy-MM-dd HH:mm:ss")}' and '{timeNow.ToString("yyyy-MM-dd HH:mm:ss")}' ";
        comSql = new SqlCommand(commandTmp, sqlCon1);
        return Convert.ToInt32(comSql.ExecuteScalar());
        }


        public void SelectUnicIpSource()
        {
        DateTime timeNow = DateTime.Now;
```

```
DateTime pastTime = DateTime.Now.AddSeconds(-30);
comSql = new SqlCommand();
string commandTmp = $"Select DISTINCT IPv4Src From netdb where DateTime BETWEEN @PastTime and @TimeNow ";
comSql.CommandText = commandTmp;
comSql.Connection = sqlCon1;
comSql.Parameters.AddWithValue("@PastTime", pastTime);
comSql.Parameters.AddWithValue("@TimeNow", timeNow);
sqlDataReader = comSql.ExecuteReader();
}

public void InsertIntoAn(Temporary pack)
{
if (pack.ipSource is null)
{
return;
}
comSql = new SqlCommand();
string sql = "Insert into netdb (IPv4Src ,IPv4Dest, PortSrc, PortDest, DateTime) ";
sql += "values (@IPv4Src ,@IPv4Dest, @PortSrc, @PortDest, @DateTime)";
comSql.CommandText = sql;
comSql.Connection = sqlCon1;
comSql.Parameters.AddWithValue("@IPv4Src", pack.ipSource);
comSql.Parameters.AddWithValue("@IPv4Dest", pack.ipDestenation);
comSql.Parameters.AddWithValue("@PortSrc", pack.portSource);
comSql.Parameters.AddWithValue("@PortDest", pack.portDestenation);
comSql.Parameters.AddWithValue("@DateTime", pack.dateTime);
comSql.ExecuteNonQuery();
```

```
        }

        public void InsertInto(Temporary pack)
        {
        string commandTmp = $"Insert into netdb (IPv4Src ,IPv4Dest, PortSrc, PortDest, DateTime) values ('{pack.ipSource}' ,'{pack.ipDestenation}', {pack.portSource},{pack.portDestenation},'{pack.dateTime.ToString("yyyy-MM-dd HH:mm:ss")}')"; //.ToString("yyyy-MM-dd HH:mm:ss")
        comSql = new SqlCommand(commandTmp, sqlCon1);
        comSql.ExecuteNonQuery();
        }

        public void DisConnect()
        {
        sqlCon1.Close();
        }

        public void Connect()
        {
        sqlCon1 = new SqlConnection(connString);
        sqlCon1.Open();
        }
        }
       }
```

Template.cs

```
using System;
using System.Collections.Generic;
using System.Text;
```

```
using System.Linq;

namespace EntropySolution
{
 [Serializable]
 public class Template
 {
 private UInt16 counter;
 private UInt16 ident1;
 private List<Field> contextList1;

 private Byte[] context1;


 public UInt16 Counter1
 {
 get
 {
 return this.counter;
 }
 }

 public UInt16 Ident1
 {
 get
 {
 return this.ident1;
 }
 }
```

```
public List<Field> ContextList1
{
get
{
return this.contextList1;
}
}

public Template(Byte[] bytes)
{
this.context1 = bytes;
this.Parse();
}

public UInt16 FieldLength
{
get
{
UInt16 tmplen = 0;
foreach (Field fields in this.contextList1)
{
 tmplen += fields.Length;
}
return tmplen;
}
}

private void Parse()
{
byte[] reverse = this.context1.Reverse().ToArray();
```

```
this.contextList1 = new List<Field>();

this.ident1 = BitConverter.ToUInt16(reverse, this.context1.Length - sizeof(Int16) - 0);
this.counter = BitConverter.ToUInt16(reverse, this.context1.Length - sizeof(Int16) - 2);

if (this.context1.Length == ((this.counter * 4) + 4))
{
for (int i = 0, j = 4; i < this.counter; i++, j += 4)
{
 Byte[] bfield = new Byte[4];
 Array.Copy(this.context1, j, bfield, 0, 4);
 Field field = new Field(bfield);
 this.contextList1.Add(field);
}
}
}
}
```

Temporary.cs

```
using System;

namespace EntropySolution
{
 [Serializable]
 public class Temporary
 {
```

```
public string ipDestenation;
public string ipSource;
public int portSource;
public DateTime dateTime;
public int portDestenation;

public string ShowTransport()
{
return "ipsrc "+ ipSource +" ipdes "+ ipDestenation + " portsrc " + portSource.ToString() + " portdest " + portDestenation.ToString() + " " + dateTime.ToString();
}
}
}
```

Form1.cs

```
using System;
using System.Collections.Generic;
using System.Windows.Forms;
using System.Net.Sockets;
using System.Net;
using System.Threading;
using LiveCharts;
using LiveCharts.Wpf;

namespace EntropySolution
{
public partial class Form1 : Form
```

```
{

DbWorker _DataBaseWorker = new DbWorker();
List<string> ListofEntrophyTime = new List<string>();
List<double> ListofEntropy1 = new List<double>();

bool workFlag = false;

public Form1()
{
InitializeComponent();
cartesianChart1.AxisX.Clear();
cartesianChart1.AxisY.Clear();
}



void StartAlgoritmWork()
{
_DataBaseWorker.Connect();
int socketIndex;

if (textBox1.Text != null)
{
socketIndex = Convert.ToInt32(textBox1.Text);
}
else
{
return;
}
```

```
Templates _templates = new Templates();

Socket socket = new Socket(AddressFamily.InterNetwork,
SocketType.Dgram, ProtocolType.Udp);
IPEndPoint iEndPoint1 = new IPEndPoint(IPAddress.Any, socketIndex);
socket.Bind(iEndPoint1);
EndPoint ep = (EndPoint)iEndPoint1;

byte[] data = new byte[2048];

while (workFlag)
{
int recived1 = socket.ReceiveFrom(data, ref ep);
byte[] bytes = new byte[recived1];
for (int i = 0; i < recived1; i++)
 bytes[i] = data[i];

Packet packet = new Packet(bytes, _templates);
List<Temporary> temporaryFDB = packet.ToTransport();

if (temporaryFDB.Count != 0)
{
 foreach (Temporary item in temporaryFDB)
 {
 _DataBaseWorker.InsertIntoAn(item);
 }
}
else
{
```

```
 continue;
}

CalculationsOfEntropy();
DrawGraph();
}
_DataBaseWorker.DisConnect();
socket.Close();
button1.Enabled = true;
}

public void StopWorkingAlgo()
{
workFlag = false;
button1.Enabled = true;
}

private void button2_Click(object sender, EventArgs e)
{
StopWorkingAlgo();
}

private void button1_Click(object sender, EventArgs e)
{
Thread thread = new Thread(StartAlgoritmWork);
thread.Start();
workFlag = true;
button1.Enabled = false;
}
```

```
public void CalculationsOfEntropy()
{
Action action = () =>
{
List<string> unicIpSrc = new List<string>();
List<int> counters = new List<int>();
int sumCount;
_DataBaseWorker.SelectUnicIpSource();
while (_DataBaseWorker.sqlDataReader.Read())
{
 unicIpSrc.Add(_DataBaseWorker.sqlDataReader["IPv4Src"].ToString());
}
_DataBaseWorker.sqlDataReader.Close();
foreach (var item in unicIpSrc)
{
 counters.Add(_DataBaseWorker.SelectCountIpSource(item));
}
_DataBaseWorker.sqlDataReader.Close();
sumCount = _DataBaseWorker.SelectCount();
_DataBaseWorker.sqlDataReader.Close();


double h1;
double H0variable;
double Htmp = 0;

double[] p = new double[unicIpSrc.Count];

for (int i = 0; i < unicIpSrc.Count; i++)
{
```

```
 p[i] = counters[i] / (double)sumCount;
 Htmp += p[i] * Math.Log(p[i], 2);
}
h1 = -Htmp;
H0variable = h1 / Math.Log(unicIpSrc.Count, 2);
ListofEntropy1.Add(H0variable);
};

if (InvokeRequired)
Invoke(action);
else
action();
}

void DrawGraph()
{
Action action = () =>
{
DateTime timeFN = DateTime.Now;

SeriesCollection seriesCol = new SeriesCollection();

ChartValues<double> chartsListHo = new ChartValues<double>();

List<string> dates = new List<string>();

cartesianChart1.LegendLocation = LegendLocation.Bottom;
chartsListHo.AddRange(ListofEntropy1);
ListofEntrophyTime.Add(timeFN.ToLongTimeString());
dates.AddRange(ListofEntrophyTime);
```

```
cartesianChart1.AxisX.Clear();

cartesianChart1.AxisX.Add(new Axis()
{
 Title = "Time",
 Labels = dates
});

LineSeries line = new LineSeries();
line.Title = "ip source entropy";
line.Values = chartsListHo;

seriesCol.Add(line);

cartesianChart1.Series = seriesCol;
};

if (InvokeRequired)
Invoke(action);
else
action();
}

private void СтартToolStripMenuItem_Click(object sender, EventArgs e)
{
button1.PerformClick();
}

private void СтопToolStripMenuItem_Click(object sender, EventArgs e)
```

```
{
button2.PerformClick();
}

private void ЗакритиToolStripMenuItem_Click(object sender, EventArgs
e)
{
Application.Exit();
}

private void ПортToolStripMenuItem_Click(object sender, EventArgs e)
{
textBox1.Focus();
}
}
}
```

# ДОДАТОК В

## Слайди презентації

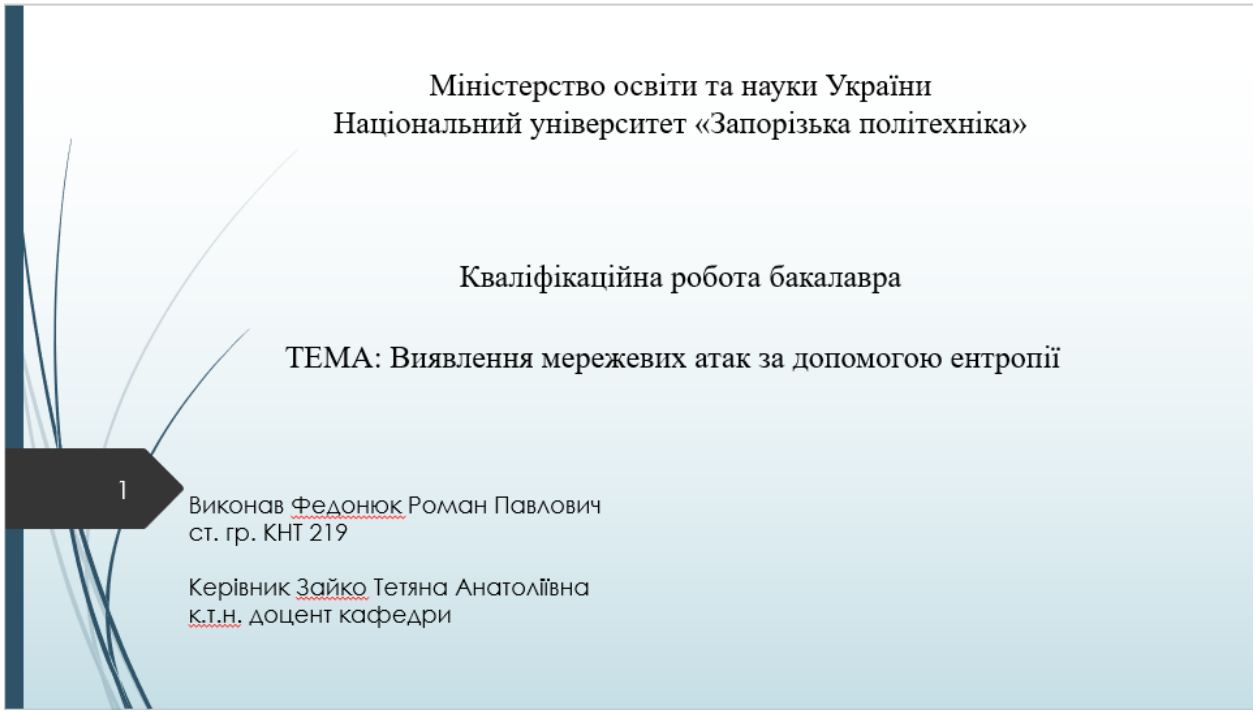


Рисунок В.1 – Слайд 1

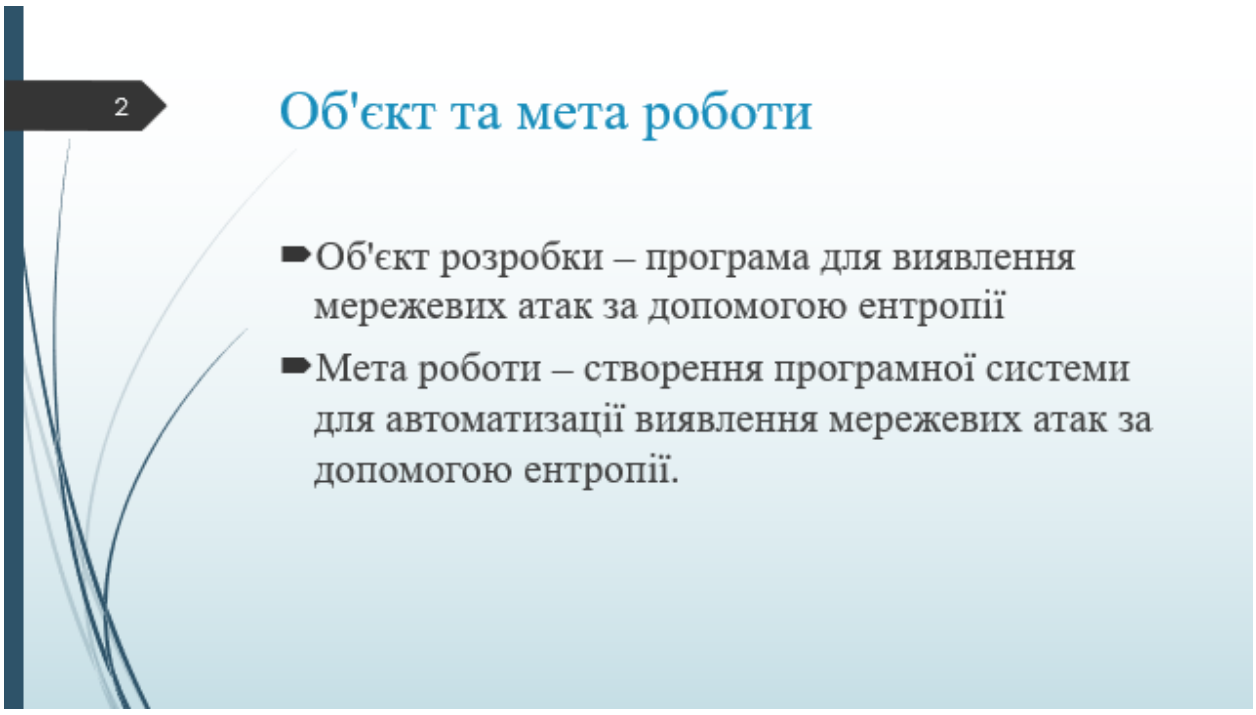


Рисунок В.2 – Слайд 2

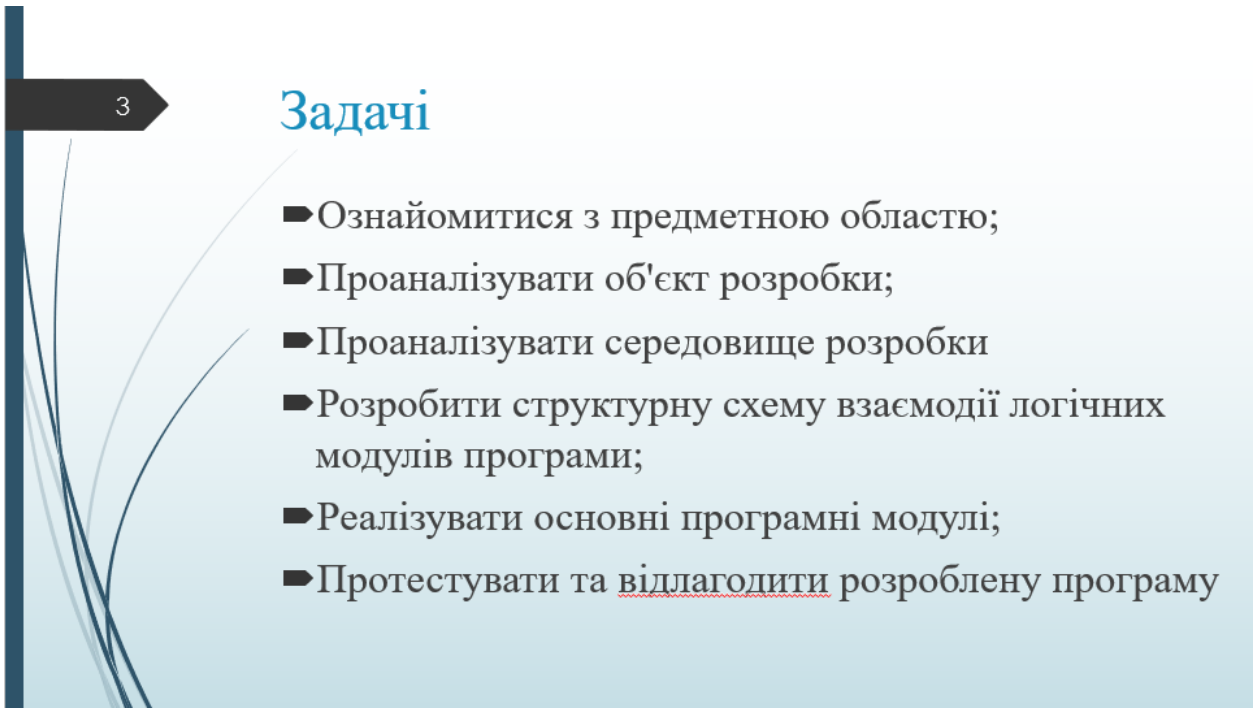


Рисунок В.3 – Слайд 3

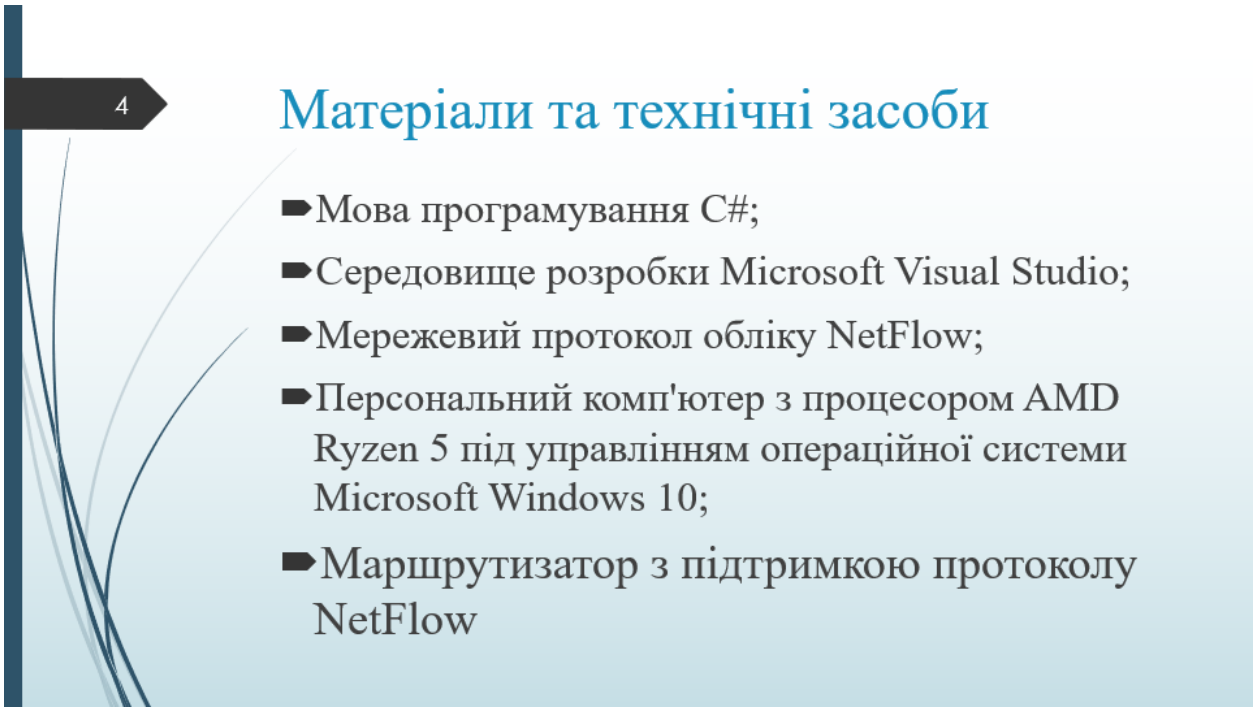


Рисунок В.4 – Слайд 4


Рисунок В.5. – Слайд 5


Рисунок В.6 – Слайд 6

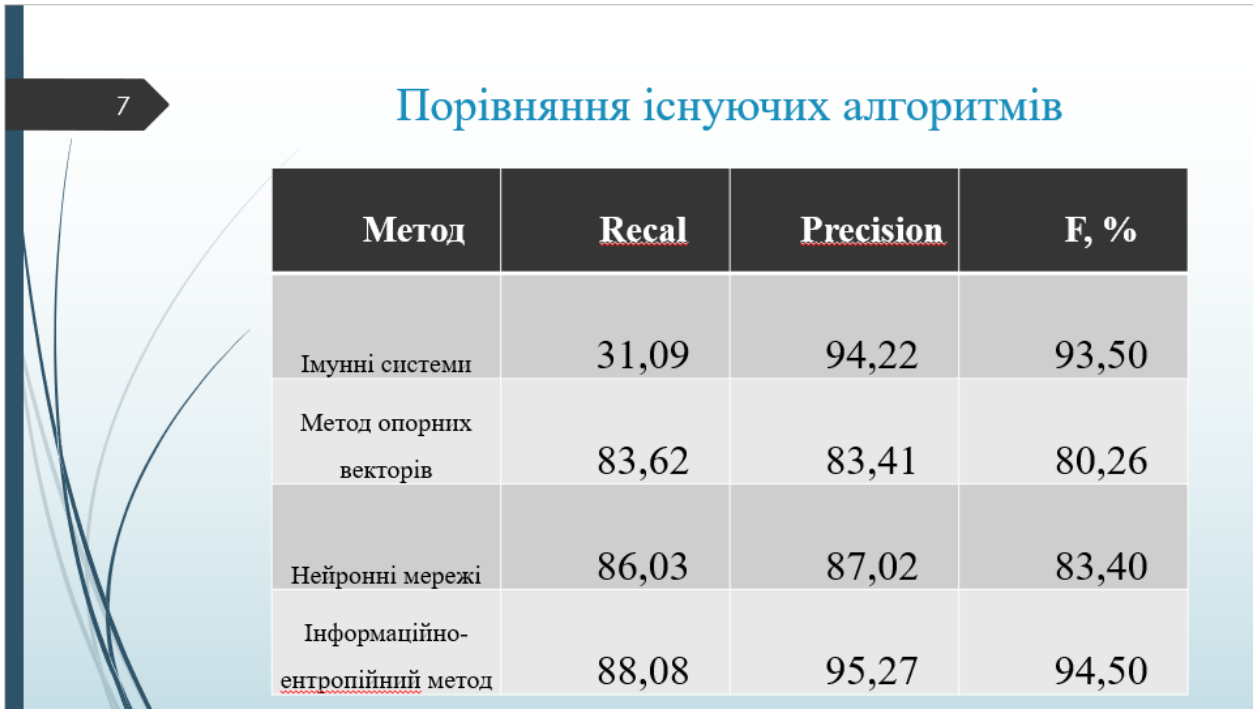


Рисунок В.7 – Слайд 7


Рисунок В.8 – Слайд 8


Рисунок В.9 – Слайд 9

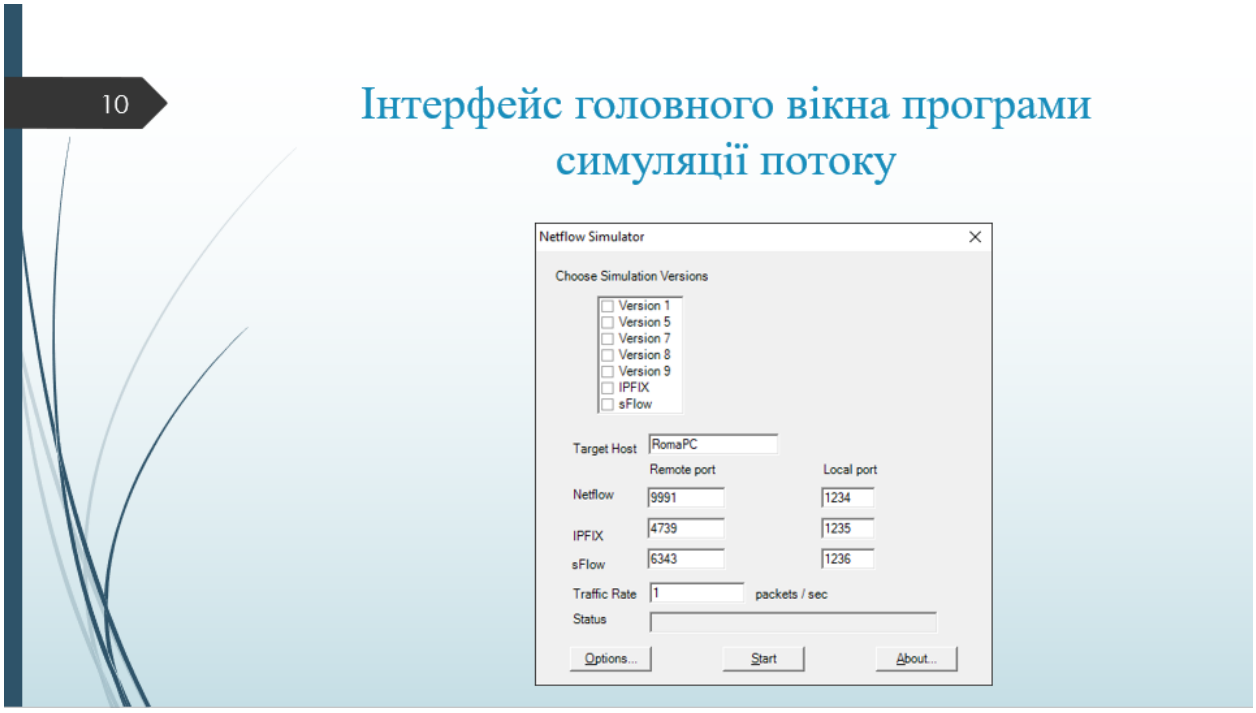


Рисунок В.10 – Слайд 10

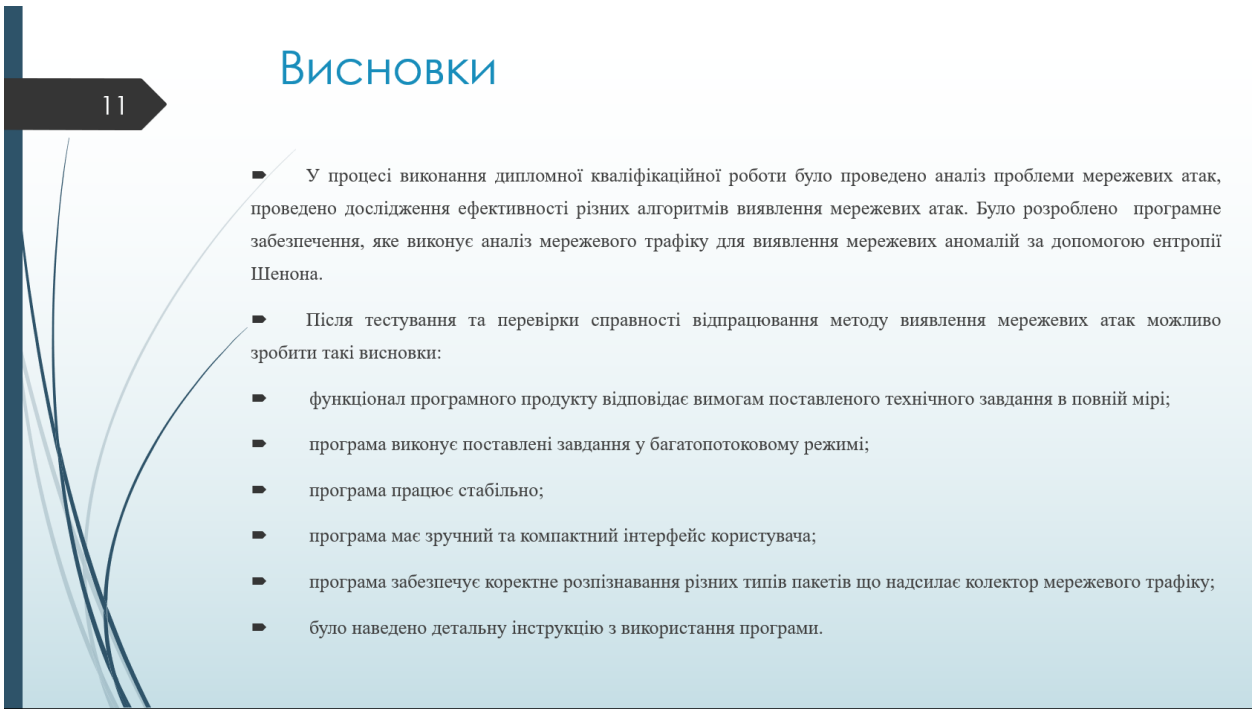


Рисунок В.11 – Слайд 11

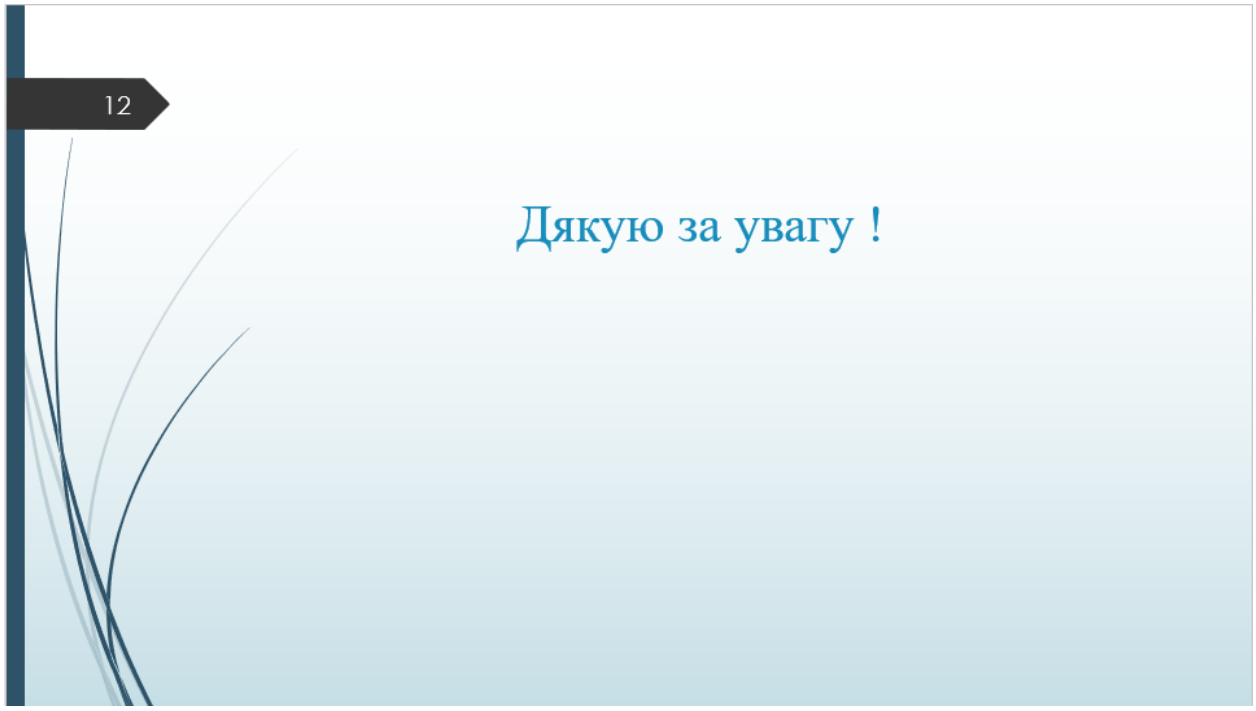


Рисунок В.12 – Слайд 12