

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)

магістр

(ступінь вищої освіти)

на тему ДОСЛІДЖЕННЯ ТА ПРОГРАМНА РЕАЛІЗАЦІЯ МЕТОДІВ  
ВИЯВЛЕННЯ АТАК У МЕРЕЖАХ МЕТОДОМ ВЕЙВЛЕТ-АНАЛІЗУ  
RESEARCH AND SOFTWARE IMPLEMENTATION OF METHODS FOR  
DETECTING NETWORK ATTACKS BY USING WAVELET ANALYSIS

Виконав: студент 2 курсу, групи КНТ-211м

Спеціальності 122 Комп'ютерні науки

(код і найменування спеціальності)

Освітня програма (спеціалізація)

Системи штучного інтелекту

Чемерис К.М.

(прізвище та ініціали)

Керівник Дубровін В.І.

(прізвище та ініціали)

Рецензент Казурова А.Є.

(прізвище та ініціали)

2022 рік

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

Інститут, факультет ФКНТ

Кафедра програмних засобів

Ступінь вищої освіти магістр

Спеціальність 122 Комп'ютерні науки

(код і найменування)

Освітня програма (спеціалізація) Системи штучного інтелекту

(назва освітньої програми (спеціалізації))

**ЗАТВЕРДЖУЮ**

Завідувач кафедри ПЗ, д.т.н, проф.

С.О. Субботін

“ ” 20\_\_ року

**З А В Д А Н Н Я**

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

Чемериса Кирила Михайловича

(прізвище, ім'я, по батькові)

1. Тема проєкту (роботи) Дослідження та програмна реалізація методів виявлення атак у мережах методом вейвлет-аналізу. Research and software implementation of methods for detecting network attacks by using wavelet analysis

керівник проєкту (роботи) Дубровін Валерій Іванович, к.т.н., професор,

(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом закладу вищої освіти від “ 4 ” листопада 2022 року № 363

2. Строк подання студентом проєкту (роботи) 1 грудня 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. Розробка архітектури програми. 3. Основні рішення щодо реалізації компонентів системи. 4. Експлуатація, тестування та експериментальне дослідження програми. Висновки. Перелік джерел посилання. Додатки.

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 |
| 1-4 Основна частина | Дубровін В.І., професор                   |                |                           |
| Нормоконтролер      | Дейнега Л.Ю., ст. викладач                |                |                           |
|                     |                                           |                |                           |
|                     |                                           |                |                           |

7. Дата видачі завдання “ 05 ” вересня 2022 року.

## КАЛЕНДАРНИЙ ПЛАН

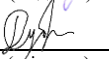
| № з/п | Назва етапів дипломного проєкту (роботи)                                           | Строк виконання етапів проєкту ( роботи ) | Примітка     |
|-------|------------------------------------------------------------------------------------|-------------------------------------------|--------------|
| 1     | Постановка завдання роботи.                                                        | 1 тиждень                                 | Завдання, ТЗ |
| 2     | Аналіз предметної області.                                                         | 2–4 тижні                                 | Розділ 1     |
| 3     | Розробка архітектури програми.                                                     | 5–6 тижні                                 | Розділ 2     |
| 4     | Основні рішення щодо реалізації компонентів системи.                               | 7–10 тижні                                | Розділ 3     |
| 5     | Експлуатація, тестування та експериментальне дослідження програмного забезпечення. | 11–12 тижні                               | Розділ 4     |
| 6     | Оформлення пояснювальної записки та документів до неї.                             | 13–14 тижні                               | Додатки      |
| 7     | Нормоконтроль та рецензування.                                                     | 15 тиждень                                |              |
| 8     | Захист роботи.                                                                     | 16 тиждень                                |              |
|       |                                                                                    |                                           |              |
|       |                                                                                    |                                           |              |
|       |                                                                                    |                                           |              |

Студент(ка)

  
( підпис )

Чемерис К.М.  
(прізвище та ініціали)

Керівник проєкту (роботи)

  
( підпис )

Дубровін В.І.  
(прізвище та ініціали)

## РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи магістра: 92 с., 9 табл., 38 рис., 2 дод., 26 джерел.

ВЕЙВЛЕТ, ВЕЙВЛЕТ-ПЕРЕТВОРЕННЯ, ШУМОУСУНЕННЯ,  
МЕРЕЖЕВИЙ ТРАФІК, МЕРЕЖЕВА АТАКА, АЛГОРИТМ МАЛЛА,  
КОЕФІЦІЄНТИ ДЕТАЛІЗАЦІЇ ТА АПРОКСИМАЦІЇ, PYTHON

Об'єкт дослідження – процес виявлення атак у мережах.

Предмет дослідження – теоретичні аспекти та практичне застосування методу вейвлет-аналізу для виявлення атак в мережах.

Мета роботи – програмна реалізація методів виявлення атак в мережах методом вейвлет-аналізу.

Матеріали, методи та технічні засоби: мова програмування Python, модуль вейвлет-перетворення PyWavelets, мережеві трафіки, персональний комп'ютер під управлінням операційної системи Windows 10 Pro 64-bit.

Результати. Розроблено методи виявлення атак у мережах з використанням вейвлет-аналізу.

Висновки. Виявлено значущі ознаки мережевого трафіку для правильного виявлення атак та розроблено програмне забезпечення для виявлення мережових атак з використанням вейвлет-аналізу та запропоновано підхід до удосконалення.

Галузь використання – виявлення мережових атак у галузі кібербезпеки.

## **ABSTRACT**

Explanatory note to the diploma qualifying work of the master: 92 pages, 9 tables, 38 figures, 2 appendixes, 26 sources.

WAVELET, WAVELET TRANSFORM, NOISE REDUCTION, NETWORK TRAFFIC, NETWORK ATTACK, MALLAT ALGORITHM, DETAIL APPROXIMATION AND COEFFICIENTS, PYTHON

Object of study is the process of detecting attacks in networks.

Subject of study are theoretical aspects and practical application of the wavelet analysis method for detecting attacks in networks.

Materials, methods, and tools: Python programming language, PyWavelets wavelet transformation module, network traffic, personal computer running Windows 10 Pro 64-bit.

Results. Methods for detecting attacks in networks using wavelet analysis have been developed.

Conclusions. Significant characteristics of network traffic for correct attack detection are identified and software for network attack detection using wavelet analysis is developed and an improvement approach is proposed.

Field of use is detection of network attacks in the field of cyber security.

## ЗМІСТ

|                                                                                          |    |
|------------------------------------------------------------------------------------------|----|
| ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ І СКОРОЧЕНЬ .....                                              | 8  |
| ВСТУП .....                                                                              | 9  |
| 1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ .....                                                        | 10 |
| 1.1 Основні поняття .....                                                                | 10 |
| 1.1.1 Мережеві аномалії та їх причини .....                                              | 10 |
| 1.1.2 Мережева атака та її види .....                                                    | 11 |
| 1.2 Існуючі методи для виявлення мережевих атак .....                                    | 14 |
| 1.3 Вейвлет-перетворення .....                                                           | 18 |
| 1.4 Вейвлет-функції .....                                                                | 20 |
| 1.5 Безперервне вейвлет-перетворення .....                                               | 24 |
| 1.6 Дискретне вейвлет-перетворення .....                                                 | 25 |
| 1.6.1 Алгоритм Малла .....                                                               | 25 |
| 1.7 Вейвлет-паketне перетворення .....                                                   | 27 |
| 1.8 Видалення шумів за допомогою вейвлетів .....                                         | 28 |
| 1.9 Аналіз існуючих способів застосування вейвлет-аналізу виявлення мережевих атак ..... | 30 |
| 1.10 Переваги та недоліки застосування ВП у виявленні мережевих атак .....               | 33 |
| 1.11 Висновки за розділом 1 .....                                                        | 34 |
| 2 РОЗРОБКА АРХІТЕКТУРИ ПРОГРАМИ .....                                                    | 35 |
| 2.1 Архітектура системи, що розробляється .....                                          | 35 |
| 2.2 Вибір мови програмування .....                                                       | 35 |
| 2.3 Вибір додаткових бібліотек .....                                                     | 38 |
| 2.4 Вибір набору даних .....                                                             | 39 |
| 2.5 Обробка даних .....                                                                  | 40 |
| 2.6 Висновки за розділом 2 .....                                                         | 42 |
| 3 ОСНОВНІ РІШЕННЯ ЩОДО РЕАЛІЗАЦІЇ КОМПОНЕНТІВ СИСТЕМИ ..                                 | 43 |
| 3.2 Попередня обробка мережного трафіку .....                                            | 43 |

|                                                                              |    |
|------------------------------------------------------------------------------|----|
| 3.3 Розробка компонентів.....                                                | 47 |
| 3.3.1 Розробка підсистеми обробки даних .....                                | 47 |
| 3.3.2 Розробка методів вейвлет-аналізу.....                                  | 48 |
| 3.3.3 Розробка графічних елементів програми.....                             | 56 |
| 3.4 Висновки за розділом 3 .....                                             | 57 |
| 4 ЕКСПЛУАТАЦІЯ, ТЕСТУВАННЯ ТА ЕКСПЕРИМЕНТАЛЬНЕ<br>ДОСЛІДЖЕННЯ ПРОГРАМИ ..... | 58 |
| 4.1 Експлуатація програмного забезпечення .....                              | 58 |
| 4.1.1 Призначення програми .....                                             | 58 |
| 4.1.2 Можливості програми .....                                              | 58 |
| 4.1.3 Мінімальний склад технічних та програмних засобів .....                | 59 |
| 4.1.4 Виконання програми.....                                                | 59 |
| 4.1.5 Повідомлення користувачу .....                                         | 63 |
| 4.2 Тестування функціональності програми .....                               | 63 |
| 4.3 Експериментальне дослідження .....                                       | 65 |
| 4.4 Висновки за розділом 4 .....                                             | 71 |
| ВИСНОВКИ.....                                                                | 72 |
| ПЕРЕЛІК ПОСИЛАНЬ.....                                                        | 74 |
| ДОДАТОК А Текст програми.....                                                | 77 |
| ДОДАТОК Б Слайди презентації.....                                            | 85 |

## ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ І СКОРОЧЕНЬ

БВП – безперервне вейвлет-перетворення;

ВП – вейвлет-перетворення;

ВПП – вейвлет пакетне перетворення;

ДВП – дискретне вейвлет-перетворення;

ІС – інформаційна система;

ІТ – інформаційні технології;

МА – мережева атака;

МТ – мережевий трафік;

ШВП – швидке вейвлет-перетворення.

## ВСТУП

Інтернет є місцем найвищої ділової активності в даний час. Велика кількість підприємств та організацій по всьому світу використовують комп'ютерні обчислювальні системи для управління різними виробничими процесами та персоналом, розподілу ресурсів та підключення користувачів. Як відомо, зростання інформаційних технологій (ІТ) викликало стрімке зростання комп'ютерної злочинності, зокрема, зросла кількість загроз, пов'язаних з несанкціонованим доступом із глобальної мережі до внутрішньої мережі підприємств [1].

Згідно зі статистиками, кількість атак поступово збільшується з кожним роком. Найбільш небезпечними та популярними типами атак є шкідливі та DDoS-атаки. До 2022 року очікується, що кількість DDoS-атак досягає 14.5 млн. І більше 90% DDoS-атаки в третьому кварталі або в 2020 році тривали менше чотирьох годин, це пояснюється, що DDoS-атаки стають менш тривалими, але більш частими та інтенсивними. Найчастіше DDoS-атакам піддаються держустанови, промисловість, медицина, фінансова галузь, сфера науки і освіти [2].

Атктуальною темою є виявлення мережевих атак (МА) та аномалій мережевого трафіку з метою попередження вторгнення та зниження ризиків від подібних порушень. Як відомо заздалегідь, проблеми з мережевою безпекою можуть виникати не тільки через атаки зловмисника, а й внаслідок відмови програмно-апаратного забезпечення, помилок користувачів, порушення функціонування підсистем тощо.

В роботі досліджуються способи виявлення мережевих аномалій та атак за допомогою вейвлет-аналізу, вивчаються та вибираються значні ознаки мережевого трафіку, розробляються та застосовуються методи вейвлет-аналізу, а також аналізуються результати та описуються шляхи удосконалення системи.

# 1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

## 1.1 Основні поняття

### 1.1.1 Мережеві аномалії та їх причини

Аномалія в сетях – це відхилення від норми або загальної закономірності в мережевому трафіку. Зазвичай аномалії видаються як викиди, шуми, незвичайності, відхилення чи винятки [3].

Відомі мережеві аномалії настільки різноманітні, що єдиної класифікації вони не піддаються. Так, існує поділ на активні і пасивні, зовнішні і внутрішні, свідомі й несвідомі аномалії і т.д. Однак дані підходи не відображають всіх характеристик досліджуваного явища і є обмеженими. Тому пропонується класифікація аномалій у мережах з точки зору об'єкта впливу – інформаційної системи (ІС), що включає програмно-апаратний комплекс і мережеву інфраструктуру [4].

Можна поділити мережеву аномалію на основні групи: програмно-апаратні проблеми, порушення безпеки, а також невідомі [4] (рис. 1.1).

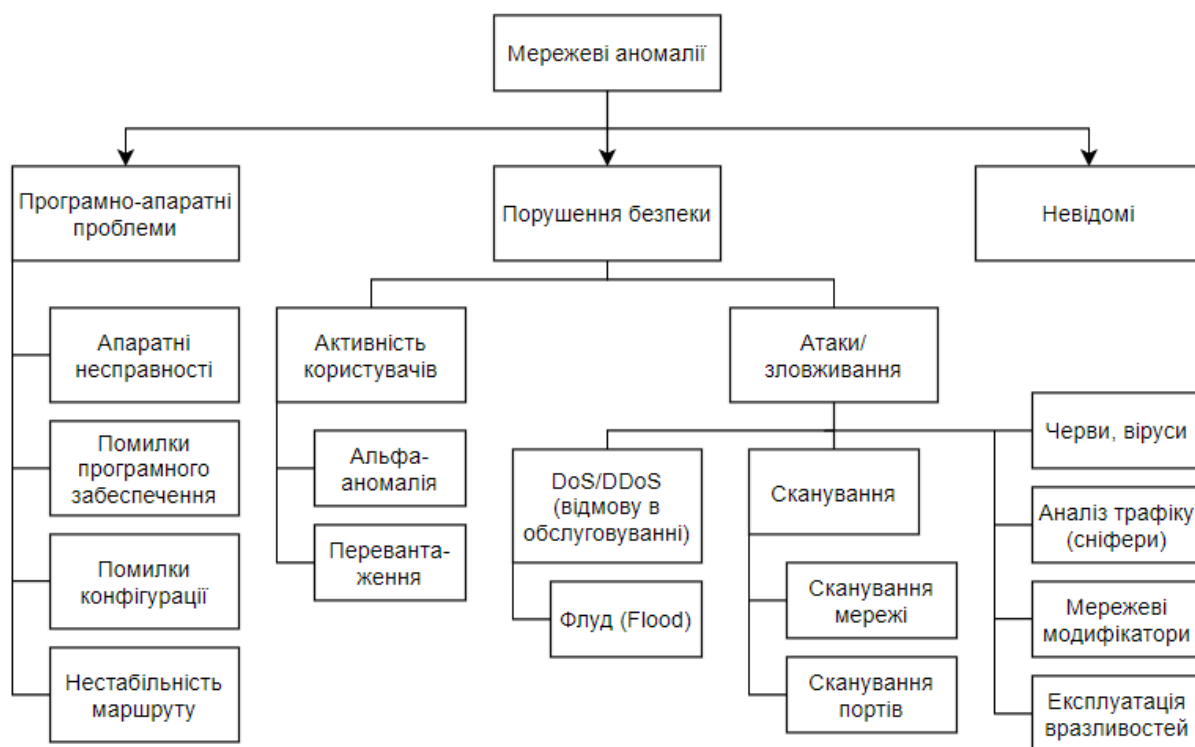


Рисунок 1.1 – Класифікація аномалій у мережах

До програмно-апаратних проблем відносяться: апаратні несправності, помилки конфігурації, помилки програмного забезпечення та нестабільність маршруту.

До порушення безпеки відносяться: активність користувачів та атаки/зловживання. До активності користувачів відносяться альфа-аномалія та перевантаження. До атак/зловживанням відносяться: сканування, атаки з метою відмови від обслуговування (DoS/DDoS), вірусна активність, поширення програмних "черв'яків", аналізатори трафіку (сніфери), мережеві модифікатори і експлуатація вразливостей.

Помилки програми можуть спричинити за собою переклад в нештатний режим з наступним припиненням надання сервісів.

Помилки зміни порушують загальну працездатність.

Апаратні несправності можуть викликати повний вихід з ладу окремих компонентів системи.

Сканування (network scan) використовується для аналізу мережевої топології та виявлення доступних сервісів для атаки. Під час сканування відбувається спроба з'єднатися з сервісами мережі шляхом звернення до вибраного порту.

Аналізатори трафіку (сніфери) призначені для перехоплення і аналізу мережевого трафіку.

Термін вразливість (vulnerability) у комп'ютерній безпеці використовується для позначення незахищених від несанкціонованого впливу компонента ІС.

Вірусна активність є результатом спроб поширення комп'ютерних вірусів та черв'яків, використовуючи мережеві ресурси [4].

### **1.1.2 Мережева атака та їх види**

Мережева атака – набір зловмисних дій, який має на меті зробити якісь дії віддалено з комп'ютером або сервером [5].

За принципом дії мережеві атаки можуть бути активними та пасивними. Активні атаки використовуються не тільки для отримання інформації з віддаленого комп'ютера, але і для порушення працездатності комп'ютера. Активні атаки досить помітні, оскільки вони залишають сліди, але такі атаки можуть залишитися непоміченими, якщо фахівці захисту мережі не будуть ретельно розбиратися в наслідках таких атак.

Пасивні атаки використовуються для отримання конфіденційної інформації віддаленого комп'ютера. Такі атаки помітити практично неможливо через відсутність прямого впливу [5].

Існує кілька різновидів мережевих атак, в тому числі наступні.

DoS (Denial of Service — Відмова в обслуговуванні) – атака, яка робить систему (сайт, програма тощо) недоступною шляхом надсилання великої кількості запитів, що перевищують пропускну спроможність мережі.

DDoS (Distributed Denial of Service — розподілена відмова в обслуговуванні) – підтип DoS атаки, який має ту ж мету, що і DoS, але виробленої не з одного комп'ютера, а з кількох комп'ютерів у мережі [6].

Атаки типу DoS і DDoS – одні з найсерйозніших атак з точки зору їх ліквідації. Вони відносяться до активної атаки. Для виконання цих атак не потрібно особливих зусиль, однак через легкість виконання та значного потенційного збитку атаки DoS і DDoS вимагають особливої уваги з боку фахівців з безпеки [6].

DoS та DDoS атаки відрізняються від більшості інших атак тим, що вони призначені для порушення нормального функціонування системи, а не для отримання будь-якої інформації про мережу [6].

Для проведення DDoS-атаки зловмисник захоплює контроль над кількома комп'ютерами, потім синхронізує їх роботу і направляє сумарний потік запитів, що посилився, з цих комп'ютерів на жертву (сервер), на якій цілиться (рис. 1.2) [6].

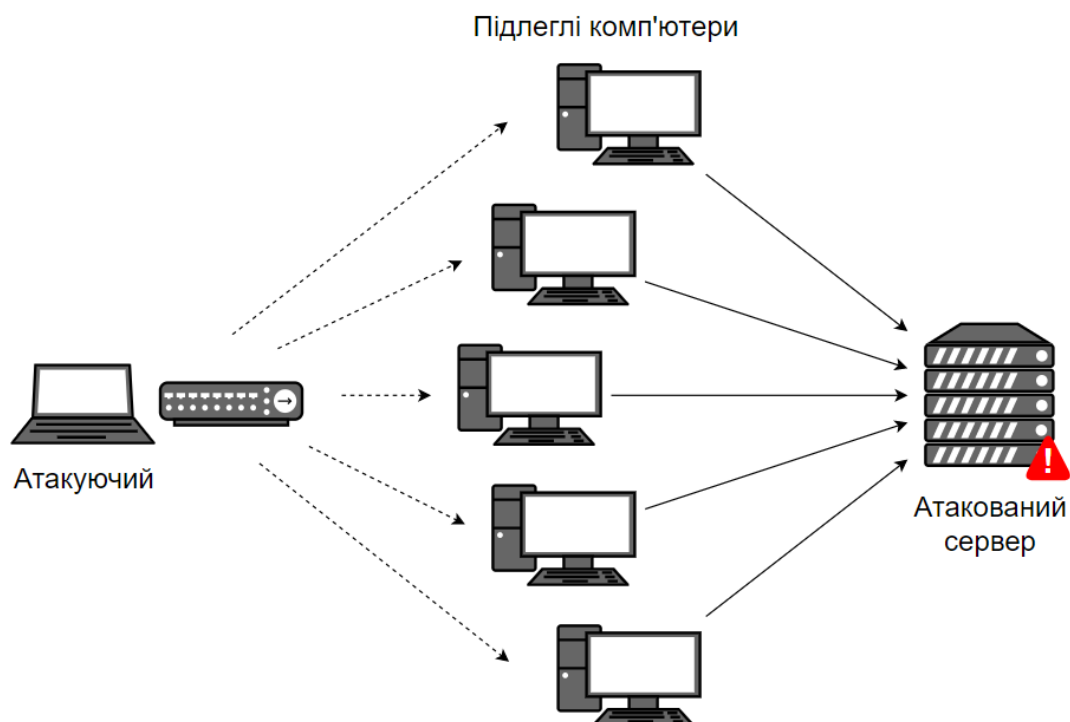


Рисунок 1.2 – Схема DDoS-атаки

TCP SYN Flood – один із мережєвих атак типу DDoS, який полягає у відправці великої кількості SYN-запитів (запитів на підключення по протоколу TCP) в досить короткий термін.

Коли клієнт і сервер встановлюють звичайне з'єднання TCP, обмін виглядає наступним чином: спочатку клієнт відправляє SYN повідомлення на сервер, потім сервер відправляє SYN-ACK повідомлення назад клієнту, потім клієнт відповідає повідомленням ACK і з'єднання встановлено.

При атаці TCP SYN Flood зловмисники інтенсивно відправляють серверу велику кількість SYN-пакетів, часто використовуючи спуфінг (з підробленими IP-адресами). Зрештою, у разі переповнення з'єднання сервера сервер може вийти з ладу або аварійно завершить роботу [4], [7].

Спуфінг (IP spoofing) – підміна IP-адрес, яка необхідна зловмисникам для знищення всіх слід своєї участі в атаці. Часто спуфінг використовують при проведенні DDoS атак та інших атак, щоб стати більш анонімним. Принцип полягає в наступному: для приховання місця знаходження джерела пакетів зловмисник змінює значення поля адреси відправника у заголовках пакетів.

Однак система може автоматично створити адресу відправника, тому зловмисник змінює її так, щоб система давала можливість відправити з комп'ютера пакети з будь-якими IP-адресами [8].

## **1.2 Існуючі методи для виявлення мережових атак**

На даний момент для виявлення мережових атак існує безліч методів, які активно розвиваються. У роботах [9], [10] було проведено аналіз великого переліку робіт та зібрано список відомих методів виявлення атак для синтезу загальної класифікації методів виявлення атак у мережах, що зображена на рис. 1.3.

Методи виявлення атак можна розділити на чотири великі групи: методи поведінки, методи на основі знань, методи машинного навчання та методи обчислювального інтелекту. Методи машинного навчання та методи обчислювального інтелекту належать до методів інтелектуального аналізу даних (ІАД).

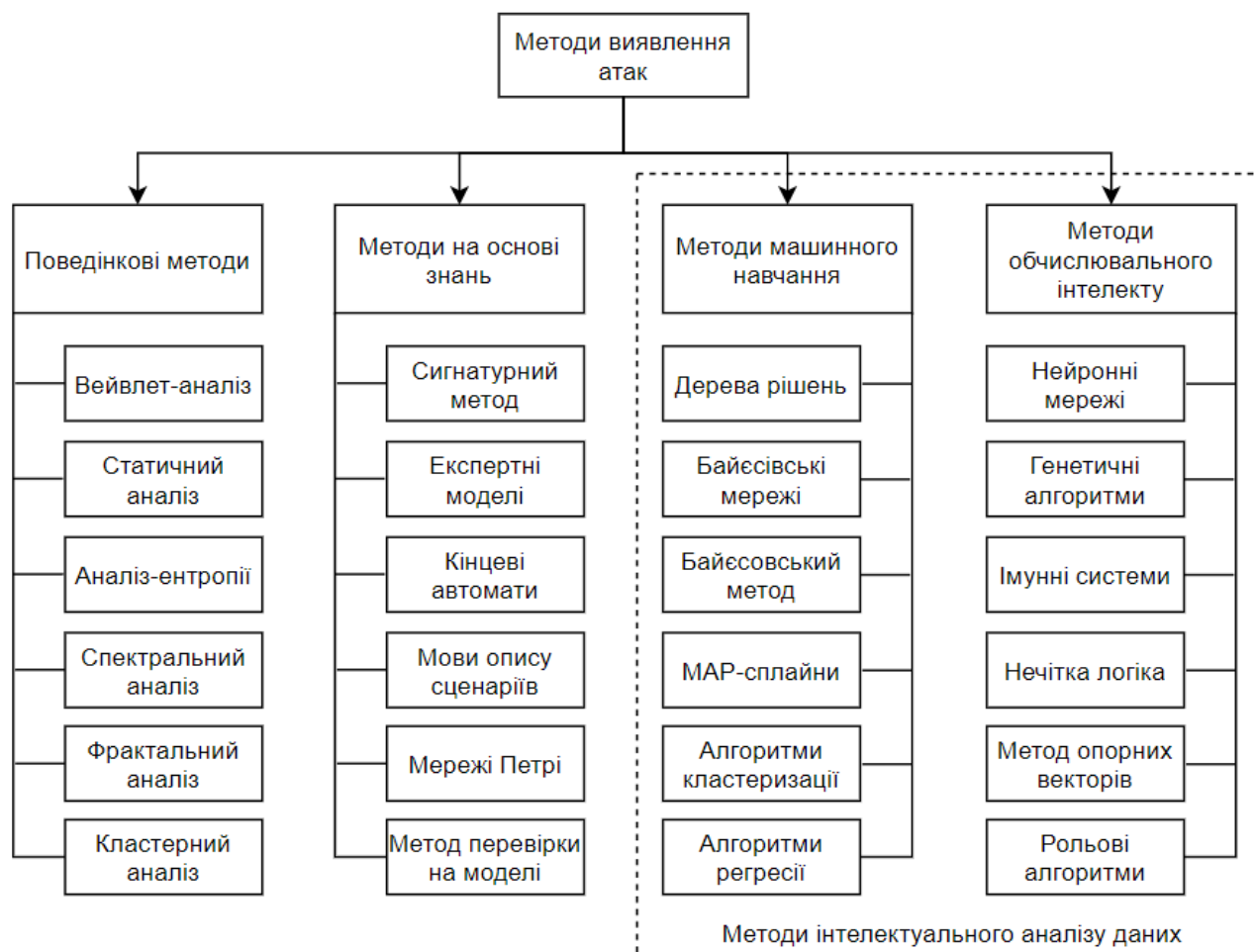


Рисунок 1.3 – Класифікація методів виявлення атак

Різні методи можуть застосовуватись для різних цілей виявлення атак. Далі розглянуто групи методів [9],[10].

**Поведінкові методи.** Ці методи засновані на використанні інформації про нормальну поведінку мережі та її порівняння з деякими параметрами спостережуваної поведінки мережі. Системи, які використовують даний підхід, порівнюють поточні показники мережного трафіку з профілем нормального мережевого трафіку, а наявність значних відхилень сприймається як можливе наявність атаки. Для проведення даної роботи спочатку необхідно провести етап попереднього налаштування, під час якого система набирає дослідів для створення моделі нормальної поведінки. Це може тривати кілька тижнів, а іноді й кілька місяців.

До поведінкових методів належать:

– вейвлет-аналіз. Цей підхід створює коефіцієнти шляхом розкладання вхідного сигналу з використанням базисних функцій. Як сигнал може розглядатися кількість байтів, пакетів або потоків за секунду або за хвилину. Вейвлет-перетворення дозволяє виділити більш корисну інформацію у вигляді сигналу з високими амплітудами, які найчастіше є аномаліями, та ігнорувати менш корисну інформацію з низькими амплітудами, які є шумами;

- статистичний аналіз;
- аналіз ентропії;
- спектральний аналіз;
- фрактальний аналіз;
- кластерний аналіз.

Методи з урахуванням знань. Ці методи виявляють мережні аномалії на основі закладеного механізму пошуку. Як процедуру пошуку можуть застосовуватися зіставлення за зразком, апарат регулярних виразів, аналіз переходу станів тощо. Ці методи мають таку назву, оскільки працюють з базою знань, до якої включені описи вже відомих атак.

До методів на основі знань відносяться:

- сигнатурний метод;
- експертні моделі;
- кінцеві автомати;
- мови опису сценаріїв;
- мережі Петрі;
- метод перевірки на моделі.

Методи машинного навчання. Такі методи також застосовуються для виявлення атак у мережах. Для цих методів, як і для методів обчислювального інтелекту як вихідні дані для навчання часто використовують шаблони як нормальної, так і аномальної поведінки мережевого трафіку.

До методів машинного навчання віднесено такі методи виявлення атак:

- дерева рішень;

- байєсівські мережі. Є одним із найчастіше використовуваних підходів для виявлення вторгнень;

- байєсовський метод;
- МАР-сплайни;
- алгоритми кластеризації;
- алгоритми регресії.

Методи обчислювального інтелекту. За допомогою таких методів можна вирішити проблему з точністю ідентифікації атаки при її поділі в часі або при проведенні кількома різними злоумисниками [9], [10].

До методів обчислювального інтелекту належать [11]:

- нейронні мережі. Мають здатність навчання за зразком та узагальнення з неповних та зашумлених даних. У процесі навчання зазвичай відбувається налаштування коефіцієнтів, асоційованих із синаптичними вагами. Для виявлення атак можуть бути використані різні моделі нейронних мереж, а саме – багатошарові мережі прямого поширення, рекурентні мережі та радіально-базисні мережі, а також карт, що організуються;

- генетичні алгоритми. Засновані на імітації біологічних принципів природного відбору та призначені для вирішення задач оптимізації;

- імунні системи;
- нечітка логіка. У деяких роботах виявлення мережевих атак нечітка логіка застосовується разом із генетичним алгоритмом;

- метод опорних векторів;
- рольові алгоритми.

Сьогодні існують також ще інші основні поведінкові алгоритми виявлення мережевих атак [1]:

- алгоритм Бродського – Дарховського. Використовує метод перевірки збігу або розходження середнього значення в рухомих вікнах;
- алгоритм на основі суми квадратів вейвлет-коефіцієнтів;

– алгоритм на основі максимуму квадратів вейвлет-коефіцієнтів, похідний від попереднього, але менш ефективний.

Вейвлет-перетворення є один з найбільш перспективних технологій пошуку потрібної інформації в базах даних мережевої безпеки.

В таблиці 1.1 наведено порівняння методів за деякими критеріями.

Таблиця 1.1 – Порівняння методів за деякими критеріями

| Критерій                          | Поведінкові методи    | Методи на основі знань | Методи машинного навчання | Методи обчислювального інтелекту |
|-----------------------------------|-----------------------|------------------------|---------------------------|----------------------------------|
| Виявлені цілі                     | Зловживання, Аномалії | Зловживання            | Зловживання, Аномалії     | Зловживання, Аномалії            |
| Простота реалізації               | —                     | ±                      | —                         | ±                                |
| Низькі витрати пам'яті            | ±                     | ±                      | +                         | +                                |
| Виявлення відомих атак/аномалій   | +                     | +                      | +                         | +                                |
| Виявлення невідомих атак/аномалій | +                     | —                      | ±                         | ±                                |
| Точність виявлення                | +                     | +                      | ±                         | ±                                |

### 1.3 Вейвлет-перетворення

Вейвлет-перетворення (ВП) є великою перевагою над іншими методами аналізу сигналів. ВП знайшло широке застосування в різних завданнях: аналіз сигналів, шифрування та дешифрування інформації, розпізнавання образів та синтезу зображень тощо [1], [12].

Згідно з аналізами існуючих методів вирішення задач з виявлення мережевих атак ВП широко використовується в системах виявлення мережевих

атак, завдяки йому частотно-часової властивості, що дозволяє розкласти сигнал на кілька частотних компонентів [1], [12].

Вейвлет – хвилеподібне коливання з амплітудою, яка починається з нуля і збільшується або зменшується, а потім повертається до нуля кілька разів, тому вейвлети називаються малими хвилями. Вейвлети можуть використовуватися для отримання інформації з різних видів даних, таких як звуковими, мережевими сигналами, зображеннями і т.д [1], [12].

Вейвлет-перетворення часто порівнюють із перетворенням Фур'є. На відміну від перетворення Фур'є, яке локалізує лише за частотою, вейвлети можуть одночасно локалізувати за часом і частотою. Вейвлети мають гарну частотно-часову адаптацію і вони можуть бути зручним інструментом для дослідження частотних характеристик сигналу. Перевага використання вейвлет-перетворення в тому, що характерні деталі (у нашому випадку мережеві аномалії), які можуть залишатися непоміченими за одного масштабу, легко можуть бути виявлені на іншому. Загалом, вейвлет-перетворення виражено наступним рівнянням [1], [4],[ 12]:

$$F(a, b) = \int_{-\infty}^{\infty} f(x) \psi_{(a,b)}^*(x) dx, \quad (1.1)$$

де  $*$  – символ комплексної пов'язаності та функція  $\psi$  – деяка функція. Функція може бути обрана довільним чином, але вона повинна задовольняти певним правилам.

Існують ортогональні вейвлети для дискретного вейвлет-перетворення і неортогональні вейвлети для безперервного, які можна використовувати.

Розрізняють дискретне вейвлет-перетворення (ДВП) та безперервне вейвлет-перетворення (БВП). У ДВП використовуються ортогональні вейвлети, і досягається більш швидке перетворення порівняно з БВП, в якому використовуються неортогональні вейвлети і допомагає бачити результат більш чіткому вигляді для людського сприйняття [1], [4], [12].

ДВП найефективніше для очищення сигналів від шумів, стиснення сигналів та зображень, а також виявлення мережевих атак. ДВП часто використовуються для виявлення мережевих атак, оскільки на кожному рівні швидко витягуються коефіцієнти деталізації та апроксимації, які потім можуть бути використані для автоматичного виявлення атак. Автоматичне виявлення мережевих атак є важливим і складним завданням [13].

А БВП в основному використовується для виявлення різких змін у сигналі та дослідженні випадкових сигналів.

#### 1.4 Вейвлет-функції

Існує сімейств вейвлет-функцій на даний момент:

- Хаара;
- Добеші;
- Койфлети;
- Сімлети;
- Шенона;
- Гаусса;
- біортогональні та інші [12].

Кожне з яких має свої переваги для вирішення завдань різних типів. Наприклад, вейвлети Гауссова типу, «мексиканського капелюха» та Морле. До грубих вейвлетів відносяться вейвлети Гауссова типу, «мексиканського капелюха» та Морле. Вони мають наступні властивості: неортогональний аналіз, відсутня функція  $\varphi(x)$ , функція  $\psi(x)$  не має компактного носія, не гарантована можливість реконструкції, симетричність, можлива безперервна декомпозиція та ін [12].

Вейвлети Мейєра належать до нескінченних регулярних вейвлетів. Вони мають наступні властивості: є функція  $\varphi(x)$ , аналіз ортогональний, функції  $\varphi(x)$  і  $\psi(x)$  не визначені явно, функції  $\varphi(x)$  і  $\psi(x)$  не мають компактного носія, не

підтримується швидкий алгоритм перетворень, можливі методи аналізу (безперервні перетворення, дискретні перетворення, але без фільтрів з кінцевою імпульсною характеристикою) [12].

Вейвлети Добеші, Койфлета та Сімлета відносяться до вейвлетів ортогонального типу. Їх основні властивості: функція  $\varphi(x)$  наявна, аналіз ортогональний, функції  $\varphi(x)$  і  $\psi(x)$  мають компактний носій, можливі безперервні перетворення і дискретні перетворення, певними труднощами виступає їх недостатня періодичність [12].

Ортогональні вейвлети володіють наступною специфікою: вейвлети Добеші несиметричні, вейвлети Симлети близькі до симетричних, вейвлети Койфлети несиметричні, мають функції  $\varphi(x)$  і  $\psi(x)$  [12].

В-сплайнові біортогональні вейвлети з компактним носієм відносяться до біортогональних вейвлетів. Їхня функція  $\varphi(x)$  наявна та аналіз відноситься до біортогонального типу. І можна використовувати перервне перетворення і дискретне перетворення. Але відсутні ортогональності [12].

Вейвлети Морлі, Гаусса та Шеннона, а також частотні В-сплайнові вейвлети відносяться до комплексних вейвлетів. Їхня функція  $\varphi(x)$  відсутня та аналіз не ортогональний. Функція  $\psi(x)$  не має компактного носія. Однак швидкий алгоритм і реконструкція неможливі [12].

При збільшенні числа коефіцієнтів вейвлета функції стають більш гладкими і це може полегшити виявлення мережевої атаки. Для виявлення атак мережного трафіку необхідно отримувати потрібну інформацію з масиву цифрових даних. Для вибору системи базисних вейвлетів було проаналізовано різні вейвлети з компактним носієм: Хаара, Добеші, Койфлети, і навіть Симлети, які найбільш якісно виділяють локальні особливості сигналів [12].

Вейвлет Хаара — один з перших та найбільш простих вейвлетів. Вони ортогональні та мають компактний носієм, добре локалізовані в просторі, але не є гладкими. Пізніше Інґрід Добеші почала розвивати теорію та запропонувала використовувати свої функції, які називаються вейвлетами Добеші [12].

Вейвлети Симлет ортогональні та майже симетричні. Ця властивість забезпечує мінімальне фазове спотворення. Вейвлети аналогічні вейвлету Добеші, крім кращої симетрії [14].

Вейвлет Койфлет є дискретним вейвлетом, розробленим Інґрід Добеші на прохання Рональда Койфмана, щоб мати масштабуючі функції зі зникаючими моментами. Вейвлет майже симетричний, їх вейвлет-функції мають  $N/3$  моментів, що зникають, і функції масштабування  $N/3-1$ , і використовувався в багатьох додатках з використанням Кальдерону-Оператори Зигмунда. Вейвлети Койфлет більш симетричні ніж Добеші, наприклад, що дає відмінну апроксимацію щодо симетричних сигналів [15].

На рисунках 1.4 – 1.9 відображені різні вейвлети та їх масштабуючі функції.

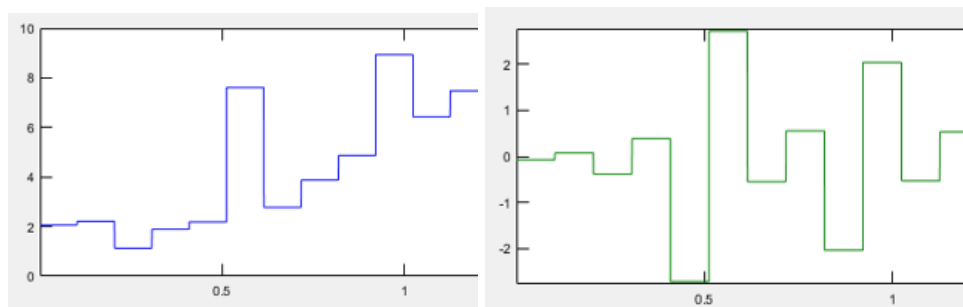


Рисунок 1.4 – Вейвлет Хаара (зліва) та його масштабуюча функція (зправа)

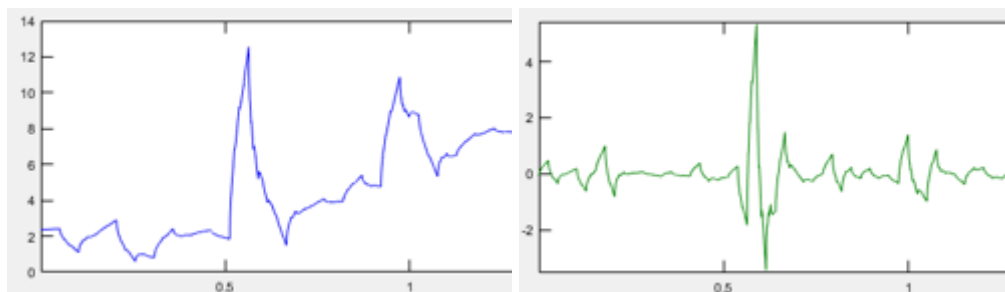


Рисунок 1.5 – Вейвлет Добеші-2 (зліва) та його масштабуюча функція (зправа)

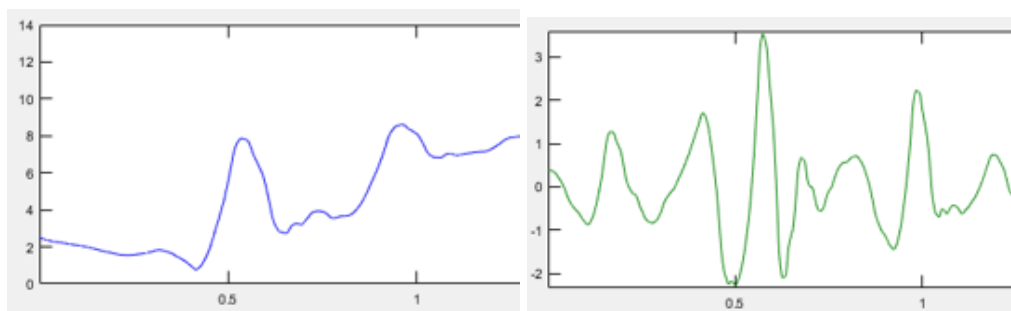


Рисунок 1.6 – Вейвлет Добеші-4 (зліва) та його масштабуюча функція (зправа)

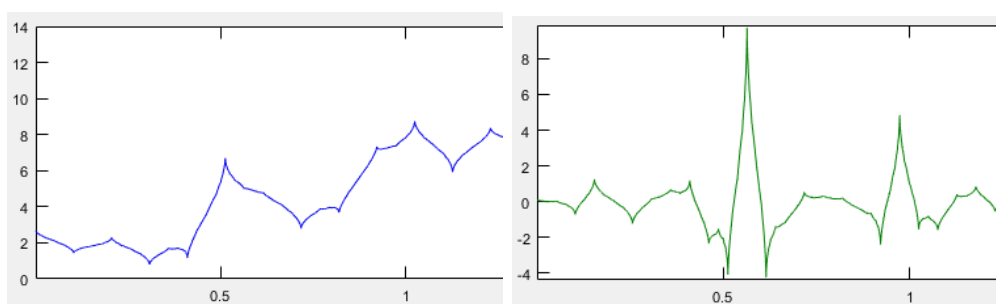


Рисунок 1.7 – Вейвлет Койфлет-1 (зліва) та його масштабуюча функція (зправа)

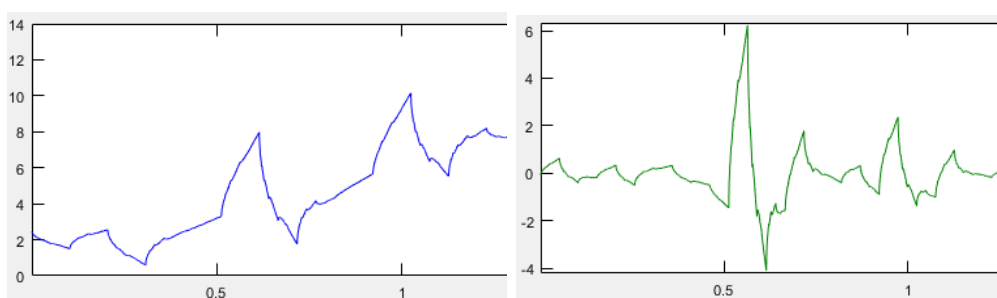


Рисунок 1.8 – Вейвлет Симплет-2 (зліва) та його масштабуюча функція (зправа)

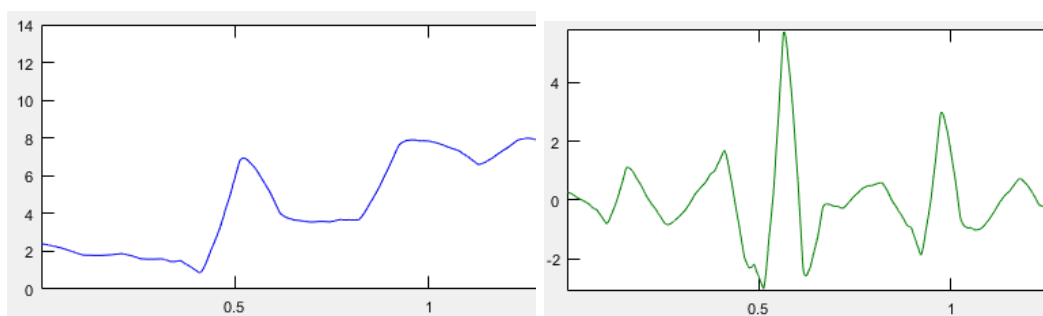


Рисунок 1.9 – Вейвлет Симплет-4 (зліва) та його масштабуюча функція (зправа)

## 1.5 Безперервне вейвлет-перетворення

Для одновимірних даних виходить зображення в площині час-частота, тому БВП повертає двовірний масив, який на один вимір більше вхідних даних. Після перетворення можна легко простежити зміну частот сигналу в часі та порівнювати цей спектр із спектрами з інших сигналів [4], [13].

Для одновимірних даних виходить зображення в площині час-частота, тому БВП повертає двовірний масив, який на один вимір більше вхідних даних. Після перетворення можна легко простежити зміну частот сигналу в часі та порівнювати цей спектр із спектрами з інших сигналів [4], [13].

Для дискретних часових послідовностей також можна використовувати дане перетворення, але за умови, що найменші переноси вейвлета повинні дорівнювати дискретизації даних. Це іноді називається безперервним вейвлет-перетворенням дискретного часу [4], [13].

Вибір вейвлета тут є важливим завданням, оскільки вирішення результату за часом і частотою залежить від вейвлета. Точніше низькі частоти мають хорошу роздільну здатність за частотами і погане за часом, а високі частоти мають погану роздільну здатність за частотами, але добре за часом. Наприклад, вейвлет Морле дає хороший дозвіл за частотами, а вейвлет Похідна Гауссіана дає хороший дозвіл за часом [4], [13].

На рисунку 1.10 графічно представлено безперервне вейвлет-перетворення. Графічно результат виконання безперервного вейвлет-перетворення зображується у вигляді спектра, який може бути представлений як у вигляді плоскої картинки, так і у вигляді об'ємної поверхні. Наявність чітко помітних ізоліній свідчить про наявність мережових атак, зокрема DDoS.

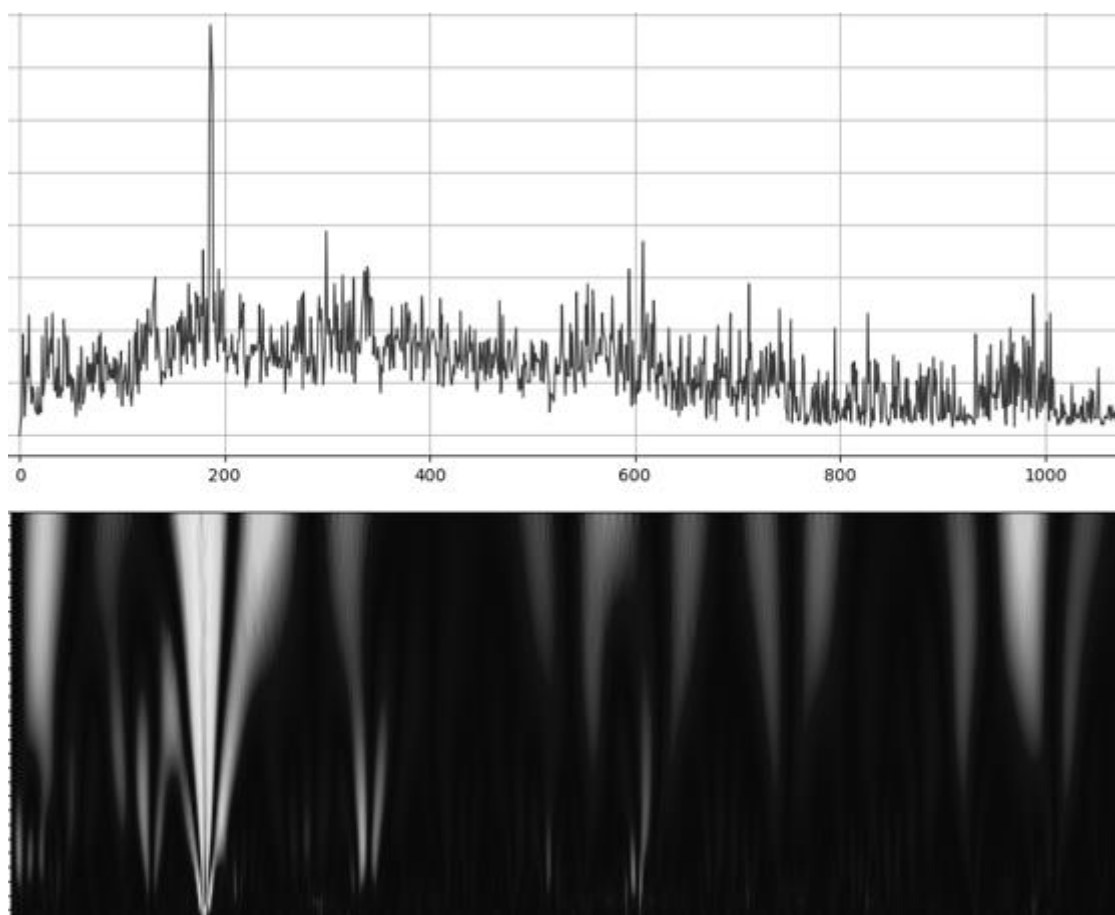


Рисунок 1.10 – Безперервне вейвлет-перетворення

## 1.6 Дискретне вейвлет-перетворення

Дискретне вейвлет-преобразование – ВП, в якому вейвлети представлені дискретними сигналами. Подається подібний сигнал на таке ж або менше число коефіцієнтів вейвлет-спектра, як і кількість точок даних сигналу. Даний вейвлет-спектр достатньо хороший для обробки та зйомки сигналів, оскільки вихідні дані не будуть отримані. ДВП часто використовується для виявлення мережових аномалій [4], [13].

### 1.6.1 Алгоритм Малла

Існує кілька видів реалізації алгоритму дискретного вейвлет-перетворення. Швидке вейвлет-перетворення або алгоритм Малла

(пірамідальний) є найбільш швидким та часто використовуваним дискретним вейвлет-перетворенням. У цьому алгоритмі два фільтри – згладжуючий і незгладжуючий складаються з коефіцієнтів вейвлету та ці фільтри рекуррентно застосовуються для отримання даних для всіх доступних масштабів [3], [4].

Гідність цього алгоритму полягає в простоті та високій швидкості: і на розкладання, і на відновлення потрібно близько  $gL$  операцій, де  $g$  – число коефіцієнтів, а  $L$  – довжина сигналу.

В якості значень  $X$  приймаються вихідні значення одновимірного числового ряду. Звідси, явний вид вейвлета потрібно тільки для розрахунку коефіцієнтів  $c$  і  $d$ , а при власне швидкому ВП використовуються вже отримані коефіцієнти на відповідному рівні деталізації. Реалізація швидкого ВП одновимірного числового ряду на основі пірамідального алгоритму обчислення вейвлет-коефіцієнтів (алгоритм Малла), наведений на рис. 1.11 [3], [4].

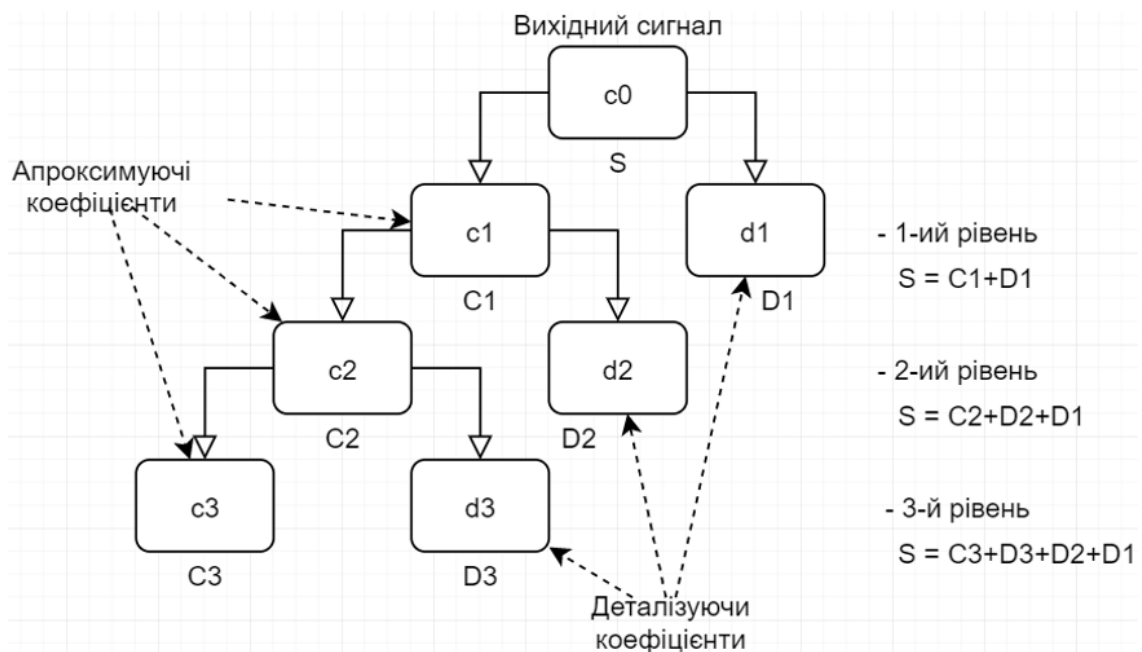


Рисунок 1.11 – Алгоритм Малла

ВП при послідовному збільшенні значень  $m$  призводить до форми швидких ітераційних обчислень вейвлет-коефіцієнтів виду:

$$\begin{aligned}c_{m+1,k} &= \sum_n h_n c_{m,2k+n}, \\d_{m+1,k} &= \sum_n g_n c_{m,2k+n},\end{aligned}\tag{1.2}$$

де  $c$  і  $d$  – коефіцієнти апроксимації і деталізації відповідно.

Принцип операції алгоритму Малла полягає в наступному [6]:

Якщо використовується повний набір даних  $S = 2^N$  і довжина сигналу дорівнює  $L$ , спочатку розраховуються дані  $S/2$  для масштабу  $L/2^{N-1}$ , потім дані  $(D/2)/2$  для масштабу  $L/2^{N-2}$ ..., поки в кінці не вийде 2 елементи даних для масштабу  $L/2$ . Результатом буде масив тієї ж довжини, що і вхідний, де дані зазвичай упорядковано від найбільш великих масштабів до найбільш дрібним.

### 1.7 Вейвлет-пакетне перетворення

На відміну від алгоритму Мала при ДВП на кожному кроці обчислюються коефіцієнти деталізації та апроксимації. Такий підхід може бути корисним і використовується в деяких випадках, оскільки в одному рівні може бути більше корисної інформації, ніж на іншому рівні [4].

Адекватність моделі може бути збільшена за рахунок використання вейвлет-пакетного перетворення (ВПП) мережевого трафіку. Повторне фільтрування деталей дозволяє виділити аномалії в мережевому трафіку. На рис. 1.12 представлено ВПП. При переході з масштабного рівня  $m$  на наступний рівень  $m+1$  коефіцієнти деталізації  $d(m,n)$  та апроксимації  $c(m,n)$  знову поділяються на високочастотні та низькочастотні частини спектрального діапазону [4].

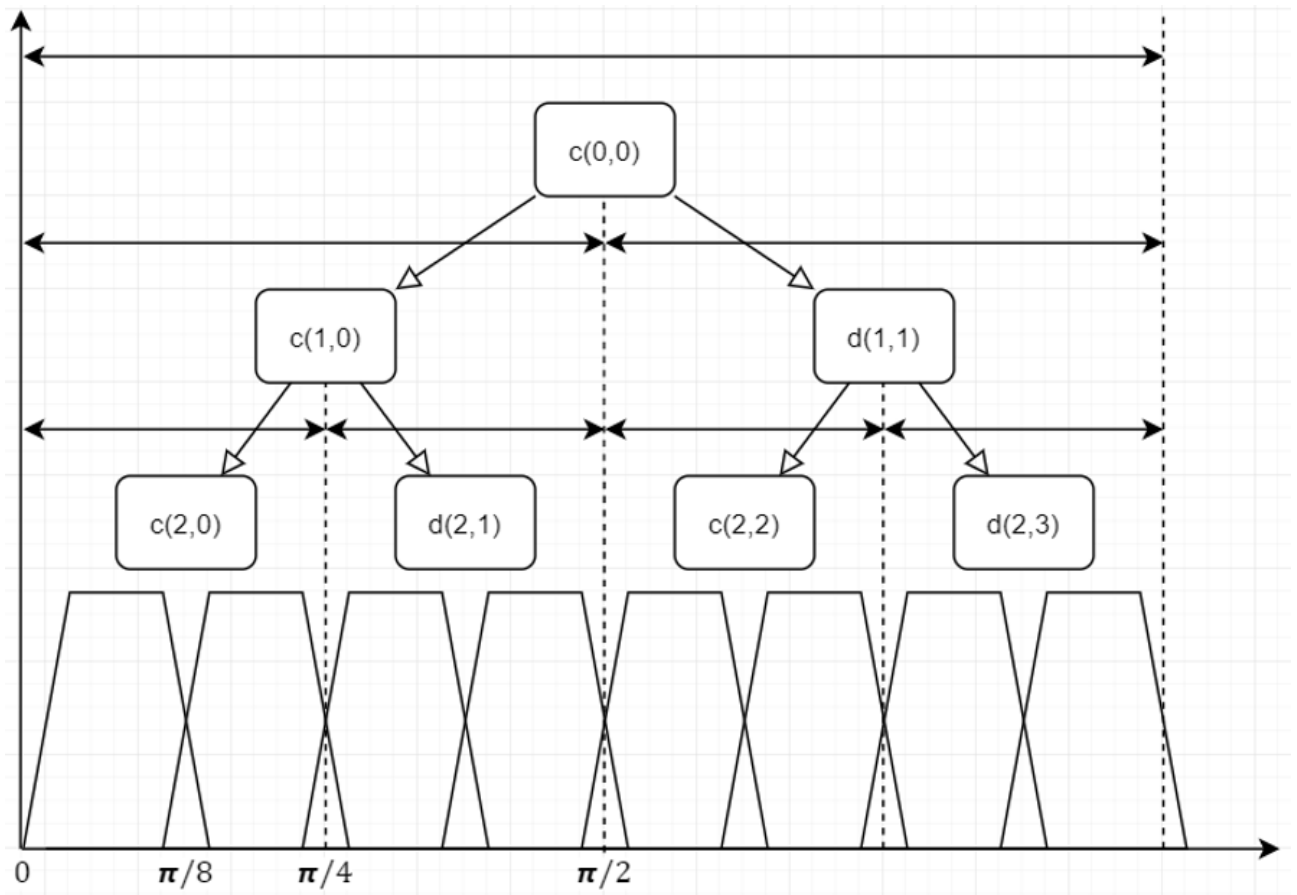


Рисунок 1.12 – Спектральні характеристики і алгоритм ВПП

### 1.8 Видалення шумів за допомогою вейвлетів

Високочастотні компоненти нерідко є шумами. Видалення шумів із сигналу є важливим процесом зменшення рівня помилок при виявленні мережних аномалій [12].

Шум часто може бути випадковим, тобто не пов'язаним із самим сигналом. Під час передачі даних перешкоди “білий шум” можуть часто виникати, які створюють коефіцієнти деталізації з великими випадковими викидами значень сигналу. Такі перешкоди можуть бути помітні на високочастотних компонентах, але в середньочастотних і низькочастотних компонентах малопомітні. Шумовий компонент краще відбивається в деталізуючих коефіцієнтах. Можна зменшити рівень шумів задавши певний поріг їхнього рівня, і зрізавши за рівнем деталізуючі коефіцієнти [12].

Є такі види процедури порогової обробки – жорстка та м'яка процедура порогової обробки.

Загалом алгоритм усунення шуму на основі вейвлетів виконаний таким чином[12].

Крок. 1. Виконується декомпозиція (вибирається вейвлет та рівень розкладання, і обчислюється ВП).

Крок 2. До кожного рівня коефіцієнтів деталізації виконується порогову обробку.

Крок 3. Виконується реконструкція сигналу, що ґрунтується на початкових коефіцієнтах апроксимації та модифікованих коефіцієнтах деталізації.

Значення, які нижче за заданий поріг, розглядаються як шуми, які можна фільтрувати.

На рис. 1.13 відображений безшумний сигнал (виділяється червоним кольором), який набагато корисніший, ніж оригінальний сигнал (синім кольором). Шумозаглушення виконується за типом – м'який поріг обмеження.

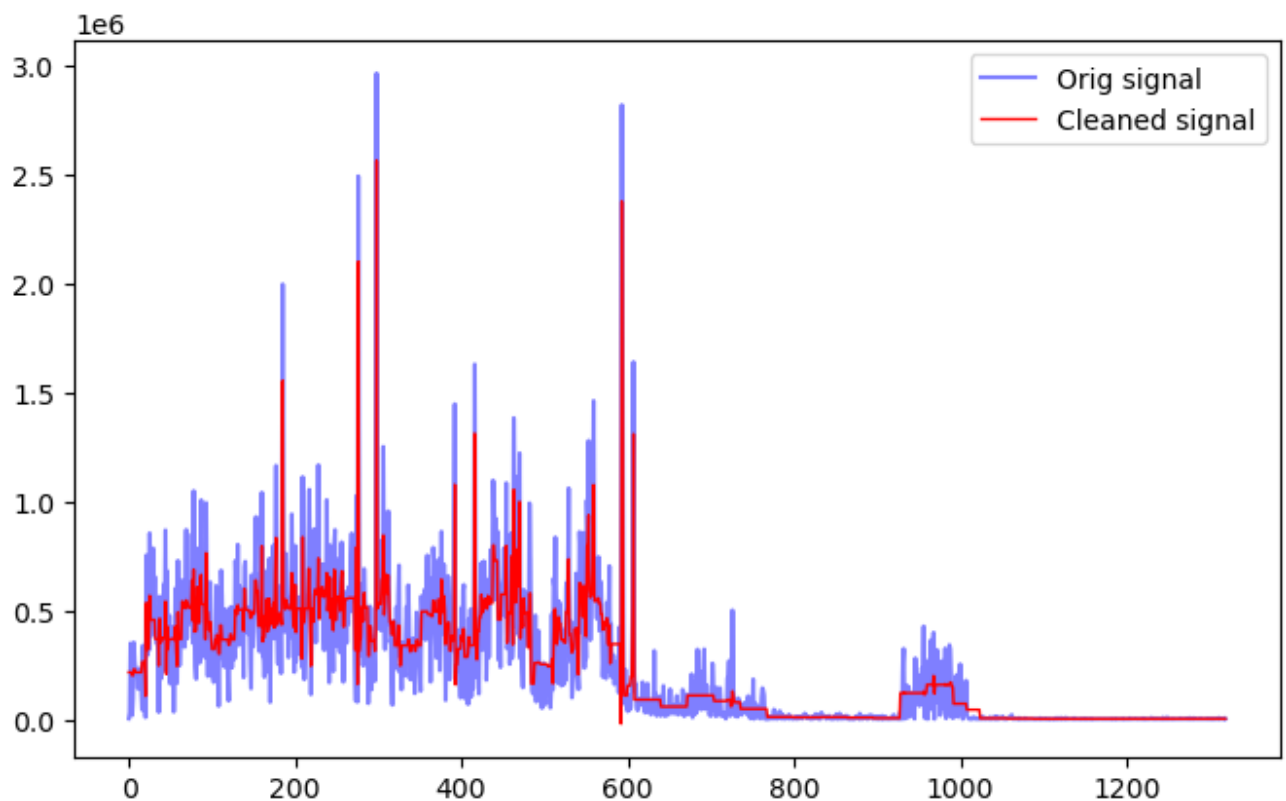


Рисунок 1.13 – Оригінальний і безшумний сигнал

## **1.9 Аналіз існуючих способів застосування вейвлет-аналізу виявлення мережових атак**

На даний момент існує багато робіт із застосування ВП для виявлення різних атак у мережах. З кожним часом робіт стають більшими і точнішими. Як давно відомо, Kwon et al. [16] використовує ВП для виявлення мережових аномалій у разі можливої атаки зломисника. З кожним часом почали ширше розглядати та розвивати ВП як інструмент виявлення мережових аномалій різного типу.

У роботі [17] для моделювання мережевого трафіку було запропоновано розвиток методу вейвлет-аналізу, який забезпечує підвищення достовірності рішень, що приймаються, систем виявлення атак в умовах параметричної невизначеності інформаційних процесів.

У роботі [18] для виявлення DDoS-атак представлений підхід, який заснований на застосуванні вейвлет-розкладання для аналізу мережного трафіку у разі впливу на нього аномалій типу SYN-flood та Slowloris. У ньому використовується вейвлет Добеші. Процес дослідження в цій роботі виконується в різних умовах, тобто з наявністю атак і без атак, для оцінки впливу атак. Згідно з результатом різні атаки впливають на зміни коефіцієнтів деталізації та апроксимації, наприклад, атака Slowloris майже не викликає зміни, що робить її більш складною для виявлення.

У роботі [19] як вихідні дані автори використовували масиви середніх значень за 5-хвилинними інтервалами: кількості байтів, пакетів, потоків в секунду, а також величина середнього розміру TCP-пакета. Ці дані подаються як дискретної послідовності частотно-часового сигналу. А наявність різких амплітуд в якомусь сигналі є мережовим аномалієм. Але є кілька груп мережових аномалій:

- аномалії, що виникають через помилки або збою мережевого обладнання (G1);
- DDoS-атаки (G2);

- навантаження в мережі (G3);
- інші аномалії, такі як обмін великою кількістю даних, помилка під час запису на датчик або надсилання даних (G4).

Виділено три компоненти – низькочастотний, середньочастотний та високочастотний. Низькочастотний компонент сигналу захоплював аномалії, тривалість яких може бути від кількох днів. Середньочастотний компонент призначений для аналізу в межах одного дня. А високочастотний компонент відповідав невеликим короткостроковим змінам, які можуть розглядатися як шум і не лише.

Після розбиття вихідного сигналу застосовують перші два компоненти в рамках ковзного вікна розміром 3 години. Далі застосовується метод граничного аналізу до виваженої суми цих компонентів. І якщо пікова точка перевищила заданий поріг, то аномалія є.

В результаті цього дослідження було зроблено висновок, що певні типи аномалій можуть бути ідентифіковані на конкретних частотах. Тобто аномалії груп G1, G2 та G4 виявляються на високих та середніх частотах, тоді як аномалії групи G3 виявляються на низьких та середніх частотах.

У роботі [20] автори підкреслили, що методи машинного навчання досягли хороших результатів у виявленні мережевих аномалій, але вони все ще стикаються з деякими проблемами, у тому числі поведінкова несхожість у даних навчання та тестування, а також обмежена можливість виявлення раніше невідомих атак.

Тому був розроблений метод моделювання мережевих сигналів для виявлення аномалій у мережах із застосуванням вейвлет-аналізу, точніше архітектура підходу складається з трьох компонентів: аналіз ознак, моделювання нормального мережевого трафіку на основі вейвлет-апроксимації та прогнозування за допомогою ARX моделі, прийняття рішення про наявність або відсутності атак. А результат був оцінений за допомогою набору даних DARPA, який включає в себе денні нормальні мережеві трафіки без атак і денні мережеві

трафіки з різними типами атак, і показує, що цей підхід забезпечує високий рівень виявлення атак.

У першому компоненті відбувається аналіз ознак мережі, точніше вибір та виділення надійних мережеских функцій, які можуть відрізнити аномальну поведінку від нормальної поведінки мережі. Для вимірювання поведінки всієї мережі є п'ять основних показників:

- FlowCount. Потік складається з декількох пакетів, що йдуть від джерела до певного пункту призначення протягом певного часу;
- AverageFlowPacketCount. Середня кількість пакетів у потоці за часовий інтервал;
- AverageFlowByteCount. Середня кількість байтів у потоці за часовий інтервал. За допомогою цього можна визначити, складається мережеский трафік з пакетів великого розміру чи ні;
- AveragePacketSize. Середня кількість байтів на пакет у потоці протягом інтервалу часу;
- FlowBehavior. Відношення FlowCount до AveragePacketSize. Вимірює аномальність поведінки потоку. Чим вище значення відношення, тим паче аномальні потоки.

На основі цих показників автори визначили 15 функцій (5 показників для протоколів TCP, UDP та ICMP) та використали такий 15-мірний вектор ознак для моделювання.

Далі у другому компоненті виконано моделювання нормального мережеского трафіку за допомогою Wavelet та ARX. У роботі використовується дискретне вейвлет-перетворення, а як вейвлет-функція була обрана Хаара, оскільки досягає найкращий результат серед інших за швидкістю та точності виконання. Моделювання нормального мережеского трафіку складається з двох етапів, а точніше вейвлет-декомпозиції/реконструкції та створення авторегресійної моделі. Щоб оцінити параметри ARX та створити модель прогнозування ARX, автори використовували вейвлет-коефіцієнти з першої

частини навчальних даних як вхідні дані, а вейвлет-коефіцієнти з іншої частини навчальних даних як дані припасування моделі. А процес припасування ARX використовується для оцінки оптимальних параметрів, які ґрунтуються на помилках найменших квадратів.

Отримавши прогнозовану модель нормального мережевого трафіку, її можна використовувати визначення аномальних сигналів від нормальних. Коли вхідні дані містять лише звичайний трафік, її вихідні дані, які називаються залишками, будуть близькими до 0, що означає, що прогнозоване значення, створене моделлю, близько до фактичних вхідних нормальних дій. А коли вхідні дані містять нормальний та аномальний трафік, залишки матимуть багато піків, де виникають аномалії. У цьому випадку залишки є математичним перетворенням, яке намагається обнулити нормальні мережеві дані та посилити аномальні дані. А щоб ідентифікувати піки залишків, автори використали алгоритм виявлення піків на основі моделі суміші Гаусса та приймають рішення про атаку.

### **1.10 Переваги та недоліки застосування ВП у виявленні мережевих атак**

Переваги:

- підвищення продуктивності під час використання ДВП;
- вейвлетні базиси можуть бути добре локалізованими як за частотою, так і за часом. При виділенні в сигналах добре локалізованих різномасштабних процесів можна розглядати масштабні рівні розкладання, які подають інформації про наявність мережевих атак;

- можливість виявити невідомих аномалій та атак у мережах;
- базові вейвлети можуть реалізуватися функціями різної гладкості.

Недоліки:

- неоднозначність вибору базових функцій;

– нетривальний вибір розміру вікна. Якщо розмір вікна набагато перевищує тривалість аномалії, то атака не буде виявлена через сильне згладжування частотного сплеску. А якщо розмір вікна занадто малий, то неминуче будуть помічені бездумні аномалії, як шуми.

### **1.11 Висновки за розділом 1**

У першому розділі було проведено аналіз мережевих аномалій та атак, їх призначень та характеристик. Далі було проведено аналіз існуючих методів виявлення мережевих аномалій та атак, їх переваги та недоліки. Після чого було ознайомлено з вейвлет-аналізом, а саме з особливостями, областями застосування, характеристиками вейвлет функцій, ДВП і БВП, а також зі способами застосування ВП у задачах виявлення мережевих атак.

## **2 РОЗРОБКА АРХІТЕКТУРИ ПРОГРАМИ**

### **2.1 Архітектура системи, що розробляється**

Архітектура розробленої системи складається з таких компонентів:

- підсистема, яка виконує перетворення вхідного мережного трафіку у потрібний для системи формат. Вхідний мережевий трафік представляється як послідовностей пакетів, підсистема генерує кілька ознак мережного трафіку кожного протоколу за заданий часовий інтервал (у разі 1 хвилина);
- підсистема, яка завантажує оброблений мережевий трафік та виконує ВП за заданими параметрами. Перед виконанням ВП виконується шумоусунення у разі необхідності, а далі проводиться ВП для отримання коефіцієнтів деталізації та апроксимації, які необхідні для візуалізації та прийняття рішень щодо можливих атак;
- інтерфейс користувача користувача, необхідний для здійснення зв'язку між підсистемами і користувачем.

### **2.2 Вибір мови програмування**

Для розробки та застосування вейвлет-перетворення підходить будь-яка мова, яка більш призначена для аналізу даних, а саме R, Python, Matlab і т.д. Кожен із них хороший для вирішення певних завдань. У кожного з них є свої переваги та недоліки. Так наприклад, Matlab добре працює з великими даними і середовище Matlab включає безліч зручних функцій і компонентів з готовими графічними елементами, тому окремо писати код на відомі функції не потрібно, а лише відразу працювати з даними через вбудовані пакети. Крім того, у Matlab більш детальні та точні візуалізації даних, наприклад, на графіках вейвлет Хаара представляється у вигляді сходинки, а у Python такої переваги немає, однак можливо реалізувати вигляд самому. А Python більш читабельний і простий у використанні, і з кожним часом стає популярнішим для аналізу даних і не тільки.

Python – високорівнева інтерпретована та об'єктно-орієнтована мова програмування з динамічною типізацією. Зручна і універсальна мова, яка використовується в аналізі даних, машинному навчанні, математичних розрахунках, та веб-розробці [21]. Йому віддають перевагу такі великі корпорації у сфері технологій. На ньому набагато простіше читати та писати програми, порівняно з іншими мовами.

Переваги мови Python:

- простота. Python легко читати або легко вивчати;
- можливість розширення. Існують бібліотеки та фреймворки для вирішення більшості завдань. Також ми можемо використовувати код на мові C з Python;
- інтерпретованість. Інтерпретатор Python існує для усіх популярних платформ;
- об'єктно-орієнтоване програмування. Підтримує орієнтоване програмування, воно дозволяє поліморфізм та успадкування.

Недоліком мови Python є невисока швидкість виконання, що обумовлено її інтерпретованістю.

Matlab – високорівнева інтерпретована мова програмування разом із пакетом прикладних програм та IDE для роботи з матричними базами даних, розробки, виконання інженерних та математичних розрахунків, а також візуалізації. Matlab активно застосовується в проектуванні та аналізі нейронних мереж для вирішення різних завдань (розпізнавання образів, класифікації, прогнозування тощо), в аналізі, обробці та покращенні якості різних сигналів, а також для побудови математичних моделей різних біологічних процесів та організмів [22].

Переваги мови Matlab:

- легкість освоєння. У Matlab простий та інтуїтивно зрозумілий синтаксис, а також маса супровідної документації від розробників;

- широкий функціонал. MatLab включає безліч бібліотек для обробки та побудови графіків, вейвлет-перетворення, розв'язання диференціальних рівнянь тощо;

- перетворення на швидкий код. Програми, написані на MatLab, можна перекласти на C/C++ за допомогою MatLab Coder. Це дозволяє виконувати програми набагато швидше.

Недоліки мови Matlab:

- швидкість. Роботу програм, написаних на MatLab суттєво уповільнюють велику кількість команд та операторів.

R – мова програмування та середовище для статистичних обчислень та графіки. Він надає широкий спектр статистичних (класифікація, кластеризація, аналіз часових рядів і т.д.) та графічних методів, що є високо розширюваним [23].

Переваги мови R:

- ефективність роботи з векторами та матрицями;
- широкий функціонал. R включає безліч пакетів для аналізу даних, вейвлет-перетворення, рішення диференціальних рівнянь тощо;
- багато якісних пакетів для візуалізації даних для різних завдань.

Недоліки мови R:

- складність навчання програмі;
- швидкість. Роботу програм також суттєво уповільнюють велику кількість команд та операторів, проте є пакети, що дозволяють підвищити швидкість роботи.

В таблиці 2.1 наведено порівняння мов програмування.

Таблиця 2.1 – Порівняння мов програмування

| <b>Критерій</b>                              | <b>Python</b>             | <b>R</b> | <b>Matlab</b> |
|----------------------------------------------|---------------------------|----------|---------------|
| Кросплатформність                            | Так                       | Так      | Так           |
| Динамічна типізація                          | Так                       | Так      | Так           |
| Візуалізації даних                           | Так                       | Так      | Так           |
| Продуктивність                               | +                         | +        | ++            |
| Простота та читабельність                    | +++                       | ++       | ++            |
| Підтримка аналізу даних                      | +++                       | +++      | +++           |
| Технічна документація та навчальні посібники | +++                       | ++       | +++           |
| Бібліотеки для використання вейвлет-аналізу  | Так                       | Так      | Так           |
| IDE                                          | PyCharm,<br>Visual Studio | RStudio  | Matlab        |

Для цієї роботи була обрана мова Python, оскільки вона зручна в читанні та використанні, а також стає популярною для аналізу даних і для таких завдань є достатньо бібліотек з достатніми документаціями. Крім того, надалі працюватиме зручно реалізувати додаткові модулі, наприклад, для моделювання нормального мережевого трафіку на основі вейвлет-апроксимації та прогнозування за допомогою авторегресійної (ARX) моделі.

### 2.3 Вибір додаткових бібліотек

Для спрощеної роботи без великих витрат часу та праці на написання великого коду вибрали такі бібліотеки Python:

- PyQt (5.15.4) – реалізація потужного фреймворку Qt для мови Python. Він переважно використовується для створення GUI. Ми його використовуємо для створення графічних вікон з елементами керування для зручної роботи;
- Csv – основна бібліотека для роботи з файлами CSV в Python. З його допомогою можемо завантажувати дані з CSV і виконувати обробку з даними як завгодно, а також зберегти CSV;
- NumPy (1.20.3) – бібліотека для Python, яка додає підтримку великих багатовимірних масивів та матриць, разом із бібліотекою високорівневих та швидких математичних функцій для операції з масивами;
- Matplotlib (3.4.3) – бібліотека для Python для візуалізації даних двовимірною та тривимірною графікою. З його допомогою можна буде візуалізувати сигнали та вейвлети коефіцієнти;
- PyWavelets (1.1.1) – бібліотека для Python для ВП. Він включає також ДВП, яке будемо використовувати в цій роботі.

## 2.4 Вибір набору даних

Оскільки вейвлет-перетворення передбачає роботу з рядом даних, дані мережного трафіку повинні бути сформовані у вигляді ряду значень, які описують зміну обсягу трафіку в часі.

Для проведення тестування в якості вхідних даних був використаний набір даних DARPA, який включає нормальні мережеві трафіки та мережеві трафіки з різними типами атак [24].

Ці мережеві трафіки були записані по 22 години на добу з понеділка до п'ятниці протягом 5 тижнів. Автори розділили їх на навчальні та тестуючі дані, де перші три тижні відносяться до навчальних даних, а останні два тижні відносяться до тестуючих даних.

Трафіки, які записані у перший і третій тиждень, не містять жодних атак. А трафіки, які записані в інші тижні, містять безліч різних типів атак, з яких

найбільш часто зустрічаються атак наведені в таблиці 2.2. Є список задокументованих атак, які скоєні другого тижня [25]. Цей список необхідний для оцінки виявлення атак.

Таблиця 2.2 – Список частих атак у мережевих трафіках

| Атака    | Опис                                                                                                                                                                                      |
|----------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| back     | Атака типу "відмова в обслуговуванні" на веб-сервер apache, коли клієнт запитує URL-адресу, що містить багато зворотних косих характеристик.                                              |
| land     | Відмова в обслуговуванні, коли віддаленому вузлу надсилається UDP-пакет з тим самим джерелом та призначенням.                                                                             |
| mailbomb | Атака типу «відмова в обслуговуванні», коли ми надсилаємо поштовому серверу багато великих повідомлень для доставки, щоб уповільнити його, можливо, ефективно зупиняючи нормальну роботу. |
| neptune  | Відмова в обслуговуванні Syn flood на одному чи кількох портах.                                                                                                                           |
| pod      | Відмова в обслуговуванні – пінг смерті.                                                                                                                                                   |
| smurf    | Відмова в обслуговуванні – істр луна відповідь флуд.                                                                                                                                      |
| syslog   | Відмова в обслуговуванні для служби syslog підключається до порту 514 з нерозв'язною вихідною IP-адресою.                                                                                 |
| teardrop | Відмова в обслуговуванні, коли неправильно фрагментовані пакети UDP призводять до перезавантаження деяких систем.                                                                         |

## 2.5 Обробка даних

Щоб охарактеризувати поведінку мережного трафіку, потрібно згенерувати кілька ознак за заданий часовий інтервал для кожного протоколу. Як

відомо, що більше ознак, тим точніше буде охарактеризовано поведінка мережного трафіку різних потоках при різних типів атак. Ми виділили такі ознаки (табл. 2.3), які будуть згенеровані для загального та кожного протоколу мережного трафіку.

Існує кілька протоколів – TCP, UDP та ICMP. UDP протокол (User Datagram Protocol) без встановлення з'єднання передає дані, не підтверджує доставку даних та не робить повтори. TCP протокол (Transmission Control Protocol), який встановлює з'єднання перед передачею, підтверджує доставку даних та при необхідності робить повтор, гарантує цілісність і правильну послідовність даних при завантаженні. ICMP протокол (Internet Control Message Protocol) використовується мережними пристроями для діагностики проблем із мережевою взаємодією. Основною метою ICMP є створення звітів про помилки. Мережеві атаки можуть використовувати цей процес, створюючи засоби руйнування, такі як атака ICMP-флуда і атака ping of death [26].

Таблиця 2.3 – Ознаки мережного трафіку

| № | Ознак            | Опис                                                                    |
|---|------------------|-------------------------------------------------------------------------|
| 1 | bytes_i          | Кількість байтів за часовий інтервал (i)                                |
| 2 | packets_i        | Кількість пакетів за часовий інтервал (i)                               |
| 3 | flows_i          | Кількість потоків за часовий інтервал (i)                               |
| 4 | avg_flow_packets | Середня кількість пакетів у потоці за часовий інтервал (i)              |
| 5 | avg_flow_bytes   | Середня кількість байтів у потоці за часовий інтервал (i)               |
| 6 | avg_packet_size  | Середній розмір пакета у потоці за часовий інтервал (i)                 |
| 7 | flow_behavior    | Відношення кількості потоків до байтів на пакет за часовий інтервал (i) |

## 2.6 Висновки за розділом 2

У цьому розділі була описана загальна архітектура системи, що розробляється, з декількох компонентів для виконання поставлених завдань. Далі були проаналізовані мови програмування та бібліотеки, необхідні для виконання завдання обробки даних та виявлення мережових аномалій. Було обрано мову програмування Python та деякі бібліотеки – PyQt для створення GUI, NumPy для підтримки обробки масивів та матриць, Csv для роботи з CSV файлами, Matplotlib для візуалізації даних та PyWavelets для застосування вейвлет-перетворення.

Далі було обрано та проаналізовано набір даних DARPA, який включає нормальні та атакуючі мережові трафіки. Було проаналізовано, яким чином мають бути опрацьовані дані.

### 3 ОСНОВНІ РІШЕННЯ ЩОДО РЕАЛІЗАЦІЇ КОМПОНЕНТІВ СИСТЕМИ

#### 3.2 Попередня обробка мережного трафіку

Для застосування ВП та аналізу мережного трафіку для вирішення про ймовірне вторгнення атак потрібні підготовлені дані з потрібними інформаціями та без зайвих незначних ознак. Ці підготовлені дані повинні бути у форматі CSV для читання програмою. Вихідними даними можуть бути будь-якого типу і головне, щоб вони містили мінімально необхідні ознаки для програми, такі як кільк. байтів у пакеті, протокол, адресу та порт відправника, адресу та порт одержувача.

Найбільш популярний тип файлу, що містить все для мережного трафіку, який був записаний утилітою або якоюсь програмою, є `.tcpdump`.

У нашому випадку ми беремо вихідний мережевий трафік у вигляді `.tcpdump` і відкриваємо через будь-яку програму, яка підтримує даний тип файлу. Найбільш популярною програмою для цього є Wireshark (рис. 3.1).

Wireshark – це така програма-аналізатор трафіку з графічним інтерфейсом для комп'ютерних мереж Ethernet та інших. Функціональність, яку надає Wireshark, дуже схожа на можливості програми `tcpdump`.

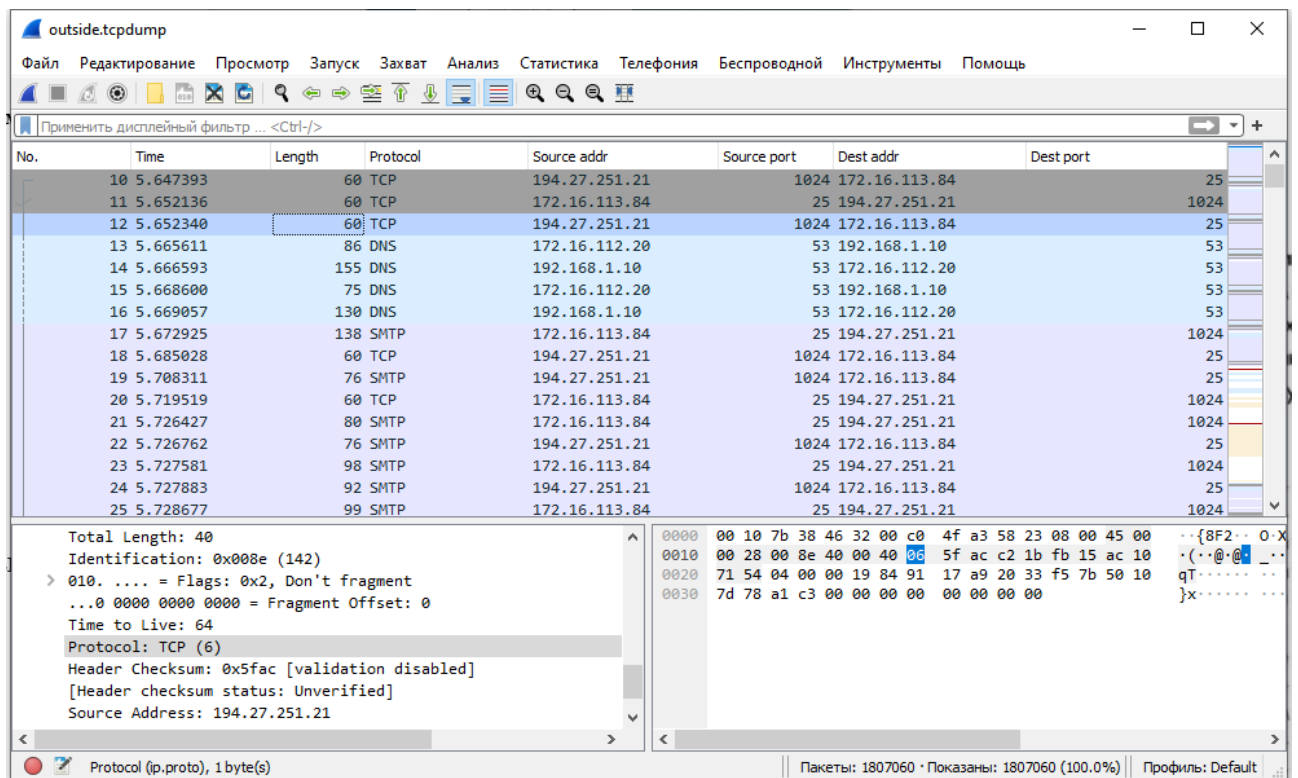


Рисунок 3.1 – Програма-аналізатор мережного трафіку Wireshark

Після завантаження або захоплення мережевого трафіку Wireshark спочатку необхідно прибрати зайві властивості пакетів і вибрати необхідні, потім збережемо (рис. 3.2). Після цього програма відображатиме лише ті вибрані властивості та експортуватиме трафік у CSV файл тільки з тими властивостями.

Необхідними властивостями мають бути такими:

- порядковий номер;
- час від початку захоплення (у секундах);
- розмір пакету (байт);
- протокол;
- адреса відправника;
- порт відправника;
- адреса одержувача;
- порт отримувача.

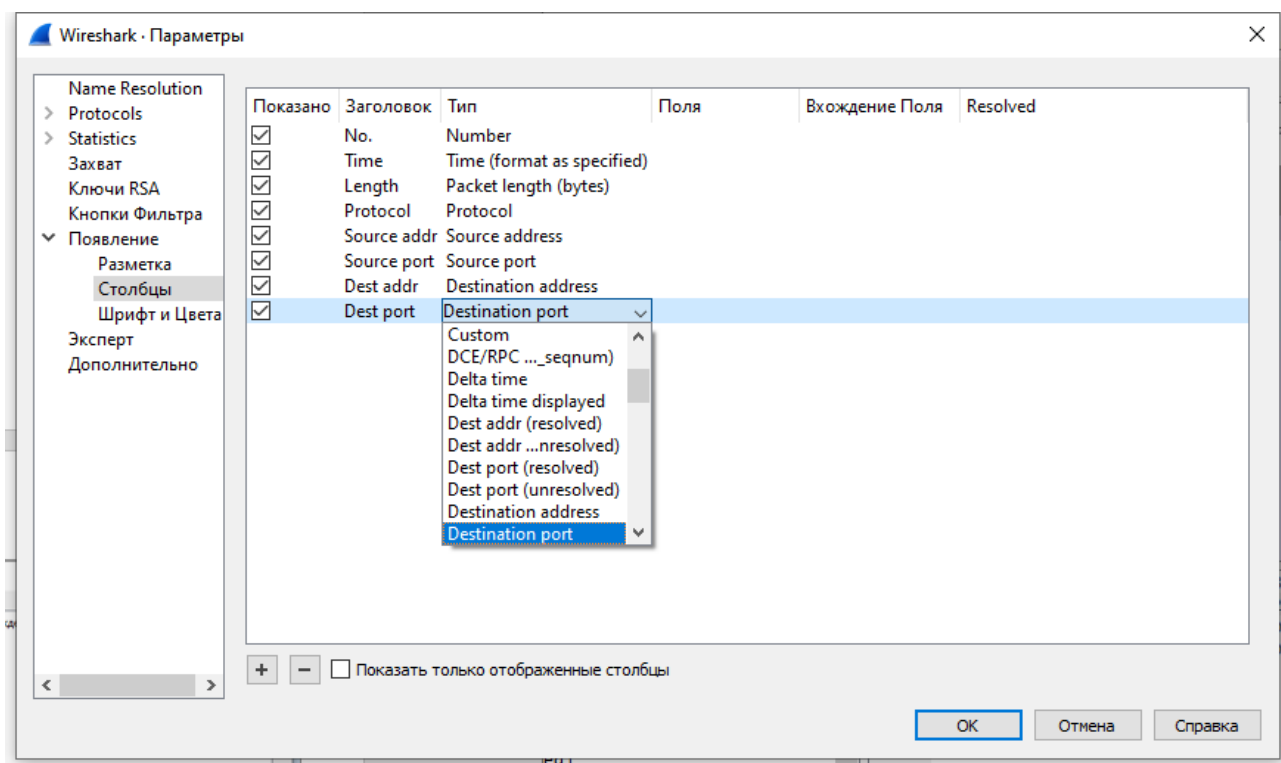


Рисунок 3.2 – Вибір необхідних ознак мережного трафіку Wireshark

Після вибору властивостей можна проходити фільтрацію для виключення непотрібних інформацій, наприклад, виключити всі пакети, у яких протоколи не рівні заданого значення, як показано на рис. 3.3. Це може бути корисним для більш точного аналізу.

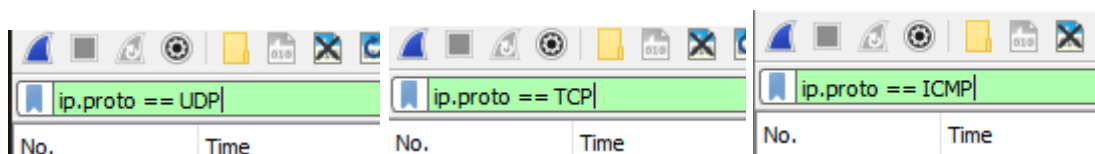


Рисунок 3.3 – Фільтрування мережного трафіку за протоколом у Wireshark

Після фільтрації можна зберегти файл CSV (рис. 3.4). У нашому випадку з одного вихідного мережного трафіку виходять 4 різні підготовлені трафіку з різними протоколами: всі протоколи, протокол ICMP, TCP та UDP. Ці підготовлені трафіки можна завантажити в нашу програму та виконувати подальші роботи.

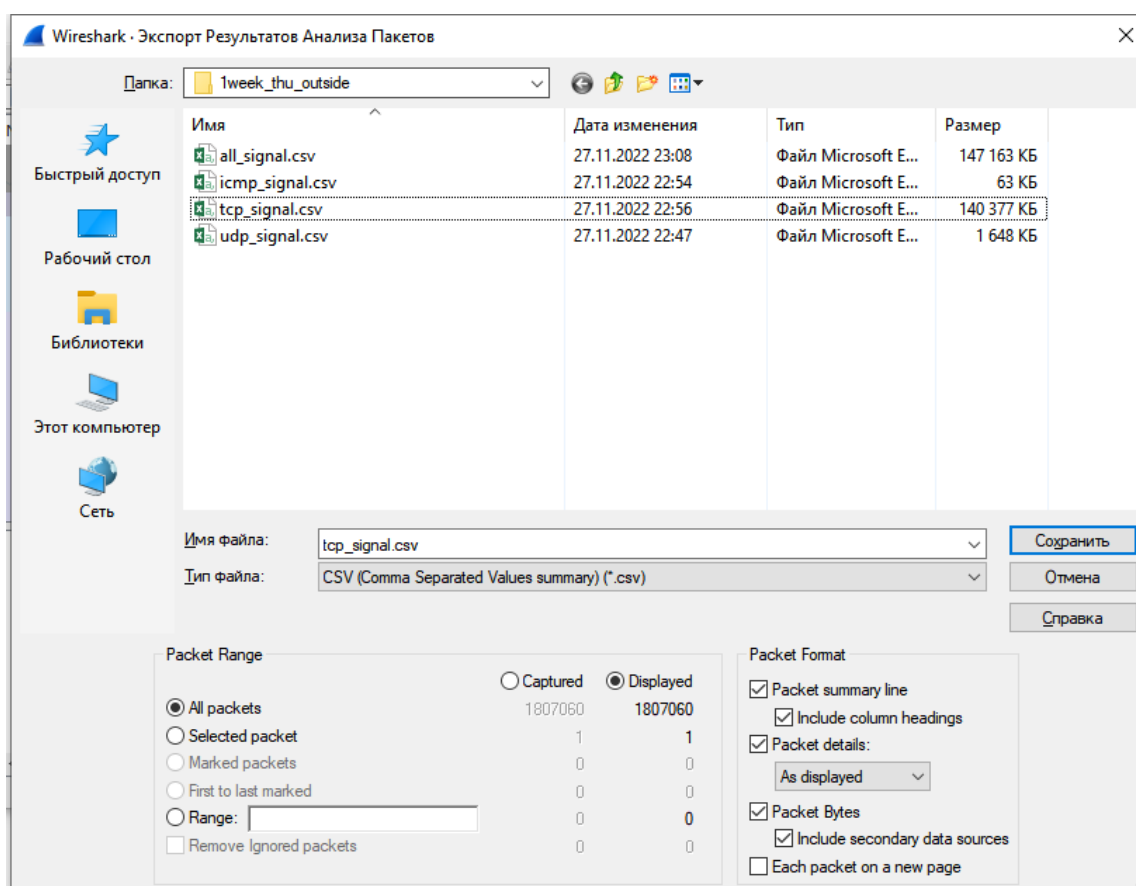


Рисунок 3.4 – Збереження підготовленого трафіку у CSV файл у Wireshark

Зміст вихідного файлу представляється на рис. 3.5.

```

1  "No.", "Time", "Length", "Protocol", "Source addr", "Source port", "Dest addr", "Dest port"
2  "1", "0.000000", "54", "LLC", "HewlettP_61:aa:c9", "", "HewlettP_61:aa:c9", ""
3  "2", "0.585053", "324", "CDP", "Cisco_38:46:32", "", "CDP/VTP/DTP/PAgP/UDLD", ""
4  "3", "1.499816", "54", "LLC", "HewlettP_61:aa:c9", "", "HewlettP_61:aa:c9", ""
5  "4", "2.499787", "54", "LLC", "HewlettP_61:aa:c9", "", "HewlettP_61:aa:c9", ""
6  "5", "3.501146", "54", "LLC", "HewlettP_61:aa:c9", "", "HewlettP_61:aa:c9", ""
7  "6", "3.870239", "60", "LOOP", "Cisco_38:46:32", "", "Cisco_38:46:32", ""
8  "7", "5.000457", "54", "LLC", "HewlettP_61:aa:c9", "", "HewlettP_61:aa:c9", ""
9  "8", "5.640683", "77", "DNS", "192.168.1.10", "53", "172.16.112.20", "53"
10 "9", "5.641887", "142", "DNS", "172.16.112.20", "53", "192.168.1.10", "53"
11 "10", "5.647393", "60", "TCP", "194.27.251.21", "1024", "172.16.113.84", "25"
12 "11", "5.652136", "60", "TCP", "172.16.113.84", "25", "194.27.251.21", "1024"
13 "12", "5.652340", "60", "TCP", "194.27.251.21", "1024", "172.16.113.84", "25"
14 "13", "5.665611", "86", "DNS", "172.16.112.20", "53", "192.168.1.10", "53"
15 "14", "5.666593", "155", "DNS", "192.168.1.10", "53", "172.16.112.20", "53"
16 "15", "5.668600", "75", "DNS", "172.16.112.20", "53", "192.168.1.10", "53"
17 "16", "5.669057", "130", "DNS", "192.168.1.10", "53", "172.16.112.20", "53"
18 "17", "5.672925", "138", "SMTP", "172.16.113.84", "25", "194.27.251.21", "1024"
19 "18", "5.685028", "60", "TCP", "194.27.251.21", "1024", "172.16.113.84", "25"
20 "19", "5.708311", "76", "SMTP", "194.27.251.21", "1024", "172.16.113.84", "25"
21 "20", "5.719519", "60", "TCP", "172.16.113.84", "25", "194.27.251.21", "1024"
22 "21", "5.726427", "80", "SMTP", "172.16.113.84", "25", "194.27.251.21", "1024"
23 "22", "5.726762", "76", "SMTP", "194.27.251.21", "1024", "172.16.113.84", "25"
24 "23", "5.727581", "98", "SMTP", "172.16.113.84", "25", "194.27.251.21", "1024"
25 "24", "5.727883", "92", "SMTP", "194.27.251.21", "1024", "172.16.113.84", "25"
26 "25", "5.728677", "99", "SMTP", "172.16.113.84", "25", "194.27.251.21", "1024"
27 "26", "5.729012", "90", "SMTP", "194.27.251.21", "1024", "172.16.113.84", "25"

```

Рисунок 3.5 – Вміст CSV файлу

### 3.3 Розробка компонентів

#### 3.3.1 Розробка підсистеми обробки даних

У лістингу 3.1 наведено код обчислення ознак із вихідного сигналу. Для початку завантажується файл CSV, потім програма читає його для генерації ознак, а потім записує їх в інший файл.

Лістинг 3.1 – Код обчислення ознак із вихідного сигналу

```
#Завантаження файлу
def loadFile(self):
    self.file_name = fd.askopenfilename(filetypes=(("CSV files", "*.csv"),
        ("All files", "*.*")))

#Процес обчислення ознак
def transf(self):
    with open(self.file_name, mode='r') as csv_file:
        csv_reader = csv.DictReader(csv_file)
        max_s = 80400 #макс. кол. екземплярів
        max_i = int(max_s/60) #кол. інтервалов

        #...
        #Повний код обчислення ознак наведено у Додатку А
    #...
#створення файлу
    with open(self.save_file_name, 'w', newline='') as file:
        writer = csv.writer(file)
        writer.writerow(['bytes_i', 'packets_i', 'flows_i',
            'avg_flow_packets', 'avg_flow_bytes', 'avg_packet_size', 'flow_behavior'])

        #додавання ознак у дані
        data = []
        for i in range(len(flows_i)):
            data.append([bytes_i[i], packets_i[i], flows_i[i],
                avg_flow_packets[i], avg_flow_bytes[i], avg_packet_size[i], flow_behavior[i]])
            #Запис даних у файлі
        writer.writerows(data)
```

На рис. 3.6 графічно подаються всі ознаки мережевого трафіку з будь-якими протоколами з тимчасовим інтервалом на 1 хвилину. Виходячи з рисунку видно, що в один час найбільше пакетів, а в інший час найбільше потоків, це доводить, чому краще мати кілька ознак для аналізу.

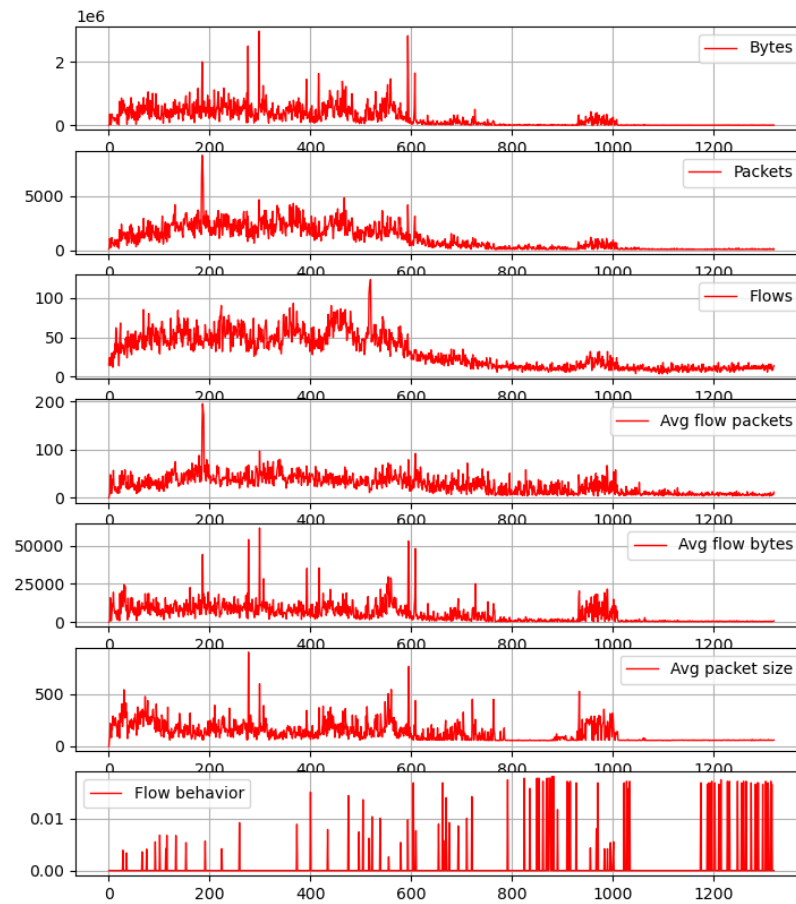


Рисунок 3.6 – Різні сигнали за ознаками

### 3.3.2 Розробка методів вейвлет-аналізу

#### 3.3.2.1 Бібліотека PyWavelets

Усі необхідні алгоритми вейвлет-аналізу знаходяться у бібліотеці PyWavelets.

Основними функціями бібліотеки PyWavelets можуть бути такими:

- дискретне вейвлет-перетворення (DWT);
- безперервне вейвлет-перетворення (CWT);
- видалення високочастотних шумів;
- вейвлет пакетне перетворення;
- зворотне дискретне вейвлет-перетворення (IDWT);
- станційне вейвлет-перетворення (SWT);
- інші.

Для завдання виявлення атак у мережах підходять дискретне та безперервне вейвлет перетворення, а також видалення високочастотних шумів для очищення сигналу.

Існують такі вейвлети, що входять до бібліотеки:

- Хаара (haar);
- Добеши (db);
- Симлет (sym);
- Койфлет (coif);
- біртогональний (bior);
- зворотний біртогональний (rbio);
- Гауса (gaus);
- мексиканський капелюх (mexh);
- Морле (morl);
- комплексний-вейвлет Гауса (cgau);
- Шеннона (shan);
- частотні вейвлети В-сплайну (fbsp);
- комплексний-вейвлет Морле (cmor).

У нашому випадку для виконання ДВП використовуються вейвлети Хаара, Добеші, Симлета та Койфлета із заданим порядком та рівнем розкладання.

### 3.3.2.2 Шумоусунення сигналу

У разі потреби можна застосувати шумоусунення сигналу шляхом видалення частини коефіцієнтів деталізації, оскільки коефіцієнти деталізації є високочастотною частиною сигналу. Це можна зробити за допомогою функції порогової обробки `pywt.threshold()` у бібліотеці `PyWavelets`:

```
pywt.threshold(data, value, mode= 'soft', substitute=0)
```

`data : array_like` – числові дані.

`value: scalar` – порогове значення.

`mode: {'soft', 'hard', 'garrote', 'greater', 'less'}` –

визначає тип порога, який буде застосовано до вхідних даних. За замовчуванням `'soft'`.

`substitute : float, optional` – основне значення (за замовчуванням: 0).

`output : array` – пороговий масив.

На рис. 3.7 показано три пороги (м'який, жорсткий та невід'ємний поріг Гаротта). Як видно, що невід'ємний поріг Гаротта проміжний між м'яким і жорстким порогом.

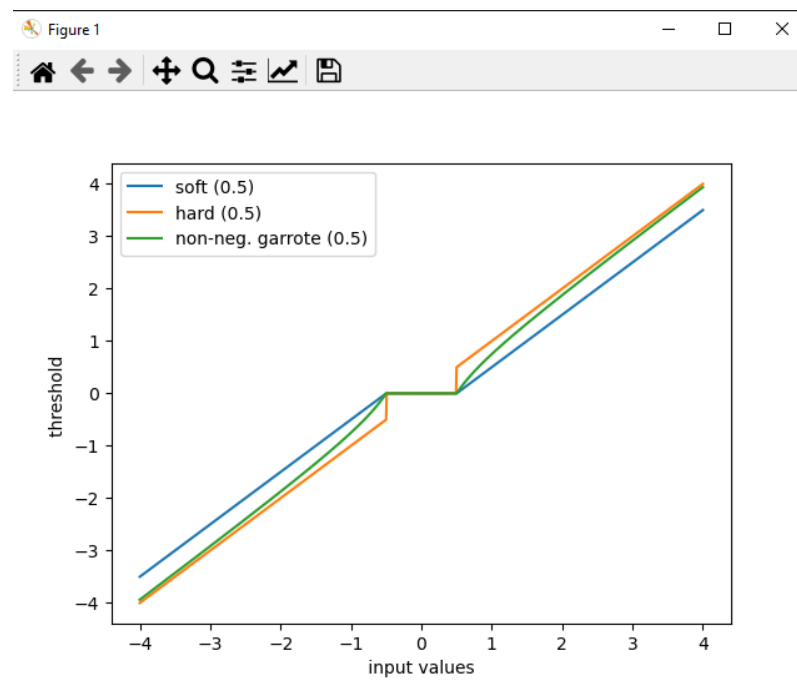


Рисунок 3.7 – Пороги

У лістингу 3.2 наведено код застосування функції порогової обробки з трьома порогами – м'який, Гаррота та жорсткий, для шумоусунення сигналу.

Лістинг 3.2 – Код застосування функції порогової обробки з трьома порогами

```
import pandas as pd
from pylab import *
import pywt
signal = df.iloc[:, 0]
v='haar'
thres=['soft', 'garrote', 'hard']
for w in thres:
    coeff = []
    def lowpassfilter(signal, thresh, wavelet=v):
        thresh = thresh*nansmax(signal)
        coeff = pywt.wavedec(signal, wavelet, level=5, mode='per')
        coeff[1:] = (pywt.threshold(i, value=thresh, mode=w) for i
in coeff[1:])
        reconstructed_signal = pywt.waverec(coeff, wavelet,
mode='per')
    return reconstructed_signal
fig, ax = subplots(figsize=(8,4))
ax.plot(signal, color="b", alpha=0.5, label='Orig signal')
rec = lowpassfilter(signal, 0.05)
ax.plot(rec, color="r", label='Cleaned signal', linewidth=1)
ax.legend()
show()
```

Результат шумоусунення сигналу відображено на рис. 3.8 після застосування різних поріг – м'який, Гаррота та жорсткий відповідно. Оригінальний сигнал виділено синім кольором, а очищений сигнал червоним. Виходячи з результату видно, що високочастотні частини сигналу видалені.

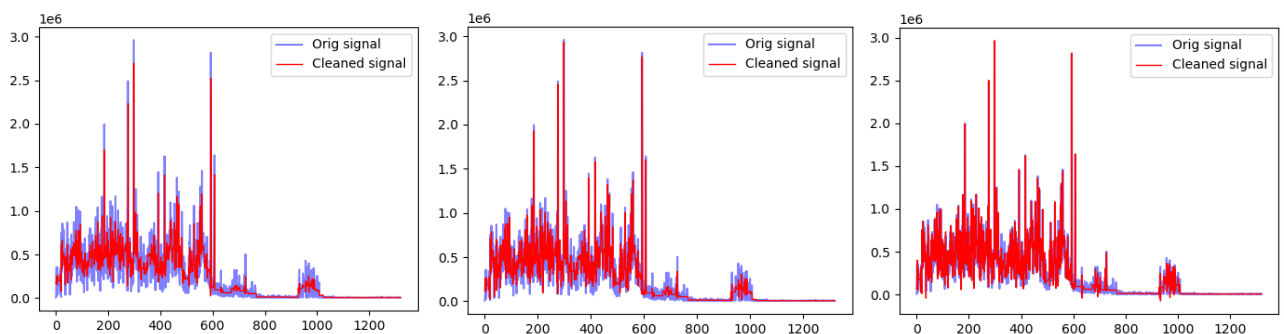


Рисунок 3.8 – Оригінальний та очищений сигнал після застосування різних поріг (м'який, Гаррота та жорсткий відповідно)

### 3.3.2.3 Застосування безперервного вейвлет-перетворення

Для БВП можна використовувати функцію `cwt(data, scales, wavelet)`, де `data` – вхідний сигнал, `scales` – масштаб (макс. частота), `wavelet` – вейвлет-функція.

Код БВП для мережного трафіку наведено у лістингу 3.3. В даному випадку використовується вейвлет `mexh` (мексиканська капелюх).

#### Лістинг 3.3 – Код застосування БВП

```
figure()
plot(signal, 'red', linewidth=1, label="Flows")
grid()
legend(loc='best')
show()

scales = np.arange(1, 31)
coef, freqs = pywt.cwt(signal, scales, 'mexh')

plt.figure();
plt.imshow(abs(coef), extent=[0, 200, 30, 1],
            interpolation='bilinear', cmap='jet', aspect='auto')
plt.gca().invert_yaxis()
plt.yticks(np.arange(1, 31, 1))
plt.xticks(np.arange(0, 201, 10))
plt.show()
```

На рис. 3.9 показані вихідний сигнал та БВП з вейвлетом «мексиканський капелюх».

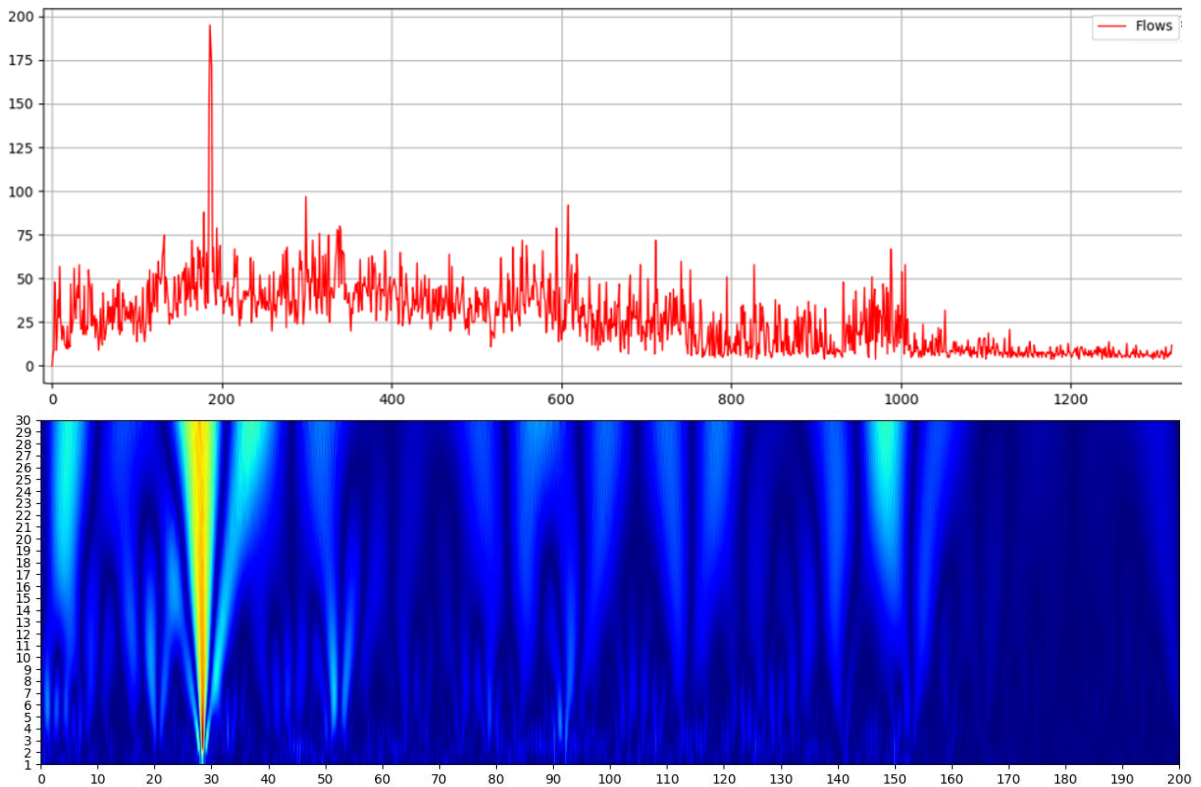


Рисунок 3.9 – Сигнал та БВП з вейвлетом mexh

#### 3.3.2.4 Застосування дискретного вейвлет-перетворення

Для використання ДВП за допомогою бібліотеки Python можна використовувати функцію `dwt(data, wavelet, mode='symmetric', axis=-1)`, де `data` – вхідний сигнал, `wavelet` – вейвлет-функція, `mode` – режим розширення сигналу. Ця функція повертає коефіцієнти апроксимації та деталізації, однак ця функція працює з одним рівнем, тому називається однорівневим ДВП. Для застосування багаторівневого ДВП використовується функція `wavedec(data, wavelet; mode='symmetric', level=None; axis=-1)`, всі параметри схожі як функції `dwt`, але з ще одним параметром `level`, який означає рівень декомпозиції. Ця функція повертає список масивів коефіцієнтів, де перший елемент є коефіцієнтами апроксимації обраного рівня та інші елементи є коефіцієнтами деталізації.

У лістингу 3.4 наведено код застосування багаторівневого ДВП.

## Лістинг 3.4 – Код застосування багаторівневого ДВП

```

self.file_name = fd.askopenfilename(filetypes=((("CSV files", "*.csv"),
    ("All files", "*.*")))
self.lineEdit.setText(self.file_name)
self.df = pd.read_csv(self.file_name, sep=",")

self.wavelet_transf(self.df.iloc[:, 3], "haar", 1, 5)
figure()
subplot(7, 1, 1)
plot(self.df.iloc[:, 3], 'r', linewidth=1, label=self.comboBox.currentText())
grid()
legend(loc='best')
subplot(7, 1, 2)
plot(self.coeffs_lvl[-1][0], 'b', linewidth=1, label='cA,level-' + self.lineEdit_2.text())
grid()
legend(loc='best')
for l in range(5):
    subplot(7, 1, 7-l+2)
    plot(self.coeffs_lvl[l][1], 'g', linewidth=1, label='cD,level-%s' %(l+1))
    grid()
    legend(loc='best')
show()

def wavelet_transf(self, x, wavelet, order, lvl):
    strWalevet = wavelet
    if wavelet != "haar":
        strWalevet += str(order)

    self.coeffs_lvl=[]
    self.coeffs_d_lvl=[]

    for l in range(lvl):
        self.coeffs_lvl.append(wavedec(x, strWalevet, level=l+1,mode="per"))
        self.coeffs_d_lvl.append(self.coeffs_lvl[l][1])

```

На рис. 3.10 відображена повна декомпозиція після застосування ДВП. А на рис. 3.11 відображено спектр коефіцієнтів деталізації, отримані ДВП.

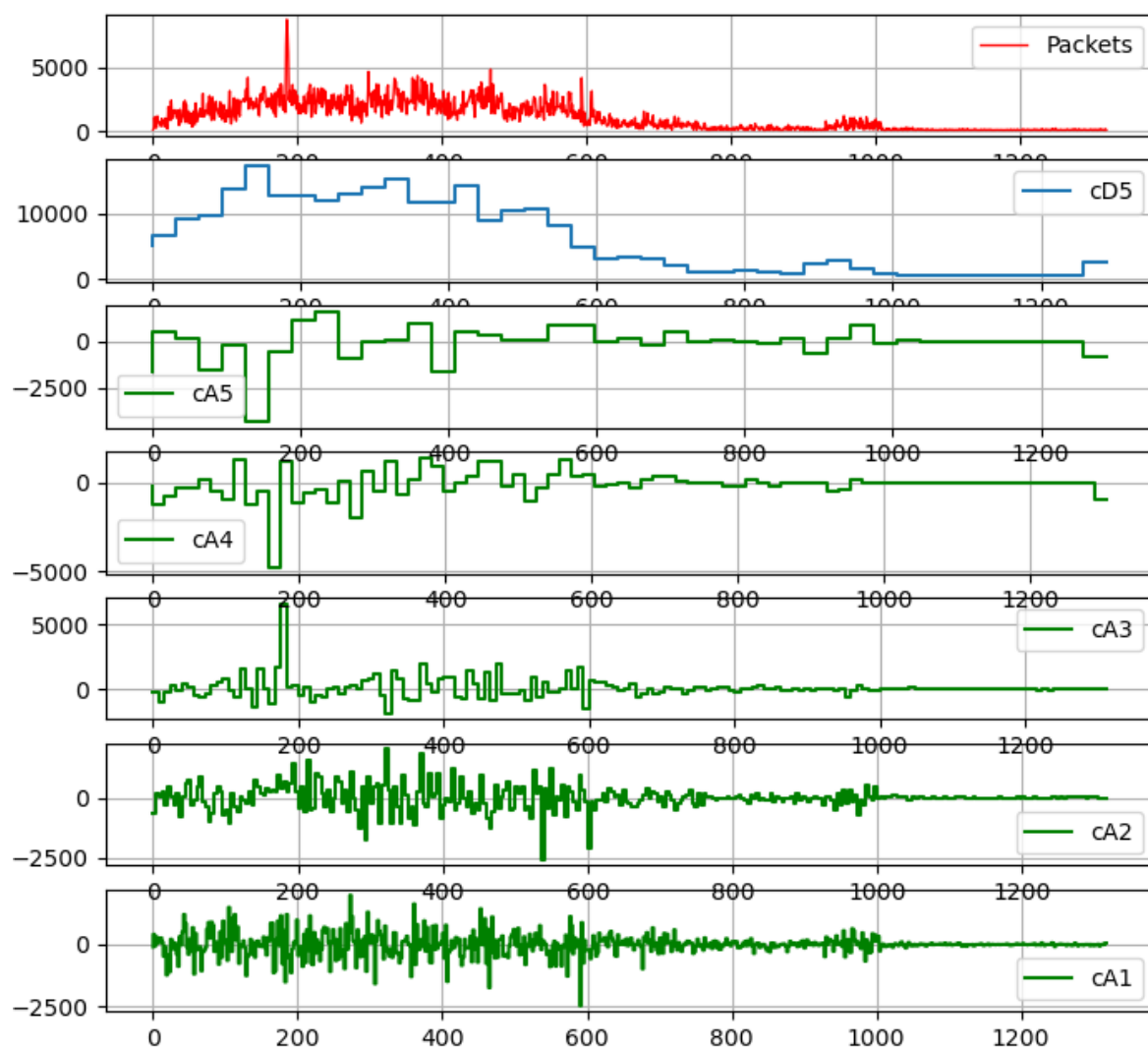


Рисунок 3.10 – DWT, повна декомпозиція

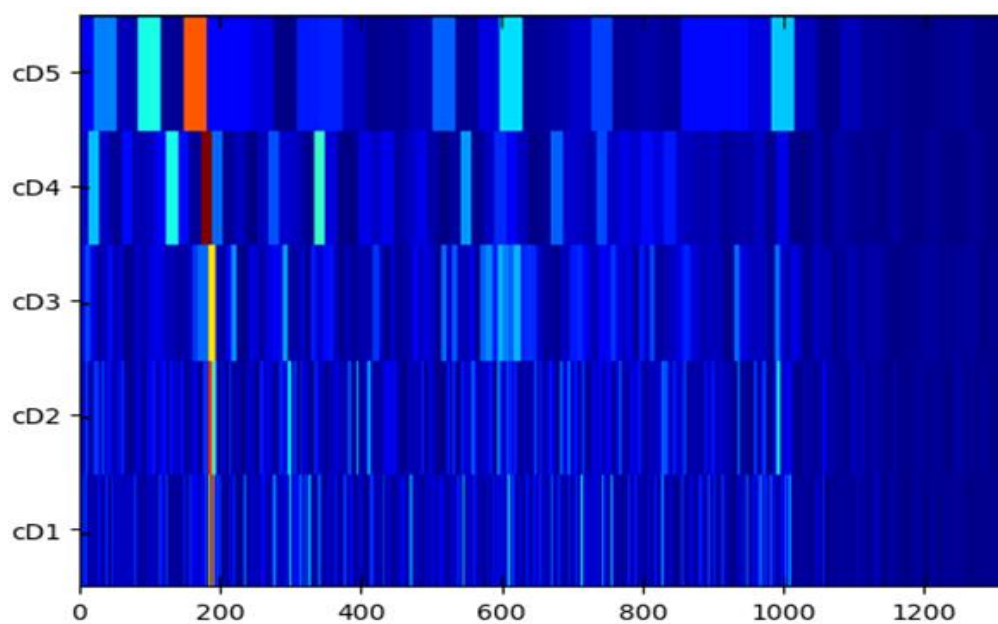


Рисунок 3.11 – Коефіцієнти деталізації у колірних діапазонах

### 3.3.3 Розробка графічних елементів програми

Для створення графічних вікон ми використовуємо середовище Qt Creator для створення інтерфейсу шляхом об'єднання графічних елементів (рис. 3.12). Після чого за допомогою плагіна “pyuic5-tool” набираємо команду для перетворення на Python файл:

```
pyuic5 <name_file>.ui -o <name_file>.py
```

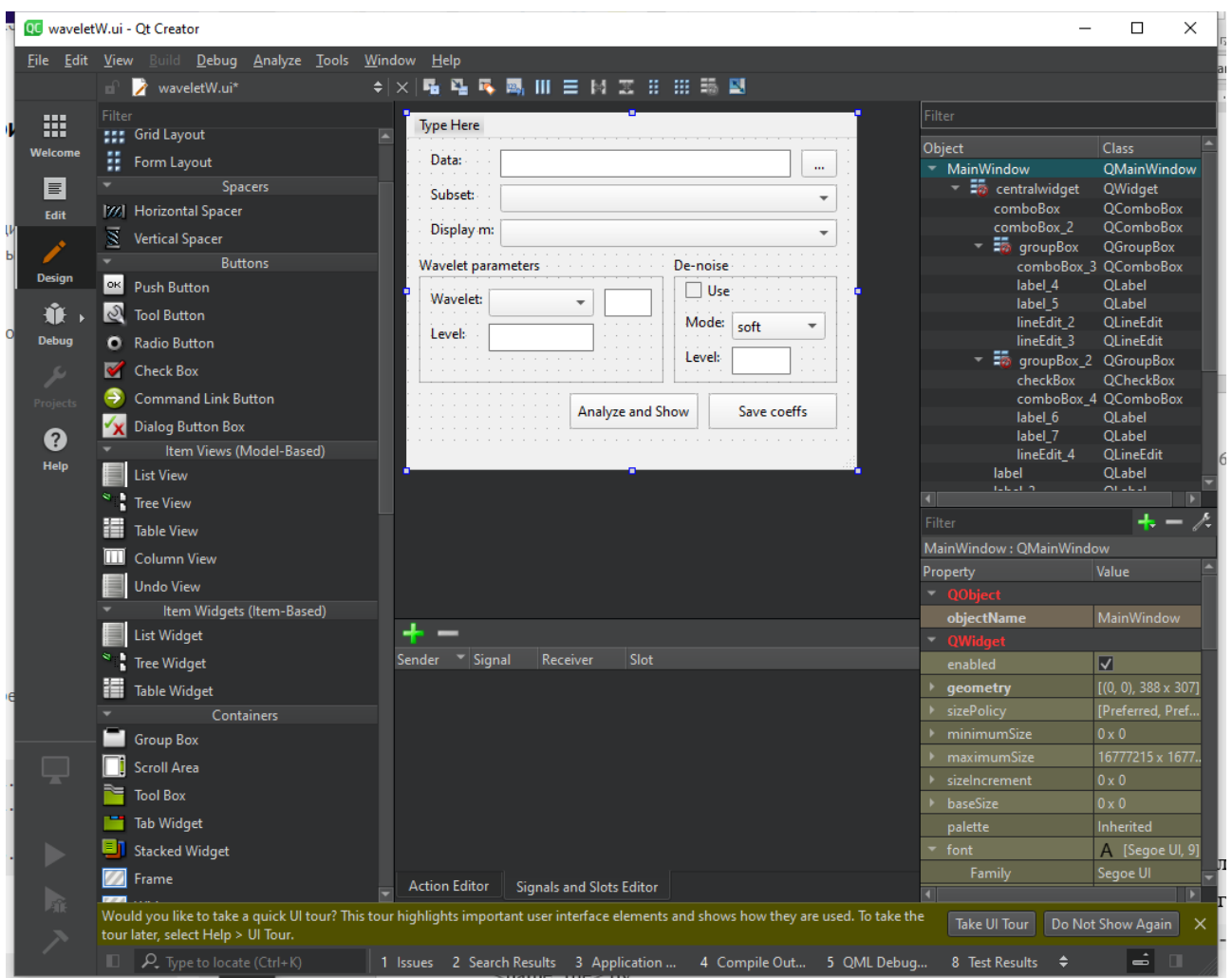


Рисунок 3.12 – Створення інтерфейсу в Qt Creator

Після успішного перетворення Python код інтерфейсу наведено у лістингу 3.5. Також є можливість редагувати інтерфейс програмно на Python без графічного редактора.

### Лістинг 3.5 – Python код інтерфейсу

```

from PyQt5 import QtCore, QtGui, QtWidgets

class Ui_MainWindow(object):
    def setupUi(self, MainWindow):
        MainWindow.setObjectName("MainWindow")
        MainWindow.resize(361, 284)
        self.centralwidget = QtWidgets.QWidget(MainWindow)
        self.centralwidget.setObjectName("centralwidget")
        self.pushButton = QtWidgets.QPushButton(self.centralwidget)
        self.pushButton.setGeometry(QtCore.QRect(310, 10, 31, 24))
        self.pushButton.setObjectName("pushButton")
        ...

        self.retranslateUi(MainWindow)
        QtCore.QMetaObject.connectSlotsByName(MainWindow)

    def retranslateUi(self, MainWindow):
        _translate = QtCore.QCoreApplication.translate
        MainWindow.setWindowTitle(_translate("MainWindow", "Wavelet 1-D"))
        self.pushButton.setText(_translate("MainWindow", "..."))
        self.label.setText(_translate("MainWindow", "Data:"))
        ...

if __name__ == "__main__":
    import sys
    app = QtWidgets.QApplication(sys.argv)
    MainWindow = QtWidgets.QMainWindow()
    ui = Ui_MainWindow()
    ui.setupUi(MainWindow)
    MainWindow.show()
    sys.exit(app.exec_())

```

Таким чином, ми створюємо кілька графічних вікон – головне вікно, вікно обробки даних, вікно для застосування вейвлет-аналізу та візуалізації різних даних.

### 3.4 Висновки за розділом 3

У цьому розділі було виконано конвертацію мережевого трафіку з tcpdump в CSV файл, виключивши непотрібні властивості. Далі був розроблений кожен компонент системи, а саме підсистема генерації ознак мережного трафіку, підсистема застосування вейвлет-аналізу та інтерфейс програми користувача.

## **4 ЕКСПЛУАТАЦІЯ, ТЕСТУВАННЯ ТА ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ ПРОГРАМИ**

### **4.1 Експлуатація програмного забезпечення**

#### **4.1.1 Призначення програми**

Основним призначенням розробленої програми є виявлення атак у мережах за допомогою методів вейвлет-аналізу.

#### **4.1.2 Можливості програми**

Програма дозволяє:

- завантажити мережний трафік у вигляді файлу CSV. Цей CSV файл має бути відфільтрований та експортований з вибраними властивостями програмою Wireshark;
- перетворити мережевий трафік у потрібному вигляді для аналізу. Тут обчислюються кілька функцій щохвилини. Функції можуть бути декількома (байтів/пакетів/потоків за хвилину, середня кількість байтів/пакетів у потоці за хвилину тощо);
- мати можливість переглядати візуалізації різних сигналів (початкового та обробленого);
- виконувати вейвлет-перетворення із заданими різними параметрами (вейвлет, порядок та рівень);
- зберігати значення (коефіцієнти), отримані вейвлет-перетворенням, для подальшої роботи (будова моделей та створення ARX).
- мати можливість переглядати різні візуалізації даних, отриманих вейвлет-перетворенням (коефіцієнти апроксимації та деталізації);
- мати можливість аналізувати потоки у вихідному мережевому трафіку за вибраний період щодо різких амплітудів вейвлет-перетворення та прийняти рішення про вторгнення атак.

### 4.1.3 Мінімальний склад технічних та програмних засобів

Для роботи системи необхідний персональний комп'ютер з характеристиками не нижче:

- процесор з тактовою частотою 1.5 ГГц або вище;
- оперативна пам'ять об'ємом 2 Гб або вище;
- 1 Гб вільного місця на диску;
- будь-який графічний адаптер;
- мережевий інтерфейс Ethernet 10/100/100 Base-TX.

Застосунком можна користуватися у будь-яких операційних системах, які підтримують Python.

### 4.1.4 Виконання програми

Перед застосуванням вейвлет-перетворенням та аналізу мережного трафіку, необхідно витягувати потрібну інформацію із записаного мережного трафіку .pcrdump у потрібному форматі для програми. Дані повинні містити щонайменше такі ознаки:

- порядковий номер;
- час від початку захоплення (у секундах);
- розмір пакета (байт);
- протокол;
- адреса відправника;
- порт відправника;
- адреса одержувача;
- порт отримувача.

Перетворений мережевий трафік представляється у вигляді CSV файлу для завантаження у програму.

Далі потрібно перетворити в іншому вигляді, де містить потрібні ознаки (кільк. байтів, пакетів, потоків, кільк. байтів та пакетів у потоці і т.д. за заданий

часовий інтервал) для загального та кожного протоколу (TCP, UDP, ICMP). Для цього в програмі потрібно вибрати Data transformation, після чого відкриється вікно, в якому потрібно завантажувати вихідний файл і вибрати шлях для збереження нового перетвореного файлу і натиснути кнопку Start processing (рис. 4.1).

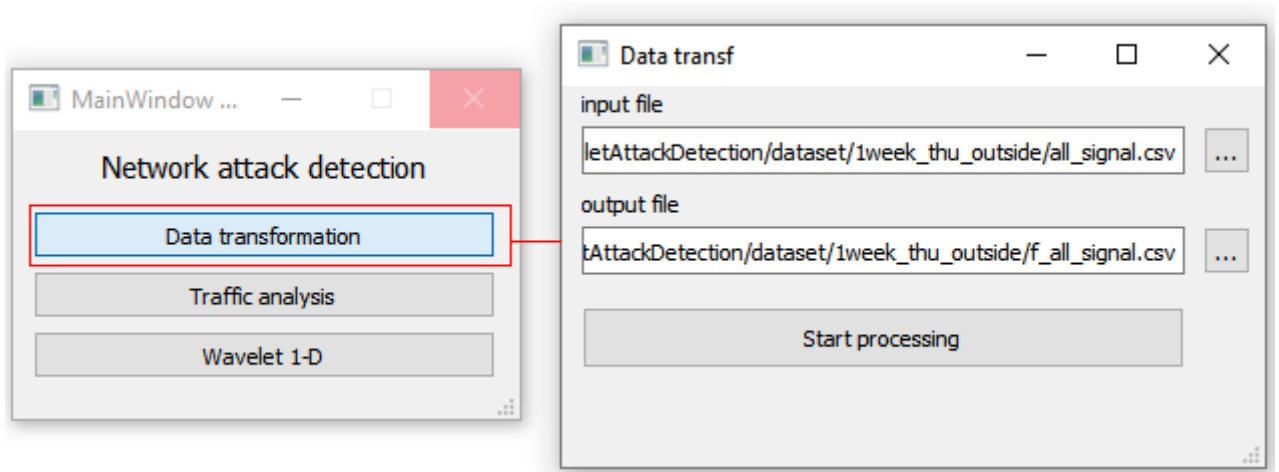


Рисунок 4.1 – Вікно попередньої обробки даних

Після чого виходить підготовлений файл, вміст якого представляється на рис. 4.2.

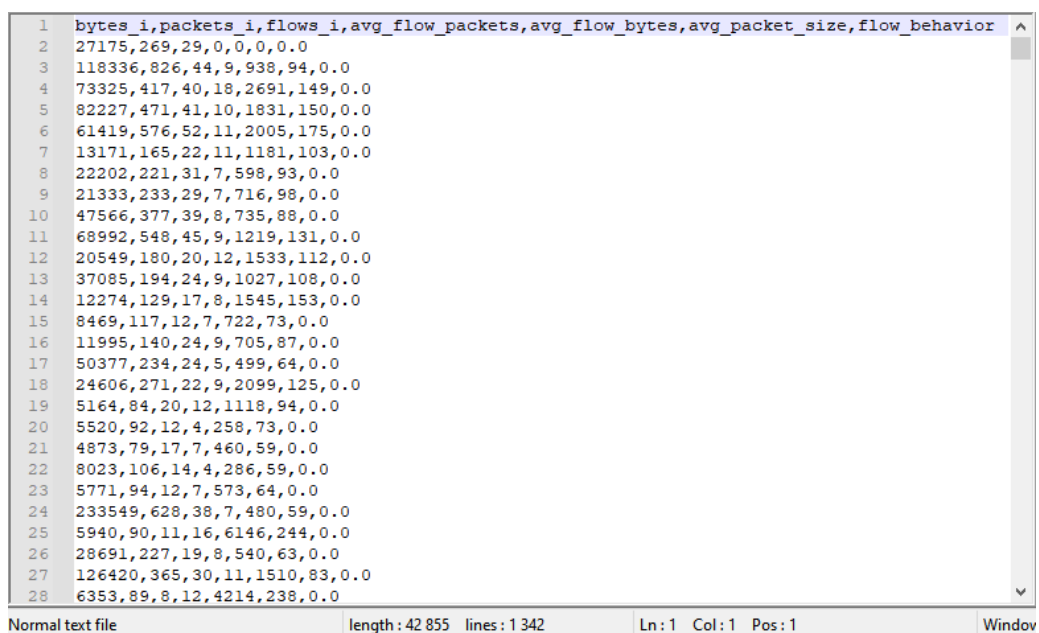


Рисунок 4.2 – Вміст підготовленого файлу

Після етапу попередньої обробки даних можна розпочати роботу з використанням вейвлет-аналізу та візуалізації даних. Для цього вибирається пункт Wavelet 1-D у головне меню, після чого відкривається вікно (рис. 4.3). У цьому вікні є можливості: завантажити дані, вибрати ознаку, тип завдання, параметри вейвлет-аналізу, відобразити графіки та зберігати коефіцієнти.

Перш ніж виконувати завдання, потрібно завантажити дані, для цього потрібно натиснути на кнопку "...", після чого відкриється вікно (рис. 4.4), в якому потрібно вибрати файл.

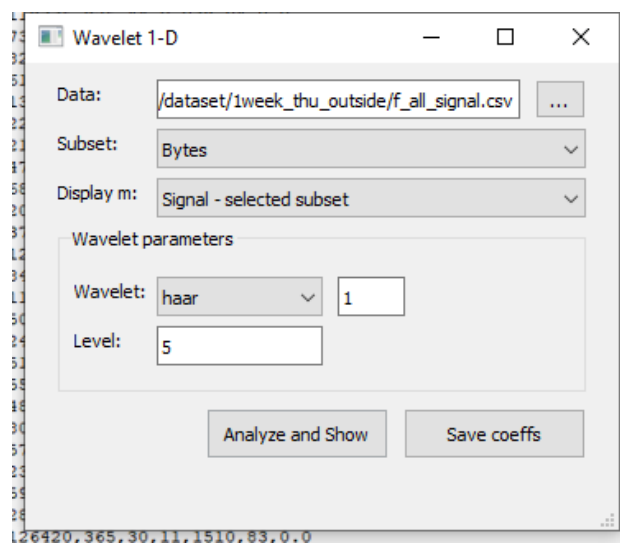


Рисунок 4.3 – Вікно застосування вейвлет перетворення

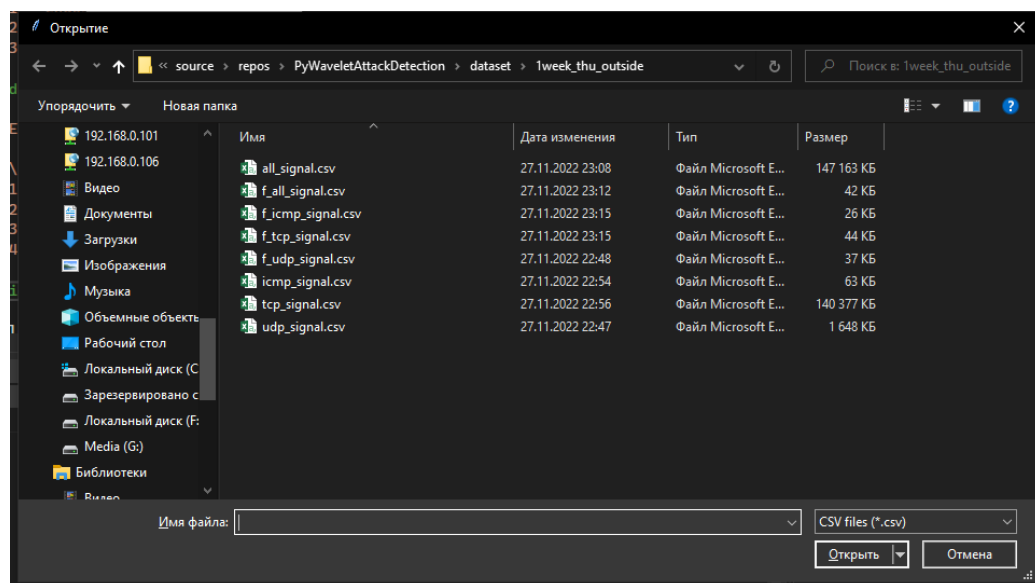


Рисунок 4.4 – Вікно вибору перетворених даних

Далі можна вибрати потрібну ознаку для аналізу (рис. 4.5).

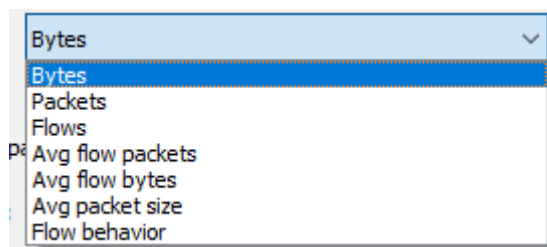


Рисунок 4.5 – Вибір ознаки

Після вибору ознаки можна вибрати тип візуалізації – вихідний сигнал вибраної ознаки, сигнали всіх ознак, ДВП (повна декомпозиція), ДВП (коефіцієнти деталізації), БВП (рис. 4.6).

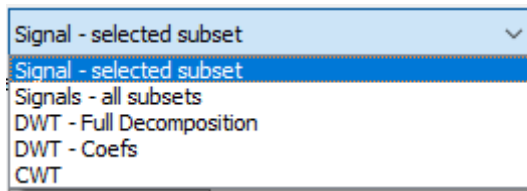


Рисунок 4.6 – Вибір типу візуалізації

Усі візуалізації у цій програмі мають бути відображені так, як на рис. 4.7.

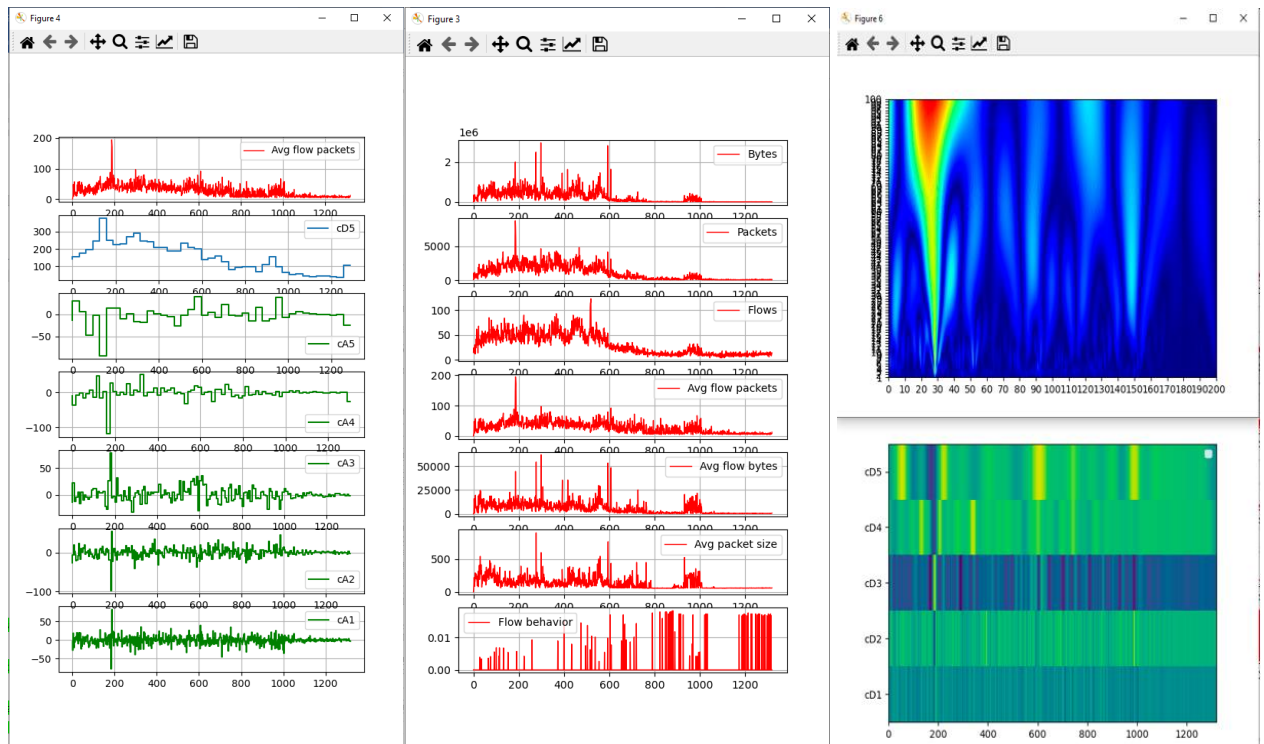


Рисунок 4.7 – Візуалізації

#### 4.1.5 Повідомлення користувачу

На даний момент існує кілька повідомлень, які можуть надаватися при виконанні різних функцій:

- «Неправильний формат файлу»;
- «Неможливо читати файл CSV, можливо, він пошкоджений»;
- «Операцію успішно завершено»;
- «Не вдалося виконати операцію».

#### 4.2 Тестування функціональності програми

При проходженні чек-листів встановлюється статус навпроти кожного протестованого пункту. Після проведення тестування системи були отримані всі результати, які пояснюють про те, що підсистема працює нормально без проблем і не викликається якийсь збій в системі.

У таблиці 4.1 представлений заповнений чек-лист після проходження тестування функціональності.

Таблиця 4.1 – Чек-лист після проходження тестування функціональності

| Чек-лист функціонального тестування               |                                                                                     |        |
|---------------------------------------------------|-------------------------------------------------------------------------------------|--------|
| №                                                 | Керування процесом                                                                  |        |
| 1                                                 | Запуск програми                                                                     | Passed |
| 2                                                 | Завершення програми                                                                 | Passed |
| 3                                                 | Дублювання процесів програми                                                        | Passed |
| 4                                                 | Спроба завершення процесу під час обробки                                           | Passed |
| Попередня обробка даних                           |                                                                                     |        |
| 5                                                 | Завантаження мережного трафіку у вигляді CSV файлу                                  | Passed |
| 6                                                 | Спроба завантаження пошкодженого файлу                                              | Passed |
| 7                                                 | Спроба завантаження файлу з іншою структурою                                        | Passed |
| 8                                                 | Перетворення даних у потрібному форматі (генерація ознак) з валідними параметрами   | Passed |
| 9                                                 | Перетворення даних у потрібному форматі (генерація ознак) з невалідними параметрами | Passed |
| 10                                                | Збереження оброблених даних у файл                                                  | Passed |
| Застосування вейвлет-перетворення та шумоусунення |                                                                                     |        |
| 11                                                | Завантаження оброблених даних                                                       | Passed |
| 12                                                | Вибір ознаки в якості вхідного сигналу                                              | Passed |
| 13                                                | Вибір типу відображень                                                              | Passed |
| 14                                                | Вибір вейвлет-функції з валідними параметрами                                       | Passed |
| 15                                                | Вибір вейвлет-функцій з невалідними параметрами                                     | Passed |
| 16                                                | Шумоусунення з різними параметрами                                                  | Passed |
| 17                                                | Застосування вейвлет-перетворення                                                   | Passed |
| 18                                                | Збереження коефіцієнтів                                                             | Passed |
| Візуалізація та аналіз                            |                                                                                     |        |
| 19                                                | Візуалізація всіх ознак (сигналів) мережного трафіку                                | Passed |
| 20                                                | Візуалізація вейвлет-коефіцієнтів (апроксимації та деталізації)                     | Passed |
| 21                                                | Аналіз мережевого трафіку за заданим інтервалом                                     | Passed |

### 4.3 Експериментальне дослідження

Після обробки даних ми розпочали експерименти. На рис. 4.8 відображено мережевий трафік (послідовність байтів за хвилину), який записано 22 години на добу, із зазначеними типами атак. У цьому випадку є 9 різних атак.

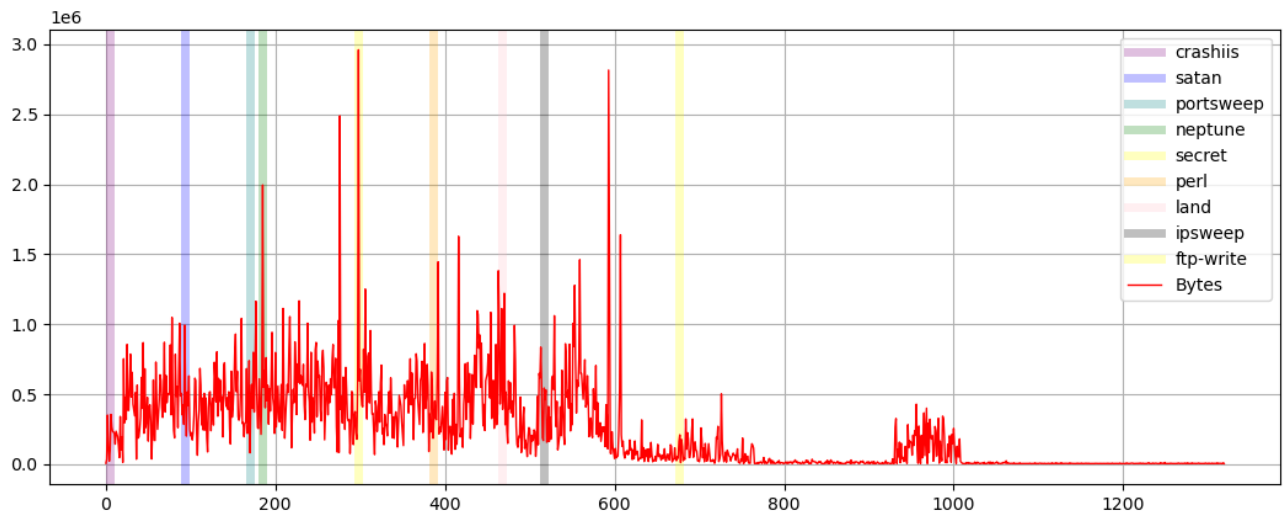


Рисунок 4.8 – Мережевий трафік (кількість байтів за 1 хвилину) із зазначеними атаками

Як видно з рисунку, що деякі атаки з першого погляду не помічені на одному сигналі, тому з вхідного сигналу було згенеровано кілька ознак (bytes, packets, flows, avg flow size тощо) з різними протоколами (TCP, UDP, ICMP). На рис. 4.9 відображено два мережеві трафіку ICMP-протоколу, записані в різні дні, — з атаками та без атак відповідно.

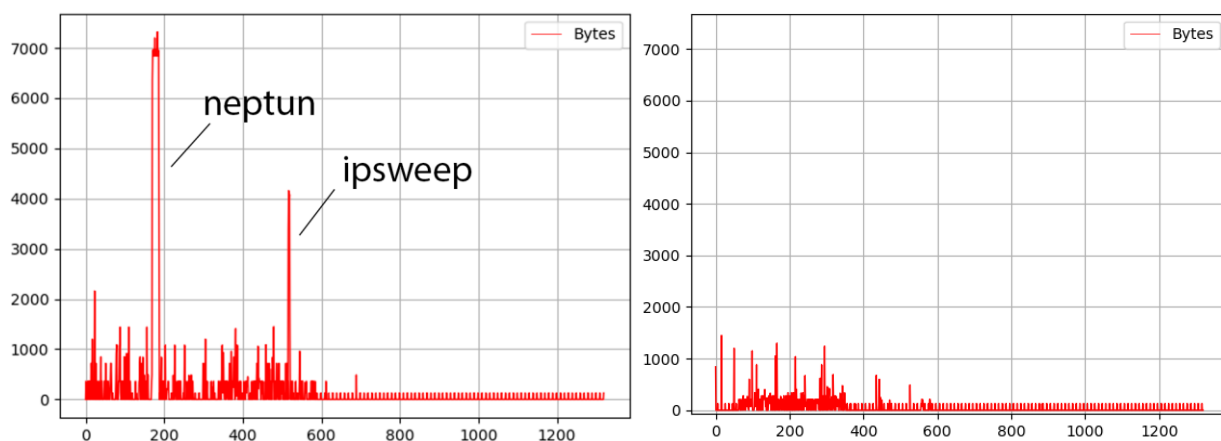


Рисунок 4.9 – Мережеві трафіки ICMP з атаками (ліворуч) та без атак (праворуч)

А на рис. 4.10-4.12 відображено сигнали кожної ознаки кожного протоколу. Одні атаки помічені на сигналах тих ознак, інші атаки помічені на сигналах інших ознак.

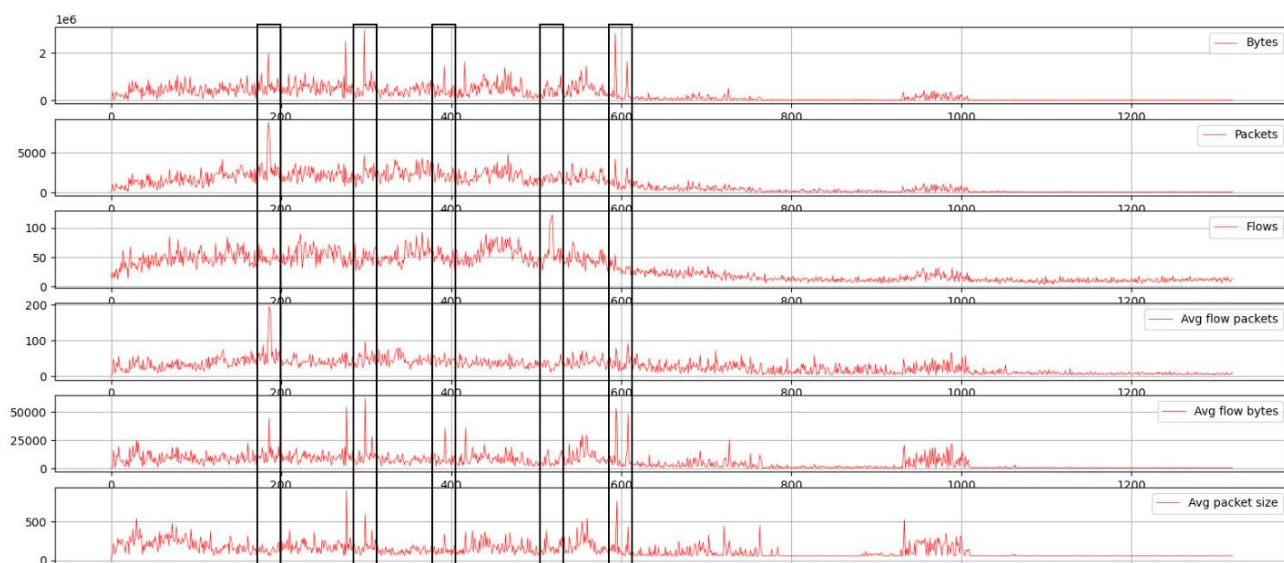


Рисунок 4.10 – Сигнали кожної ознаки TCP-протоколу

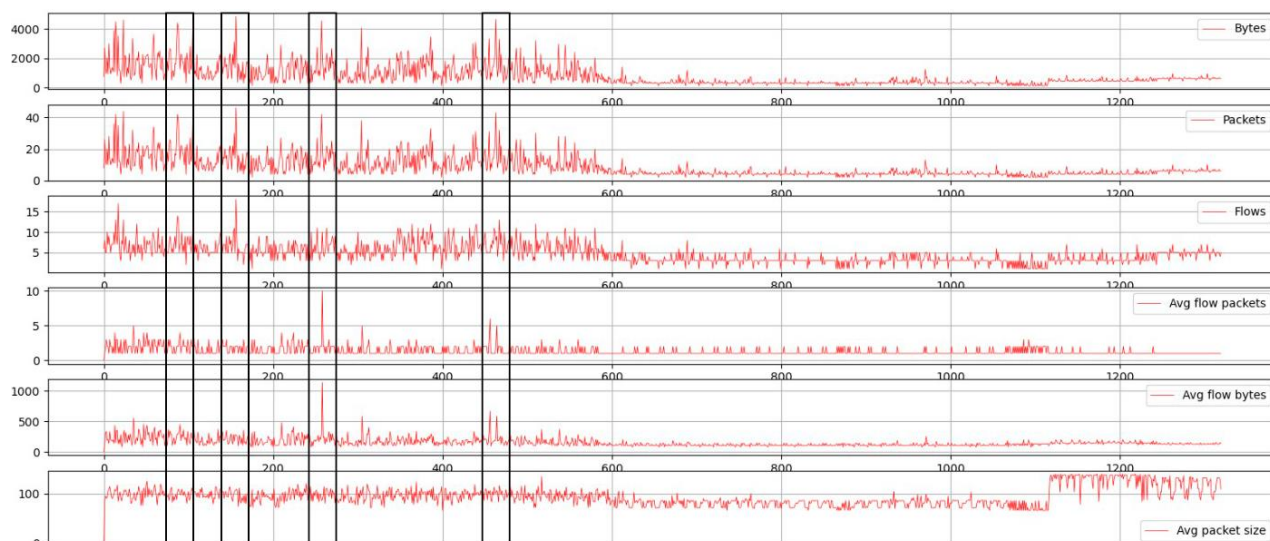


Рисунок 4.11 – Сигнали кожної ознаки UDP-протоколу

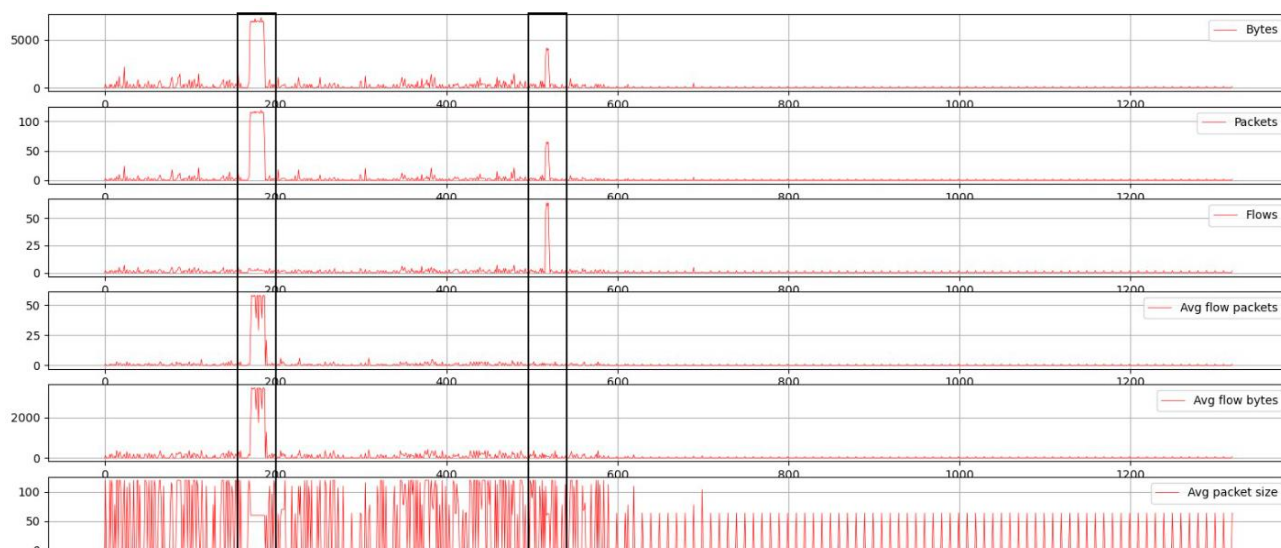


Рисунок 4.12 – Сигнали кожної ознаки ICMP-протоколу

У таблиці 4.2 показано, який протокол використовує кожна атака. Атаки portswep та neptune використовують протоколи TCP, ICMP, тому добре помічені на графіках таких протоколів, а атака ipsweep не помічена, оскільки використовує лише протокол ICMP.

Таблиця 4.2 – Атаки та протоколи

| <b>Атака</b> | <b>TCP</b> | <b>UDP</b> | <b>ICMP</b> |
|--------------|------------|------------|-------------|
| portswep     | +          | -          | +           |
| Neptune      | +          | -          | +           |
| Secret       | +          | +          | -           |
| Perl         | +          | +          | -           |
| ipsweep      | -          | -          | +           |

А в таблиці 4.3 показано, за якою ознакою можна виявити атаки. Атаки portswep, Neptune та perl відправляють велику кількість пакетів максимального розміру, тому легко виявляються на графіках кількості байтів та пакетів за хвилину. Атака secret посилає невелику кількість пакетів, але максимального розміру. Атака ipsweep групою (з кількома IP) відправляє велику кількість потоків з великою кількістю пакетів великого розміру, тому атака добре помічена на всіх ознаках протоколу ICMP.

Таблиця 4.3 – Ознаки кожної атаки

| <b>Атака</b> | <b>Bytes</b> | <b>Packets</b> | <b>Flows</b> |
|--------------|--------------|----------------|--------------|
| portswep     | +            | +              | -            |
| Neptune      | +            | +              | -            |
| Secret       | +            | -              | -            |
| Perl         | +            | +              | -            |
| ipsweep      | +            | +              | +            |

Застосувавши ДВП, на рис. 4.13 відображена повна декомпозиція, в якій кілька вікон – сигнал, коефіцієнти апроксимації останнього рівня та коефіцієнти деталізації на всіх рівнях розкладання. Тут використовується вейвлет Хаара для двох сигналів – TCP (Packets) та ICMP (Avg flow packets). Як видно з рисунку, аномалії добре помічені на деяких рівнях, де виділено рамки.

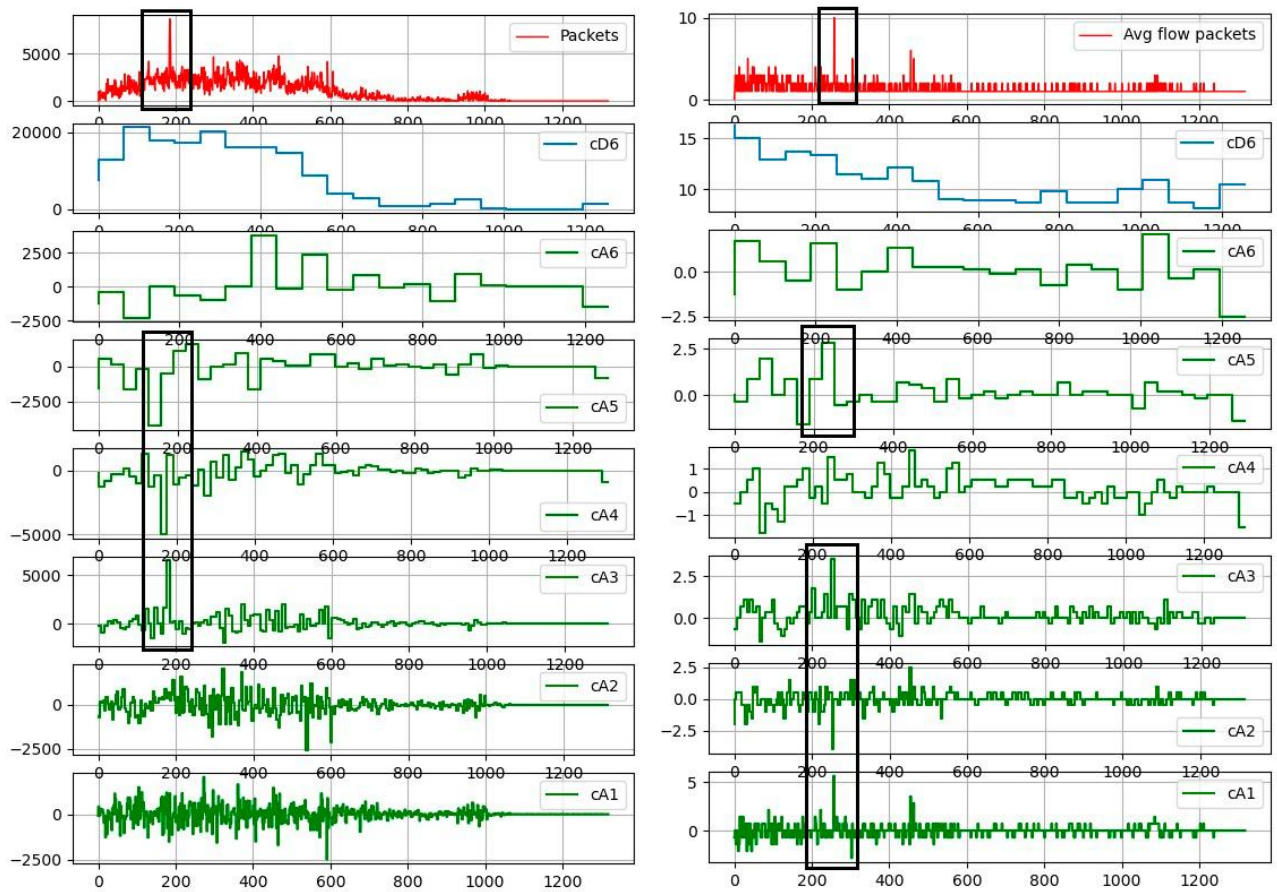


Рисунок 4.13 – Вейвлет Хаара за різними рівнями розкладання сигналу TCP (пакетів в минуту) та сигналу UDP (середня кількість пакетів у потоці за хвилину) відповідно

Після застосування 5 різних вейвлетів були отримані результати, які занесені в таб. 4.4-4.5, де для кожного вейвлета кожного рівня деталізації (від 1 до 5) встановлена оцінка виявлення атак. Для оцінки існують три типи значень:

- «-» – аномалії не помічені на відповідному рівні;
- «+» – аномалії важко помічені чи не всі аномалії помічені;
- «++» – всі аномалії добре помічені на відповідному рівні.

Таблиця 4.4 – Оцінки виявлення атаки neptune у детальних вікнах (TSP, packets)

| Рівень розкладання | Хаара | Добеші-2 | Добеші-4 | Койфлет-1 | Симплет-4 |
|--------------------|-------|----------|----------|-----------|-----------|
| 1                  | -     | -        | -        | -         | -         |
| 2                  | -     | ++       | +        | ++        | ++        |
| 3                  | ++    | ++       | ++       | ++        | ++        |
| 4                  | ++    | +        | +        | ++        | ++        |
| 5                  | +     | -        | ++       | +         | +         |

Таблиця 4.5 – Оцінки виявлення атаки secret у детальних вікнах (ICMP, avg flow packets)

| Рівень розкладання | Хаара | Добеші-2 | Добеші-4 | Койфлет-1 | Симплет-4 |
|--------------------|-------|----------|----------|-----------|-----------|
| 1                  | ++    | ++       | ++       | +         | +         |
| 2                  | ++    | ++       | -        | +         | ++        |
| 3                  | +     | -        | +        | ++        | ++        |
| 4                  | -     | +        | ++       | +         | -         |
| 5                  | +     | -        | -        | -         | -         |

Порівняльний аналіз характеристик достовірності виявлення аномалій показує, що тривалі за часом атаки, наприклад, neptune, легко виявляються на середніх і високих рівнях розкладання, але не виявляються на низьких рівнях розкладання. А короткочасні атаки, навпаки, легко виявляються на низьких і середніх частотах, наприклад, атака secret.

Проте характеристики виявлення аномалій можуть залежати як від тривалості атак, а й від розміру вікна. У нашому випадку розмір вікна дорівнює 1 хвилині, а загальна довжина сигналу – 22 год. Якщо, наприклад, розмір вікна дорівнює 5 секунд, то для аналізу довелося б працювати з великою кількістю рівнів розкладання (10 або більше).

Для подальшого вдосконалення запропоновано такі шляхи:

- реалізувати підсистему автоматичного перетворення записаного мережного трафіку на потрібний формат. Особливо корисно для автоматичного виявлення атак;
- збільшення кількості набору даних. Чим більше набору даних, тим більше інформації про різні атаки, відповідно тим вище вірогідність правильного виявлення атак;
- паралельне використання кількох вейвлет-функцій;
- застосування вейвлет-аналізу для кількох вікон різного розміру (1 с., 10 с., 30 с., 1 мин.).

#### **4.4 Висновки за розділом 4**

У цьому розділі були розглянуті призначення, функції програми, умови виконання програми та інструкції щодо використання програми. Далі було проведено функціональне тестування програми, у результаті програма працює правильно без збоїв. Крім того було виконано експериментальне дослідження та отримані результати, а також запропоновано покращення системи.

## ВИСНОВКИ

У процесі виконання дипломної кваліфікаційної роботи магістра було виконано такі завдання:

- аналіз актуальних проблем з інформаційною безпекою, аналіз різновидів мережових аномалій та атак;
- аналіз існуючих методів виявлення мережових аномалій та порівняння між методами;
- аналіз існуючих способів застосування вейвлет-аналізу виявлення мережових атак;
- аналіз безперервного і дискретного вейвлет-перетворення, вейвлет функцій та їх порівняння;
- розробка архітектури системи, вибір мови програмування та необхідних бібліотек, а також набору даних (мережові трафіки з поміченими атаками та нормальні трафіки без атак);
- обробка набору даних та реалізація компонентів системи – компонент для генерації ознак мережного трафіку, компонент для шумоусунення сигналу, отримання вейвлетів коефіцієнтів за допомогою ДВП, візуалізації даних та інтерфейс програми;
- тестування програмного забезпечення та експериментальне дослідження.

Для виконання та тестування програми для виявлення мережових атак з використанням вейвлет-аналізу була взята частина набору даних DARPA, яка включає добовий мережовий трафік з різними типами атак і нормальний мережовий трафік без атак. Для підвищення ймовірності правильного виявлення мережових атак було згенеровано кілька ознак, таких як кількість байтів, пакетів, потоків за хвилину, середній розмір пакетів і потоків за хвилину. Завдяки згенерованим ознакам майже всі відомі атаки добре були виявлені методом вейвлет-аналізу.

Крім того, були запропоновані шляхи для вдосконалення, які можуть допомогти досягати найкращі результати виявлення мережових атак, а саме збільшити ймовірність правильного виявлення та зменшити ймовірність помилкового виявлення.

Для виконання шумозаглушення сигналу, виявлення мережових атак мережевого трафіку з використанням різних вейвелт-функцій був використана бібліотека PyWavelets на мові програмування Python.

## ПЕРЕЛІК ПОСИЛАНЬ

1. Чемерис К.М. Застосування методу вейвлет-аналізу для виявлення атак в мережах /К.М. Чемерис, Л.Ю. Дейнега // Наука і техніка Повітряних Сил Збройних Сил України. - 2022. - № 1(46). - С. 99-107.
2. An Evolving Threat Landscape? 2022 Cyber Attack Statistics, Data, and Trends [Electronic recourse]. - Access mode: <https://complexdiscovery.com/an-evolving-threat-landscape-2022-cyber-attack-statistics-data-and-trends>.
3. Чемерис К. М., Дейнега Л. Ю. Виявлення несанкціонованих дій і атак в мережах методом вейвлет-аналізу [Електронний ресурс] //Тиждень науки-2022. Тези доповідей науково-практичної конференції, Запоріжжя, 18–22 квітня 2022 р. / Редкол. :В. В. Наумик (відпов. ред.) Електрон. дані. – Запоріжжя : НУ «Запорізька політехніка», 2022. – 1 електрон. опт. диск (DVD-ROM); 12 см. – Назва з тит. екрана. – С. 159-161.
4. Шелухин О.И. Обнаружение вторжений в компьютерные сети (сетевые аномалии) [Текст] / О.И. Шелухин, Д.Ж. Сакалема, А.С. Филинова. – М: Горячая линия – Телеком, 2016. – 221 с.
5. Що таке мережева атака [Електрон. ресурс]. - Режим доступу: <https://uaeu.top/digital-online/shcho-take-merezheva-ataka.html>.
6. Що таке DDoS-атака? [Електрон. ресурс]. - Режим доступу: <https://cip.gov.ua/ua/faqs/sho-take-ddos-ataka>.
7. What is a TCP SYN Flood [Electronic recourse]. - Access mode: <https://www.imperva.com/learn/ddos/syn-flood/>
8. IP Spoofing & Spoof Attacks [Electronic recourse]. - Access mode: <https://www.kaspersky.com/resource-center/threats/ip-spoofing>.
9. Жилін. А.В. Технології захисту інформації в інформаційно-телекомунікаційних системах : навч. посіб. / А. В. Жилін, О. М. Шаповал, О. А. Успенський ; ІСЗЗІ КПП ім. Ігоря Сікорського. – Київ : КПП ім. Ігоря Сікорського, Політехніка, 2021. – 213 с.

10. Рубан І.В. Класифікація методів виявлення аномалій в інформаційних системах [Текст] /І.В. Рубан, В.О. Мартовицький, С.О. Партика // Системи озброєння і військова техніка.. – 2016. – № 3(47). – С. 100-105.
11. Early Detection of Network Attacks Using Deep Learning [Electronic recourse] / Tanwir Ahmad, Dragos Truscan, Juri Vain, Ivan Porre. – Access mode: <https://arxiv.org/pdf/2201.11628.pdf>.
12. Твердохліб Ю. В. Методи та інформаційна технологія комплексного оцінювання параметрів вейвлет-перетворення нестационарних сигналів: автореф. дис, ... канд. техн. наук: 05.13.06 / Ю.В. Твердохліб. – Запоріжжя:ЗНТУ, 2018. – 178 с.
13. Wavelet Transform [Electronic recourse]. - Access mode: <http://gwyddion.net/documentation/user-guide-en/wavelet-transform.html>.
14. Gao Robert X. Wavelets: Theory and Applications for Manufacturing [Text] / Robert X. Gao, Ruqiang Yan. – Springer Science & Business Media, 2010. – 226 p.
15. Dong Wei, B.E., M.S. Coiflet-type wavelets: theory, design, and applications [Electronic recourse] / B.E., M.S. Dong Wei. – 1998. – 139 p. – Access mode: [https://users.ece.utexas.edu/~bevans/students/phd/dong\\_wei/phd.pdf](https://users.ece.utexas.edu/~bevans/students/phd/dong_wei/phd.pdf)
16. Wavelet methods for the detection of anomalies and their application to network traffic analysis [Electronic recourse] / D. Ko K. Kwon, M. Vannucci, A.N. Reddy, S. Kim // Quality and Reliability Engineering International. – 2006. – Vol. 22, Is. 8. – P. 953-969. - Access mode: <https://doi.org/10.1002/qre.781>.
17. Соловьев Н. А. Обнаружение вторжений на основе вейвлет-анализа сетевого трафика / Н. А. Соловьев, Н. А. Тишина, И. Г. Дворовой //Вестник УГАТУ. – 2010. – Т. 14. – №5(40). – С. 188–194.
18. Фактиева Р. Р. Применение вейвлетов Добеши для обнаружения аномалий сетевого трафика / Р. Р. Фактиева // Вестник БГУ. – 2013. – №2. – С. 88-94.

19. A signal analysis of network traffic anomalies / P. Barford, J. Kline, D. Plonka, A. Ron // Proceedings of the 2nd ACM SIGCOMM Workshop on Internet measurement. – 2002. – P. 71–82.

20. Lu Wei. Network Anomaly Detection Based on Wavelet Analysis [Electronic resource] / Wei Lu, Ali A. Ghorbani. – Access mode: [https://www.researchgate.net/publication/220057667\\_Network\\_Anomaly\\_Detection\\_Based\\_on\\_Wavelet\\_Analysis](https://www.researchgate.net/publication/220057667_Network_Anomaly_Detection_Based_on_Wavelet_Analysis).

21. Python [Электрон. ресурс]. - Режим доступа: <https://uk.wikipedia.org/wiki/Python>.

22. Matlab [Electronic resource]. - Access mode: <https://www.mathworks.com/products/matlab.html>.

23. What is R? [Electronic resource]. - Access mode: <https://www.r-project.org/about.html>.

24. 1999 DARPA Intrusion Detection Evaluation Dataset | MIT Lincoln Laboratory [Electronic resource]. - Access mode: <https://www.ll.mit.edu/r-d/datasets/1999-darpa-intrusion-detection-evaluation-dataset>.

25. MIT Lincoln Laboratory: DARPA Intrusion Detection Evaluation [Electronic resource]. - Access mode: [https://archive.ll.mit.edu/ideval/docs/detections\\_1999.html](https://archive.ll.mit.edu/ideval/docs/detections_1999.html).

26. What is ICMP? [Electronic resource]. - Access mode: <https://www.cloudflare.com/learning/ddos/glossary/internet-control-message-protocol-icmp/>.

**ДОДАТОК А**  
**Текст програми**

У лістингу А.1 наведено код підпрограми для попередньої обробки даних.

### Лістинг А.1 – Код підпрограми для попередньої обробки даних

```
from PyQt5 import QtCore, QtGui, QtWidgets
from tkinter import *
from tkinter import filedialog as fd
import csv
from pylab import *
from numpy import*
import pandas as pd

class Ui_MainWindow(object):
    def setupUi(self, MainWindow):
        MainWindow.setObjectName("MainWindow")
        MainWindow.resize(351, 190)
        self.centralwidget = QtWidgets.QWidget(MainWindow)
        self.centralwidget.setObjectName("centralwidget")
        self.label = QtWidgets.QLabel(self.centralwidget)
        self.label.setGeometry(QtCore.QRect(10, 0, 71, 16))
        self.label.setObjectName("label")
        self.label_2 = QtWidgets.QLabel(self.centralwidget)
        self.label_2.setGeometry(QtCore.QRect(10, 50, 71, 16))
        self.label_2.setObjectName("label_2")
        self.lineEdit = QtWidgets.QLineEdit(self.centralwidget)
        self.lineEdit.setGeometry(QtCore.QRect(10, 20, 301, 24))
        self.lineEdit.setObjectName("lineEdit")
        self.lineEdit_2 = QtWidgets.QLineEdit(self.centralwidget)
        self.lineEdit_2.setGeometry(QtCore.QRect(10, 70, 301, 24))
        self.lineEdit_2.setObjectName("lineEdit_2")
        self.pushButton = QtWidgets.QPushButton(self.centralwidget)
        self.pushButton.setGeometry(QtCore.QRect(10, 110, 301, 31))
        self.pushButton.setObjectName("pushButton")
        self.pushButton_2 = QtWidgets.QPushButton(self.centralwidget)
        self.pushButton_2.setGeometry(QtCore.QRect(320, 20, 24, 24))
        self.pushButton_2.setObjectName("pushButton_2")
        self.pushButton_3 = QtWidgets.QPushButton(self.centralwidget)
        self.pushButton_3.setGeometry(QtCore.QRect(320, 70, 24, 24))
        self.pushButton_3.setObjectName("pushButton_3")
        MainWindow.setCentralWidget(self.centralwidget)
        self.menubar = QtWidgets.QMenuBar(MainWindow)
        self.menubar.setGeometry(QtCore.QRect(0, 0, 351, 21))
        self.menubar.setObjectName("menubar")
        MainWindow.setMenuBar(self.menubar)
        self.statusbar = QtWidgets.QStatusBar(MainWindow)
        self.statusbar.setObjectName("statusbar")
        MainWindow.setStatusBar(self.statusbar)

        self.retranslateUi(MainWindow)
        QtCore.QMetaObject.connectSlotsByName(MainWindow)

    def retranslateUi(self, MainWindow):
        _translate = QtCore.QCoreApplication.translate
        MainWindow.setWindowTitle(_translate("MainWindow", "Data transf"))
        self.label.setText(_translate("MainWindow", "input file"))
        self.label_2.setText(_translate("MainWindow", "output file"))
        self.pushButton.setText(_translate("MainWindow", "Start processing"))
        self.pushButton_2.setText(_translate("MainWindow", "..."))
        self.pushButton_3.setText(_translate("MainWindow", "..."))
        self.pushButton_2.clicked.connect(lambda: self.loadFile())
        self.pushButton_3.clicked.connect(lambda: self.saveFile())
        self.pushButton.clicked.connect(lambda: self.transf())
```

```

#Завантаження файлу
def loadFile(self):
    self.file_name = fd.askopenfilename(filetypes=(("CSV files", "*.csv"),
        ("All files", "*.*")))
    self.lineEdit.setText(self.file_name)

#Збереження файлу
def saveFile(self):
    self.save_file_name = fd.asksaveasfilename(filetypes=(("CSV files", "*.csv"),
        ("All files", "*.*")))
    self.lineEdit_2.setText(self.save_file_name)

#Процес обчислення ознак
def transf(self):
    with open(self.file_name, mode='r') as csv_file:
        csv_reader = csv.DictReader(csv_file)
        #max_s = 79200
        max_s = 80400 #макс. кол. екземплярів
        max_i = int(max_s/60) #кол. інтервалів

        #кол. байтов за інтервал
        bytes_i = []
        bytes_i = [0 for i in range(max_i)]

        #кол. пакетов за інтервал
        packets_i = []
        packets_i = [0 for i in range(max_i)]

        #кол. потоків за інтервал
        flows_i = []
        flows_i = [0 for i in range(max_i)]

        #середнє кол. пакетов в потоке за інтервал
        avg_flow_packets = []
        avg_flow_packets = [0 for i in range(max_i)]

        #середнє кол. байтов в потоке за інтервал
        avg_flow_bytes = []
        avg_flow_bytes = [0 for i in range(max_i)]

        #середнє кол. байтов в пакеті за інтервал
        avg_packet_size = []
        avg_packet_size = [0 for i in range(max_i)]

        #отношение кол. потоків к байтам на пакет за інтервал
        flow_behavior = []
        flow_behavior = [0.0 for i in range(max_i)]

    temp_ip_i = []

    curr_i = 0
    last_i = 0;
    #процесс вычисления признаков - потоків
    for row in csv_reader:
        curr_i = int(int(float(row["Time"])/60));
        found = False
        for t_ip_i in temp_ip_i:
            if (t_ip_i[0] == row["Source addr"] and t_ip_i[1] == row["Dest addr"]):
                found = True
                t_ip_i[2] += 1
                t_ip_i[3] += int(row["Length"])
            if found == False:
                temp_ip_i.append([row["Source addr"], row["Dest addr"], 1, int(row["Length"])])
                flows_i[curr_i] += 1

```

```

packets_i[curr_i] += 1;
bytes_i[curr_i] += int(row["Length"]);

#вычисление признаков для каждого интервала
if curr_i != last_i:
    for t_ip_i in temp_ip_i:
        avg_flow_packets[curr_i] += t_ip_i[2]
        avg_flow_bytes[curr_i] += t_ip_i[3]
        avg_packet_size[curr_i] += int(float(avg_flow_bytes[curr_i])/avg_flow_packets[curr_i])
    avg_flow_packets[curr_i] = int(float(avg_flow_packets[curr_i])/len(temp_ip_i))
    avg_flow_bytes[curr_i] = int(float(avg_flow_bytes[curr_i])/len(temp_ip_i))
    avg_packet_size[curr_i] = int(float(avg_packet_size[curr_i])/len(temp_ip_i))
    flow_behavior[curr_i] = float(flows_i[curr_i]/avg_packet_size[curr_i])
    temp_ip_i.clear()
last_i = curr_i

#создание файла
with open(self.save_file_name, 'w', newline=") as file:
    writer = csv.writer(file)
    writer.writerow(['bytes_i', 'packets_i', 'flows_i',
'avg_flow_packets', 'avg_flow_bytes', 'avg_packet_size', 'flow_behavior'])

#добавление признаков в данные
data = []
for i in range(len(flows_i)):
    data.append([bytes_i[i], packets_i[i], flows_i[i], avg_flow_packets[i], avg_flow_bytes[i], avg_packet_size[i],
flow_behavior[i]])
#Запись данных в файл
writer.writerows(data)

if __name__ == "__main__":
    import sys
    app = QtWidgets.QApplication(sys.argv)
    MainWindow = QtWidgets.QMainWindow()
    ui = Ui_MainWindow()
    ui.setupUi(MainWindow)
    MainWindow.show()
    sys.exit(app.exec_())

```

У лістингу А.2 наведено код підпрограми для застосування вейвлет-перетворення

## Лістинг А.2 – Код підпрограми для застосування вейвлет-перетворення

```

from PyQt5 import QtCore, QtGui, QtWidgets
from tkinter import *
from tkinter import filedialog as fd
import csv
from pylab import *
from numpy import *
import pandas as pd
import pywt
from pywt import wavedec
from pywt import waverec
import statistics

class Ui_MainWindow(object):
    def setupUi(self, MainWindow):
        MainWindow.setObjectName("MainWindow")
        MainWindow.resize(361, 284)

```

```

self.centralwidget = QtWidgets.QWidget(MainWindow)
self.centralwidget.setObjectName("centralwidget")
...
self.pushButton = QtWidgets.QPushButton(self.centralwidget)
self.pushButton.setGeometry(QtCore.QRect(310, 10, 31, 24))
self.pushButton.setObjectName("pushButton")
self.lineEdit = QtWidgets.QLineEdit(self.centralwidget)
self.lineEdit.setGeometry(QtCore.QRect(80, 10, 221, 24))
self.lineEdit.setObjectName("lineEdit")
self.label = QtWidgets.QLabel(self.centralwidget)
self.label.setGeometry(QtCore.QRect(20, 10, 49, 16))
self.label.setObjectName("label")
self.comboBox = QtWidgets.QComboBox(self.centralwidget)
self.comboBox.setGeometry(QtCore.QRect(80, 40, 261, 24))
self.comboBox.setObjectName("comboBox")
self.label_2 = QtWidgets.QLabel(self.centralwidget)
self.label_2.setGeometry(QtCore.QRect(20, 40, 49, 16))
self.label_2.setObjectName("label_2")
self.label_3 = QtWidgets.QLabel(self.centralwidget)
self.label_3.setGeometry(QtCore.QRect(20, 70, 61, 16))
self.label_3.setObjectName("label_3")
self.comboBox_2 = QtWidgets.QComboBox(self.centralwidget)
self.comboBox_2.setGeometry(QtCore.QRect(80, 70, 261, 24))
self.comboBox_2.setObjectName("comboBox_2")
self.groupBox = QtWidgets.QGroupBox(self.centralwidget)
self.groupBox.setGeometry(QtCore.QRect(20, 100, 321, 101))
self.groupBox.setObjectName("groupBox")
self.label_4 = QtWidgets.QLabel(self.groupBox)
self.label_4.setGeometry(QtCore.QRect(10, 30, 61, 16))
self.label_4.setObjectName("label_4")
self.comboBox_3 = QtWidgets.QComboBox(self.groupBox)
self.comboBox_3.setGeometry(QtCore.QRect(60, 30, 101, 24))
self.comboBox_3.setObjectName("comboBox_3")
self.lineEdit_2 = QtWidgets.QLineEdit(self.groupBox)
self.lineEdit_2.setGeometry(QtCore.QRect(60, 60, 101, 24))
self.lineEdit_2.setObjectName("lineEdit_2")
self.label_5 = QtWidgets.QLabel(self.groupBox)
self.label_5.setGeometry(QtCore.QRect(10, 60, 61, 16))
self.label_5.setObjectName("label_5")
self.lineEdit_3 = QtWidgets.QLineEdit(self.groupBox)
self.lineEdit_3.setGeometry(QtCore.QRect(170, 30, 41, 24))
self.lineEdit_3.setObjectName("lineEdit_3")
self.pushButton_2 = QtWidgets.QPushButton(self.centralwidget)
self.pushButton_2.setGeometry(QtCore.QRect(110, 210, 111, 31))
self.pushButton_2.setContextMenuPolicy(QtCore.Qt.DefaultContextMenu)
self.pushButton_2.setAcceptDrops(False)
self.pushButton_2.setAutoFillBackground(False)
self.pushButton_2.setObjectName("pushButton_2")
self.pushButton_3 = QtWidgets.QPushButton(self.centralwidget)
self.pushButton_3.setGeometry(QtCore.QRect(230, 210, 111, 31))
self.pushButton_3.setObjectName("pushButton_3")
MainWindow.setCentralWidget(self.centralwidget)
self.menubar = QtWidgets.QMenuBar(MainWindow)
self.menubar.setGeometry(QtCore.QRect(0, 0, 361, 21))
self.menubar.setObjectName("menubar")
MainWindow.setMenuBar(self.menubar)
self.statusbar = QtWidgets.QStatusBar(MainWindow)
self.statusbar.setObjectName("statusbar")
MainWindow.setStatusBar(self.statusbar)

self.retranslateUi(MainWindow)
QtCore.QMetaObject.connectSlotsByName(MainWindow)

def retranslateUi(self, MainWindow):
    _translate = QtCore.QCoreApplication.translate

```

```

MainWindow.setWindowTitle(_translate("MainWindow", "Wavelet 1-D"))
self.pushButton.setText(_translate("MainWindow", "..."))
self.label.setText(_translate("MainWindow", "Data:"))
self.label_2.setText(_translate("MainWindow", "Subset:"))
self.label_3.setText(_translate("MainWindow", "Display m:"))
self.groupBox.setTitle(_translate("MainWindow", "Wavelet parameters"))
self.label_4.setText(_translate("MainWindow", "Wavelet:"))
self.label_5.setText(_translate("MainWindow", "Level:"))
self.pushButton_2.setText(_translate("MainWindow", "Analyze and Show"))
self.pushButton_3.setText(_translate("MainWindow", "Save coeffs"))
self.pushButton.clicked.connect(lambda: self.loadFile())
self.pushButton_2.clicked.connect(lambda: self.analyze_and_show())

self.subset_list = ["Bytes", "Packets", "Flows", "Avg flow packets", "Avg flow bytes", "Avg packet size", "Flow
behavior"]
self.comboBox.addItem(self.subset_list)
self.comboBox.setCurrentText("Bytes")

display_list = ["Signal - selected subset", "Signals - all subsets", "DWT - Full Decomposition", "DWT - Coefs", "CWT"]
self.comboBox_2.addItem(display_list)
self.comboBox_2.setCurrentText("Signal - selected subset")

wavelet_list = ["haar", "db", "sym", "coif"]
self.comboBox_3.addItem(wavelet_list)
self.comboBox_3.setCurrentText("haar")

def loadFile(self):
    self.file_name = fd.askopenfilename(filetypes=((("CSV files", "*.csv"),
        ("All files", "*.*"))))

    self.lineEdit.setText(self.file_name)
    self.df = pd.read_csv(self.file_name, sep=",")

    self.show_id_attacks = False
    self.show_h_lines = False

def analyze_and_show(self):
    if self.comboBox_2.currentIndex() == 0:
        figure()
        plot(self.df.iloc[:, self.comboBox.currentIndex()], 'r', linewidth=1, label=self.comboBox.currentText())

        grid()
        legend(loc='best')
        show()
    elif self.comboBox_2.currentIndex() == 1:
        figure()
        for l in range(7):
            subplot(7, 1, l+1)
            plot(self.df.iloc[:, l], 'r', linewidth=1, label=self.subset_list[l])
            grid()
            legend(loc='best')
            show()

    elif self.comboBox_2.currentIndex() == 2:
        self.wavelet_transf(self.df.iloc[:, self.comboBox.currentIndex()], self.comboBox_3.currentText(),
int(self.lineEdit_3.text()), int(self.lineEdit_2.text()))
        figure()
        subplot(int(self.lineEdit_2.text())+2, 1, 1)
        plot(self.df.iloc[:, self.comboBox.currentIndex()], 'r', linewidth=1, label=self.comboBox.currentText())
        grid()
        legend(loc='best')

        subplot(int(self.lineEdit_2.text())+2, 1, 2)
        if self.show_h_lines:
            med = statistics.median(self.coeffs_lvl[-1][0])
            avg = statistics.mean(self.coeffs_lvl[-1][0])

```

```

        print("MEDIAN = " + str(med))
        print("MEAN = " + str(avg))
        axhline(y = med, linewidth=1, color = 'red')
        axhline(y = avg, linewidth=1, color = 'orange')
        x = [float(len(self.df.iloc[:, self.comboBox.currentIndex()])/len(self.coefs_lvl[-1][0]))*i for i in
range(len(self.coefs_lvl[-1][0]))]
        step(x,self.coefs_lvl[-1][0], label="cD"+str(int(self.lineEdit_2.text())))
        grid()
        legend(loc='best')

        for l in range(int(self.lineEdit_2.text())):
            subplot(int(self.lineEdit_2.text()+2, 1, int(self.lineEdit_2.text()-l+2)
            x = [float(len(self.df.iloc[:, self.comboBox.currentIndex()])/len(self.coefs_lvl[l][1]))*i for i in
range(len(self.coefs_lvl[l][1]))]
            step(x,self.coefs_lvl[l][1], color="green", label="cA"+str(l+1))
            grid()
            legend(loc='best')
            show()

    elif self.comboBox_2.currentIndex() == 3:

        self.wavelet_transf(self.df.iloc[:, self.comboBox.currentIndex()], self.comboBox_3.currentText(),
int(self.lineEdit_3.text()), int(self.lineEdit_2.text()))

        figure()
        for i, ci in enumerate(self.coefs_d_lvl):
            imshow(ci.reshape(1, -1), extent=[0, len(self.df.iloc[:,0])-1, i + 0.5, i + 1.5], cmap='viridis', aspect='auto',
interpolation='bessel') #inferno nearest

        labelsD = []
        for i in range(len(self.coefs_d_lvl)):
            labelsD.append("cD" + str(i+1))

        ylim(0.5, len(self.coefs_d_lvl) + 0.5)
        yticks(range(1, len(self.coefs_d_lvl) + 1), labelsD)
        legend(loc='best')
        show()

    elif self.comboBox_2.currentIndex() == 4:

        scales = np.arange(1, 101)
        x = self.df.iloc[:, self.comboBox.currentIndex()]
        coef, freqs = pywt.cwt(x, scales, 'mexh')

        figure();
        imshow(abs(coef), extent=[0,200,100,1], interpolation='bilinear', cmap='jet', aspect='auto')
        gca().invert_yaxis()
        yticks(np.arange(1,101,1))
        xticks(np.arange(0,201,10))
        show()

def wavelet_transf(self, x, wavelet, order, lvl):
    strWalevet = wavelet
    if wavelet != "haar":
        strWalevet += str(order)

    self.coefs_lvl=[]
    self.coefs_d_lvl=[]

    for l in range(lvl):
        self.coefs_lvl.append(wavedec(x, strWalevet, level=l+1,mode="periodic"))
        self.coefs_d_lvl.append(self.coefs_lvl[l][1])

if __name__ == "__main__":
    import sys

```

```
app = QtWidgets.QApplication(sys.argv)
MainWindow = QtWidgets.QMainWindow()
ui = Ui_MainWindow()
ui.setupUi(MainWindow)
MainWindow.show()
sys.exit(app.exec_())
```

**ДОДАТОК Б**  
**Слайди презентації**

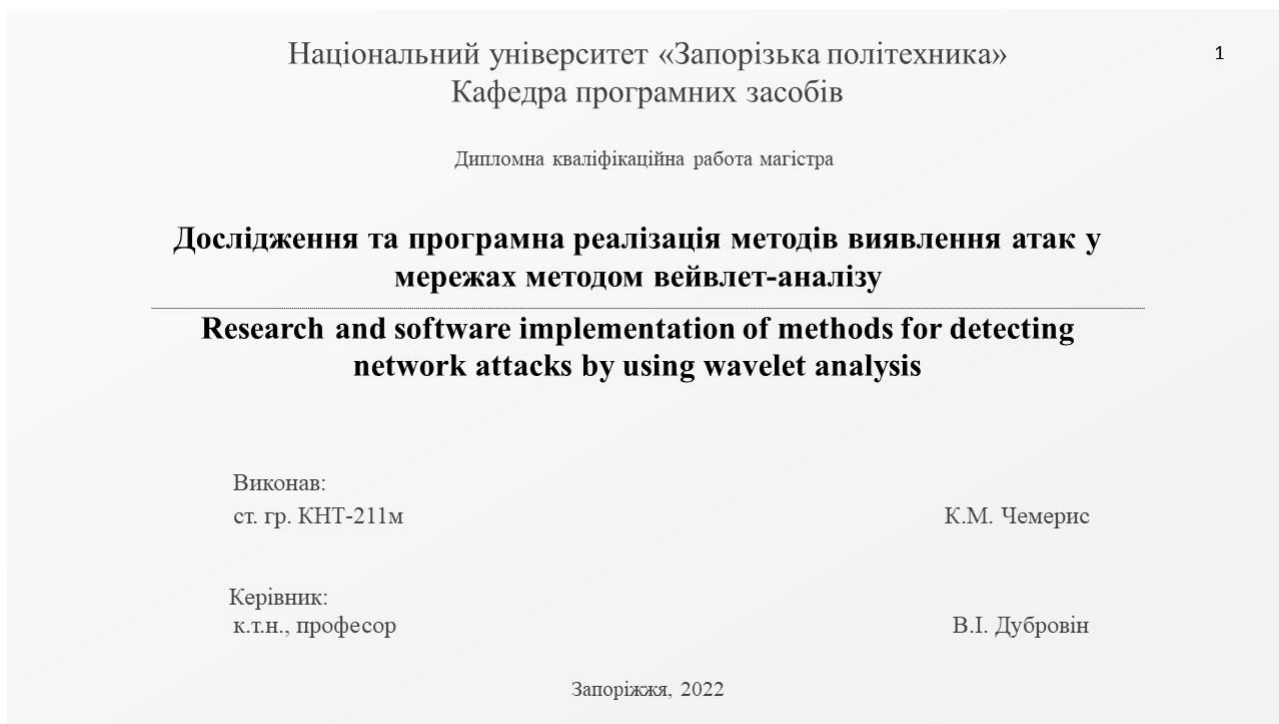


Рисунок Б.1 – Слайд 1

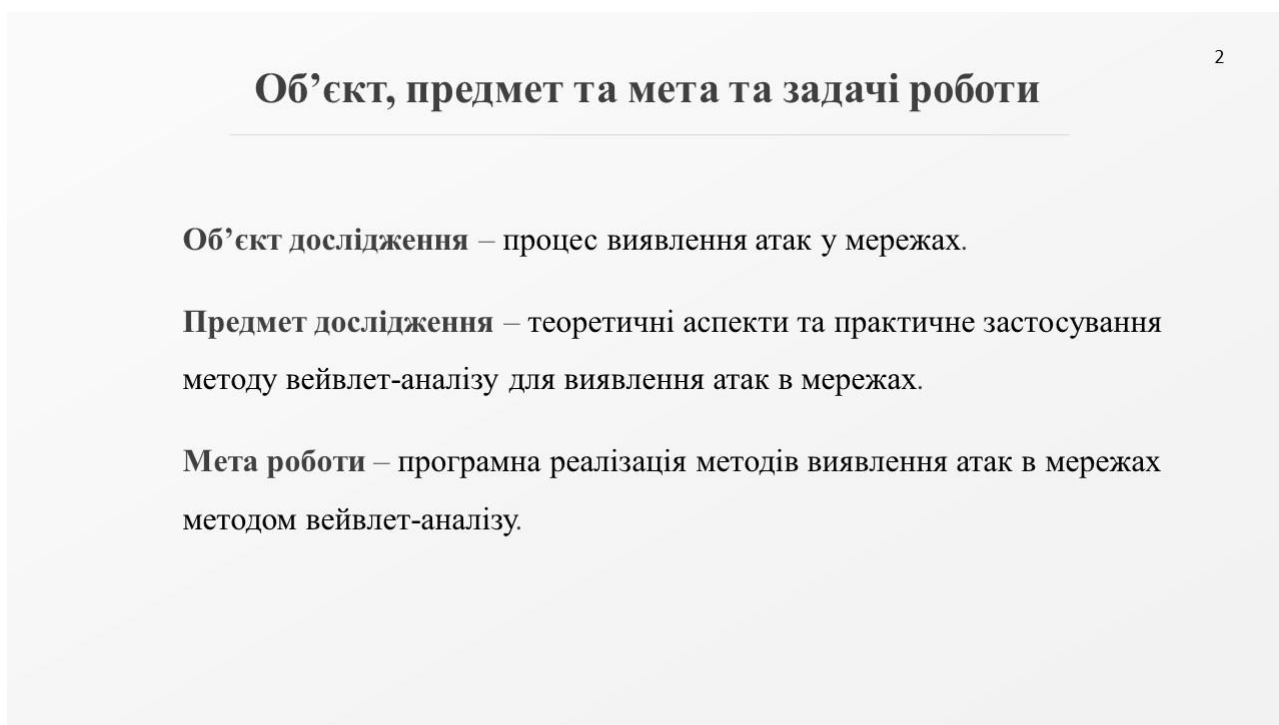


Рисунок Б.2 – Слайд 2

## Мережеві атаки

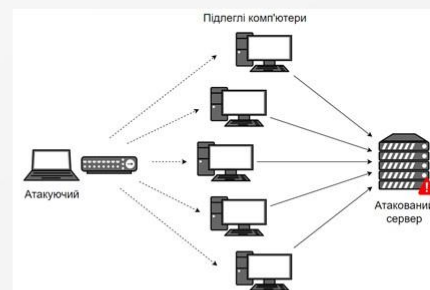
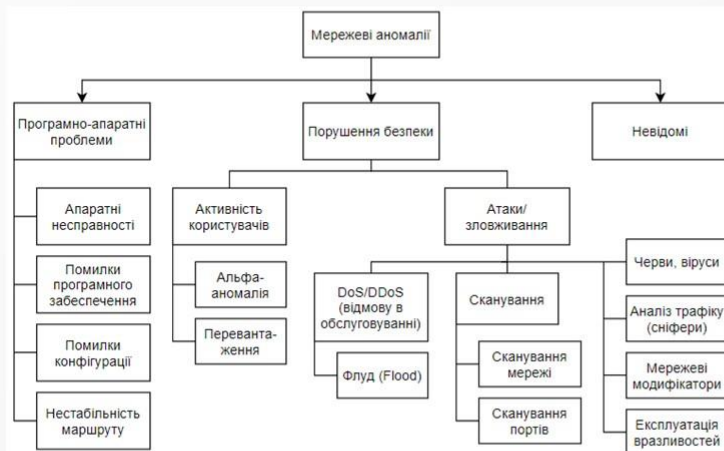


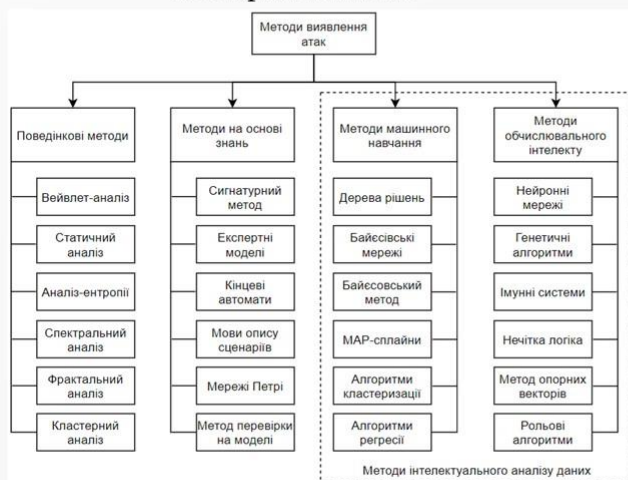
Схема DDoS-атаки

Класифікація аномалій у мережах

Рисунок Б.3 – Слайд 3

## Відомі методи виявлення мережевих атак

### Класифікація методів



### Порівняння методів

| Критерій                          | Поведінкові методи    | Методи на основі знань | Методи машинного навчання | Методи обчислюв. інтелекту |
|-----------------------------------|-----------------------|------------------------|---------------------------|----------------------------|
| Виявлені цілі                     | Зловживання, Аномалії | Зловживання            | Зловживання, Аномалії     | Зловживання, Аномалії      |
| Простота реалізації               | —                     | ±                      | —                         | ±                          |
| Низькі витрати пам'яті            | ±                     | ±                      | +                         | +                          |
| Виявлення відомих атак/аномалій   | +                     | +                      | +                         | +                          |
| Виявлення невідомих атак/аномалій | +                     | —                      | ±                         | ±                          |
| Точність виявлення                | +                     | +                      | ±                         | ±                          |

Обрано метод вейвлет-аналізу

Рисунок Б.4 – Слайд 4

## Вейвлет-перетворення

Вейвлет-перетворення дозволяє виділяти одночасно як частотну, так і тимчасову компоненти мінливості.

Ортогональні вейвлети для ДВП:

- Хаара
- Добеші
- Койфлета
- Сімлета

Неортогональні вейвлети для БВП:

- Гауссова
- Мексиканська капелюха
- Морле

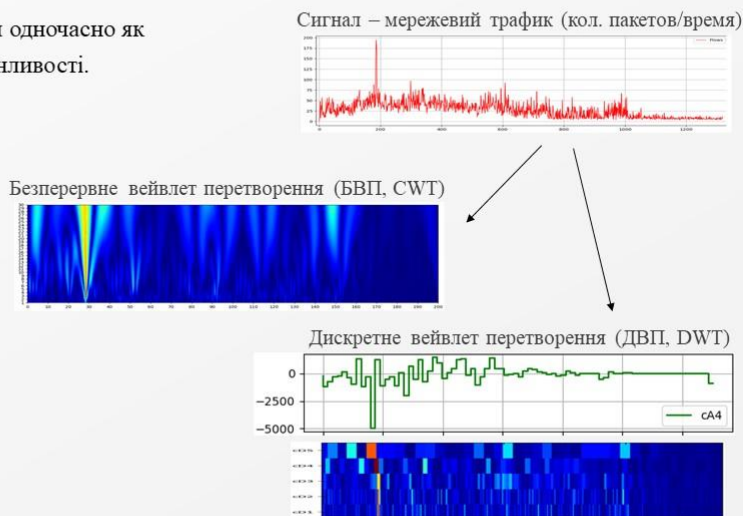


Рисунок Б.5 – Слайд 5

## Дискретне вейвлет перетворення. Алгоритм Малла

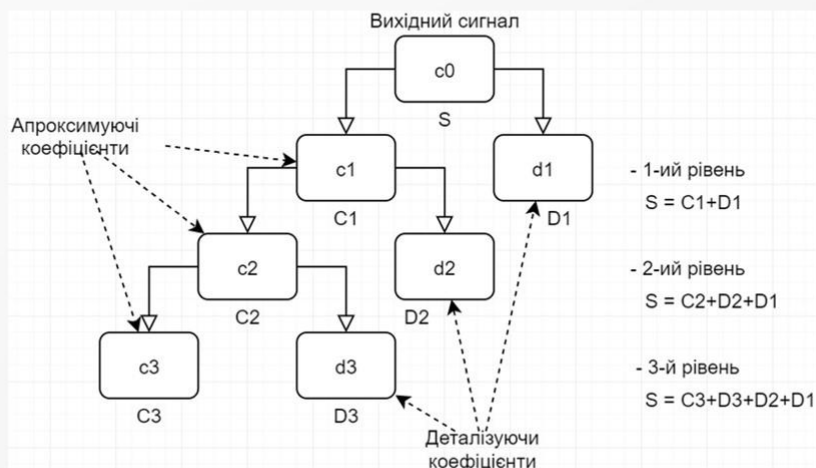


Рисунок Б.6 – Слайд 6

## Основні функції системи

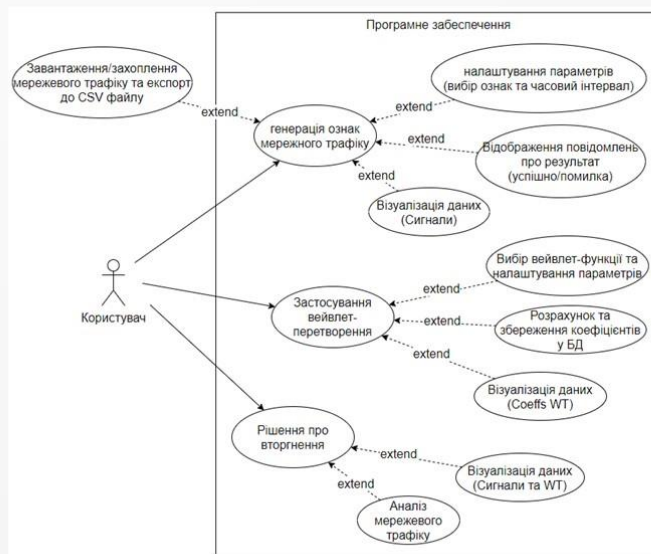


Рисунок Б.7 – Слайд 7

## Вибір програмних засобів

| Критерій                            | Python                 | R       | Matlab |
|-------------------------------------|------------------------|---------|--------|
| Кросплатформність                   | Так                    | Так     | Так    |
| Динамічна типізація                 | Так                    | Так     | Так    |
| Візуалізації даних                  | Так                    | Так     | Так    |
| Продуктивність                      | +                      | +       | ++     |
| Простота та читабельність           | +++                    | ++      | ++     |
| Підтримка аналізу даних             | +++                    | +++     | +++    |
| Документація та навчальні посібники | +++                    | ++      | +++    |
| Бібліотеки вейвлет-аналізу          | Так                    | Так     | Так    |
| IDE                                 | PyCharm, Visual Studio | RStudio | Matlab |

### Додаткові бібліотеки Python:

- PyQt - для створення GUI;
- Csv - для роботи з файлами CSV;
- NumPy - для роботи з масивами;
- Matplotlib - для візуалізації даних;
- PyWavelets - для вейвлет-перетворення.

Обрана мова програмування Python

Рисунок Б.8 – Слайд 8

## Набір даних

Використаний набір даних DARPA, який включає:

Навчальні дані:

- перший тиждень (без атак);
- другий тиждень (містить 23+ атак);
- третій тиждень (без атак).

Тестові дані:

- четвертий тиждень;
- п'ятий тиждень.

На рисунку зображено мережевий трафік, записаний протягом 22 годин у четвер другого тижня з 9 атаками.



Рисунок Б.9 – Слайд 9

## Попередня обробка даних

Попередня обробка даних у цій роботі виконується за двома етапами:

- 1) Експорт даних з tcpdump до csv файлу з властивостями:
  - порядковий номер;
  - час від початку захоплення;
  - розмір пакету (байт);
  - протокол;
  - адреса відправника;
  - порт відправника;
  - адреса одержувача;
  - порт одержувача.
- 2) Генерація ознак для кожного протоколу.

Для кожного протоколу (TCP, UDP, ICMP) згенеровано ознаки

| № | Ознак            | Опис                                                                    |
|---|------------------|-------------------------------------------------------------------------|
| 1 | bytes_i          | Кількість байтів за часовий інтервал (i)                                |
| 2 | packets_i        | Кількість пакетів за часовий інтервал (i)                               |
| 3 | flows_i          | Кількість потоків за часовий інтервал (i)                               |
| 4 | avg_flow_packets | Середня кількість пакетів у потоці за часовий інтервал (i)              |
| 5 | avg_flow_bytes   | Середня кількість байтів у потоці за часовий інтервал (i)               |
| 6 | avg_packet_size  | Середній розмір пакета у потоці за часовий інтервал (i)                 |
| 7 | flow_behavior    | Відношення кількості потоків до байтів на пакет за часовий інтервал (i) |

Рисунок Б.10 – Слайд 10

## Характеристики мережєвих атак

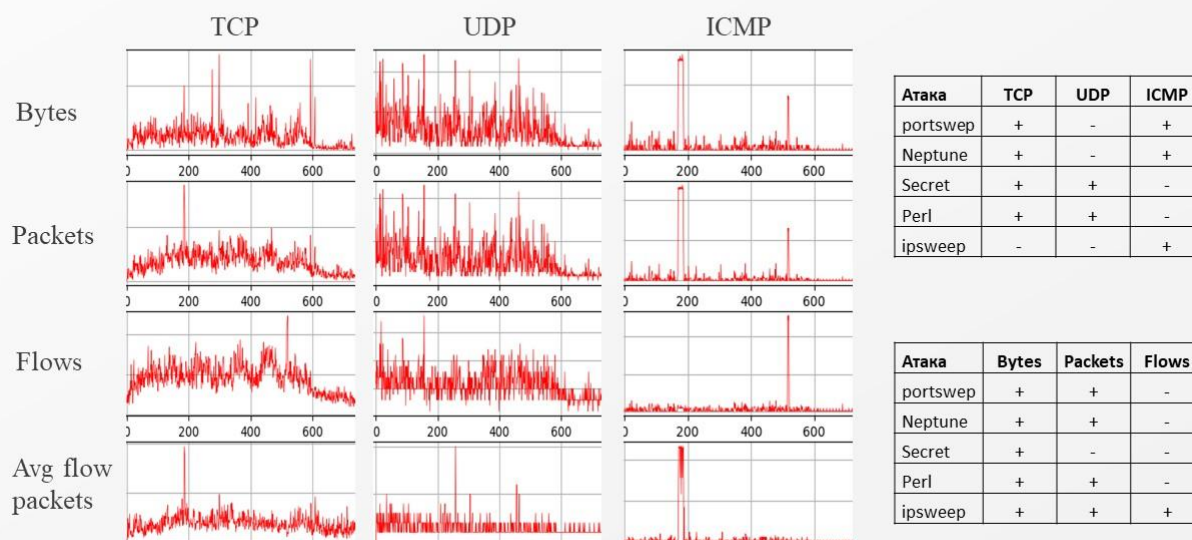


Рисунок Б.11 – Слайд 11

## Результати виявлення мережєвих атак

Оцінки виявлення атаки secret у вікнах (ICMP, avg flow packets)

| Рівень розкладання | Хаара | Добет-2 | Добет-4 | Койфлет-1 | Симплет-4 |
|--------------------|-------|---------|---------|-----------|-----------|
| 1                  | -     | -       | -       | -         | -         |
| 2                  | -     | ++      | +       | ++        | ++        |
| 3                  | ++    | ++      | ++      | ++        | ++        |
| 4                  | ++    | +       | +       | ++        | ++        |
| 5                  | +     | -       | ++      | +         | +         |

Оцінки виявлення атаки secret у вікнах (ICMP, avg flow packets)

| Рівень розкладання | Хаара | Добет-2 | Добет-4 | Койфлет-1 | Симплет-4 |
|--------------------|-------|---------|---------|-----------|-----------|
| 1                  | ++    | ++      | ++      | +         | +         |
| 2                  | ++    | ++      | -       | +         | ++        |
| 3                  | +     | -       | +       | ++        | ++        |
| 4                  | -     | +       | ++      | +         | -         |
| 5                  | +     | -       | -       | -         | -         |

- аномалії не помічені на відповідному рівні;
- + аномалії важко помічені чи не всі аномалії помічені;
- ++ всі аномалії добре помічені на відповідному рівні.

Вейвлет Хаара за різними рівнями розкладання сигналу (TCP, пакетів за хвилину)

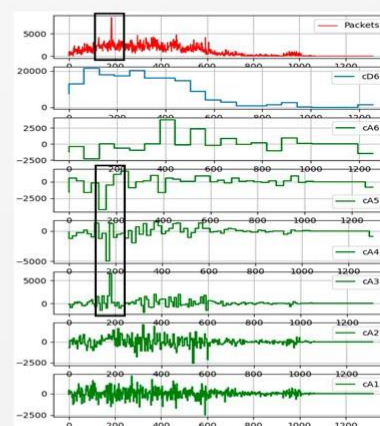


Рисунок Б.12 – Слайд 12

## Проблеми розроблюваної системи та рішення

### Проблеми:

- ручне перетворення мережевого трафіку в потрібний для системи формат;
- недостатність використання однієї вейвлет-функції;
- є вірогідність помилкового виявлення.

### Рішення:

- реалізувати підсистему автоматичного перетворення записаного мережного трафіку на потрібний формат;
- паралельне використання кількох вейвлет-функцій;
- збільшення кількості набору даних;
- застосування вейвлет-аналізу для кількох вікон різного розміру (1, 15, 60 сек.).

Рисунок Б.13 – Слайд 13

## Висновки

- проаналізовано предметну область та порівняно існуючі методи виявлення мережевих атак;
- описано архітектуру системи, обрано програмні засоби та набір даних;
- розроблено інтерфейс та компоненти системи для обробки даних та виявлення мережевих атак із застосуванням вейвлет-аналізу та візуалізацій.
- проведено тестування та отримано результати виявлень мережевих атак.

Для подальшого вдосконалення запропоновано використовувати моделювання нормального мережевого трафіку на основі вейвлет-апроксимації та прогнозування за допомогою ARX моделі та прийняття рішення про наявність або відсутності атак.

Основні положення роботи було представлено в тезах доповіді науково-практичної конференції Тиждень науки–2021 та у журналі «Наука і техніка Повітряних Сил Збройних Сил України – 2022».

Рисунок Б.14 – Слайд 14