Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет комп'ютерної інженерії та управління
(повна назва)

Кафедра електронних обчислювальних машин
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА

Пояснювальна записка

Рівень вищої освіти другий (магістерський)

Метод моніторингу трафіка в корпоративній мережі

(тема)

Виконав:
здобувач 2 року навчання,
групи СПм-23-5
Владислав СМІРНОВ
(власне ім'я, прізвище)

Спеціальність
123 «Комп'ютерна інженерія»
(код і повна назва спеціальності)

Тип програми освітньо-наукова
(освітньо-професійна або освітньо-наукова)

Освітня програма
Системне програмування
(повна назва освітньої програми)

Керівник: доц. Володимир ФЕДОРЧЕНКО
(посада, власне ім'я, прізвище)

Допускається до захисту

Завідувач кафедри ЕОМ _____ Андрій КОВАЛЕНКО
(підпис) (власне ім'я, прізвище)

2025 р.

Харківський національний університет радіоелектроніки

Факультет комп’ютерної інженерії та управління

Кафедра електронних обчислювальних машин

Рівень вищої освіти другий (магістерський)

Спеціальність 123 «Комп’ютерна інженерія»
(код і повна назва)

Тип програми освітньо-наукова
(освітньо-професійна або освітньо-наукова)

Освітня програма Системне програмування
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
“____” _____ 20__ р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувачеві Смірнову Владиславу Руслановичу
(прізвище, ім’я, по батькові)

1. Тема роботи Метод моніторингу трафіка в корпоративній мережі

затверджена наказом по університету від “ 21 ” квітня 2025 р. № 296 Ст

2. Термін подання здобувачем роботи до екзаменаційної комісії 16 червня 2025 р.

3. Вхідні дані до роботи
моніторинг трафіку
діагностика проблем в роботі мережі
виявлення мережевих атак
діаграма класів для опису логічних з’єднань

4. Перелік питань, що потрібно опрацювати у роботі
Аналіз предметної оласті та постановка завдання
Моделі та алгоритми моніторингу корпоративної мережі
Програмні засоби аналізу трафіка в мережі

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій 17 слайдів

6. Консультанти розділів роботи (заповнюється за наявності консультантів згідно з наказом, зазначеним у п.1)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Строк / терміни виконання етапів роботи	Примітка
1	Отримання завдання та аналіз літератури	21.04.2025–30.04.2025	
2	Огляд існуючих рішень та алгоритмів	01.05.2025–12.05.2025	
3	Розробка методу	13.05.2025–22.05.2025	
4	Вибір програмних засобів	23.05.2025–30.05.2025	
5	Програмна реалізація	31.05.2025–02.06.2025	
6	Аналіз отриманих результатів	03.06.2025–05.06.2025	
7	Оформлення записки	06.06.2025–15.06.2025	

Дата видачі завдання " 21 " квітня 2025 р.

Здобувач _____
(підпис)

Керівник роботи _____ доц. Володимир ФЕДОРЧЕНКО
(підпис) (посада, власне ім'я, прізвище)

РЕФЕРАТ

Пояснювальна записка кваліфікаційної роботи: 75 с., 19 рис., 2 дод., 8 джерел.

МОНІТОРИНГ МЕРЕЖЕВОГО ТРАФІКУ, КОРПОРАТИВНІ МЕРЕЖІ, ЛОГІЧНІ ПОТОКИ, АНАЛІЗ ПРОТОКОЛІВ, ЕНТРОПІЯ ПАКЕТІВ, ВИЯВЛЕННЯ АНОМАЛІЙ, ВІЗУАЛІЗАЦІЯ ТРАФІКУ, ОБРОБКА PCAP, МЕРЕЖНА БЕЗПЕКА, GOOGLE COLAB.

Метою кваліфікаційної роботи є розробка, реалізація та апробація методу моніторингу мережного трафіку в корпоративному середовищі, який забезпечує виділення логічних потоків, аналіз протоколів, оцінку ентропії корисного навантаження, виявлення аномалій та побудову візуалізацій для підтримки мережної безпеки.

У ході виконання кваліфікаційної роботи були досліджені методи моніторингу трафіку у корпоративній мережі, також було реалізовано повноцінний проєкт у середовищі Google Colab, що включає генерацію тестового трафіку, обробку .pcap-файлів, агрегацію даних у структурованому вигляді, реалізацію методу логічного поділу на потоки та аналізу поведінки протоколів. Проведений аналіз дозволив виявити характерний розподіл довжин мережевих пакетів, який вказує на переважання коротких службових запитів. Аналіз ентропії продемонстрував наявність як відкритих, так і зашифрованих потоків, що дозволяє робити висновки про рівень конфіденційності переданих даних. Візуалізація активності IP-адрес виявила ключові вузли мережевої взаємодії, а аналіз активності протоколів у часі дозволив виділити періоди потенційної аномальної поведінки.

ABSTRACT

Master’s thesis: 75 pages, 19 figures, 2 appendices, 8 sources.

NETWORK TRAFFIC MONITORING, CORPORATE NETWORKS, LOGICAL FLOW EXTRACTION, PROTOCOL ANALYSIS, PACKET ENTROPY, ANOMALY DETECTION, TRAFFIC VISUALIZATION, PCAP PROCESSING, NETWORK SECURITY, GOOGLE COLAB.

The major goal of this thesis is the development, implementation, and validation of a network traffic monitoring method tailored for corporate environments. The proposed method enables the identification of logical flows, protocol analysis, entropy evaluation of payloads, anomaly detection, and the generation of visualizations to support network security management.

In order to various methods for monitoring traffic in corporate networks were studied. A complete implementation was carried out using the Google Colab environment, encompassing the generation of test traffic, processing of .pcap files, structured data aggregation, and the execution of a method for logical flow separation and protocol behavior analysis. The conducted analysis revealed a characteristic distribution of network packet lengths, indicating a dominance of short service requests. Entropy analysis confirmed the presence of both open and encrypted flows, allowing inferences about the confidentiality level of transmitted data. Visualization of IP activity helped identify key nodes in the network, while protocol activity over time exposed potential intervals of anomalous behavior.

The developed method allows for the integration of the extracted insights into network monitoring systems, serves as a foundation for automatic anomaly detection, and demonstrates the efficiency of combining traditional networking tools with modern approaches to data analysis and visualization.

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

AI – штучний інтелект

API – інтерфейс прикладного програмування

CSV – формат табличних даних, розділених комами

DDoS – розподілена атака на відмову в обслуговуванні

DNS – система доменних імен

DoS – атака на відмову в обслуговуванні

GUI – графічний інтерфейс користувача

HTTP – протокол передавання гіпертексту

ICMP – протокол керування повідомленнями в мережі Інтернет

IDS – система виявлення вторгнень

IP – протокол Інтернету

IPS – система запобігання вторгненням

LSTM – довготривала короткочасна пам’ять

ML – машинне навчання

PCAP – формат файлу для зберігання перехоплених мережевих пакетів

SIEM – система управління інформацією та подіями безпеки

SSL – протокол захищених сокетів

TCP – протокол керування передачею

TLS – протокол транспортного рівня безпеки

UDP – протокол дейтаграм користувача

VPN – віртуальна приватна мережа

ВСТУП

У сучасному інформаційному суспільстві корпоративні мережі стали критично важливою складовою функціонування організацій, забезпечуючи безперервний потік даних між користувачами, сервісами, офісами та центрами обробки інформації. У зв’язку з цим питання надійності, безпеки та стабільності таких мереж виходить на перший план, зокрема в умовах зростаючих кіберзагроз, ускладнення внутрішньої інфраструктури та появи нових способів обходу засобів захисту. Одним із найефективніших способів контролю над станом мережі, виявлення порушень та реагування на потенційні атаки є метод моніторингу трафіку.

Моніторинг мережевого трафіку – це процес збору, аналізу та інтерпретації даних, що передаються через мережу, з метою виявлення відхилень від норми, аномалій, атак або несправностей. Цей метод забезпечує комплексне бачення внутрішніх і зовнішніх процесів, що відбуваються в корпоративному середовищі, і дозволяє вчасно ідентифікувати проблеми, ще до того як вони переростають у критичні інциденти.

Метод моніторингу трафіку передбачає застосування спеціалізованих інструментів – аналізаторів, які можуть бути апаратними або програмними. Вони виконують фільтрацію, дешифрування, обробку та логування пакетів, і часто інтегруються з системами виявлення вторгнень (IDS), запобігання атак (IPS) або централізованими платформами безпеки (SIEM). На основі отриманих даних формуються звіти, здійснюється оцінка ефективності мережі, виявляються вузькі місця, а також аналізується поведінка користувачів.

Окреме місце у структурі моніторингу займають методи аналізу вкладених тунелів, що використовуються для обходу традиційних фаєрволів або прихованого витоку даних. Сучасні рішення також мають справлятися з

модифікованими або зашифрованими пакетами, в яких складно ідентифікувати справжню сутність переданої інформації. Тому розробка і впровадження нових алгоритмів, що дозволяють ефективно виявляти аномалії, є актуальним напрямом дослідження.

У цій роботі розглянуто метод моніторингу трафіку в корпоративній мережі з акцентом на побудову моделі аналізу, розробку логічних з'єднань, реалізацію програмного забезпечення та створення інструментів аналізу даних. Аналіз охоплює повний цикл – від захоплення мережевих пакетів до генерації звітів з висновками щодо безпеки або ефективності мережної взаємодії.

Метою кваліфікаційної роботи є розробка, реалізація та апробація методу моніторингу мережного трафіку в корпоративному середовищі, який забезпечує виділення логічних потоків, аналіз протоколів, оцінку ентропії корисного навантаження, виявлення аномалій та побудову візуалізацій для підтримки мережної безпеки.

Доцільно виокремити такі завдання дослідження, які забезпечують її досягнення:

Проаналізувати сучасні підходи до моніторингу мережевого трафіку в корпоративних мережах, зокрема методи аналізу пакетів, виявлення аномалій і логічного групування даних:

- розробити концепцію методу моніторингу, що включає формальні моделі представлення пакетів, логічних потоків, протоколів і їх взаємозв'язків;
- сформувати підхід до побудови логічних потоків за допомогою механізму п'ятиелементного ключа (5-tuple: IP-адреси, порти, протокол), з можливістю реконструкції пов'язаних пакетів у логічні зв'язки;
- запропонувати метод аналізу стеку протоколів, що дозволяє виділяти вкладені рівні та визначати контексти взаємодії у трафіку;
- реалізувати процедуру оцінювання ентропії корисного навантаження для виявлення ймовірного шифрування або ненормативних передач даних;

- інтегрувати механізми виявлення аномалій, базовані на статистичних ознаках потоків: розмір, частота, ентропія, розподілення активності тощо;

- створити прототип програмного засобу в середовищі Google Colab, що включає генерацію тестового трафіку, обробку .pcap-файлів, аналіз логічних потоків та візуалізацію результатів;

- забезпечити візуальне представлення логічних з’єднань у вигляді графа, а також побудову графіків розподілу активності IP-адрес, протоколів, обсягів трафіку тощо;

- провести експериментальну апробацію розробленого методу на згенерованих даних, включаючи моделювання аномальних сценаріїв, з метою перевірки ефективності розробленого підходу.

Об’єктом дослідження у кваліфікаційній роботі є процеси передавання, обробки та контролю мережевого трафіку в корпоративному середовищі, зокрема ті аспекти, що стосуються логічної структури трафіку, функціонування мережевих протоколів, формування потоків даних, а також виявлення відхилень та потенційних аномалій у рамках забезпечення кібербезпеки корпоративної мережевої інфраструктури.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Мережні пакети

Аналіз мережевого трафіку є ключовим компонентом інформаційної безпеки та управління IT-інфраструктурою, оскільки дозволяє здійснювати повноцінний контроль за передачею даних у мережі, виявляти аномалії, підозрілу активність, а також оптимізувати використання ресурсів. У контексті корпоративного середовища, де велика кількість пристроїв, користувачів і сервісів щодня генерує значні обсяги даних, ефективний аналіз трафіку є не лише засобом захисту, а й інструментом для прийняття управлінських рішень.

Процес аналізу трафіку включає кілька етапів:

- захоплення даних: фіксація вхідних і вихідних мережевих пакетів за допомогою спеціалізованих програм або обладнання. Це може бути здійснено як на кінцевих пристроях, так і на рівні маршрутизаторів, комутаторів або дзеркальних портів;
- попередня обробка: декодування заголовків пакетів, виділення основних метаданих (IP-адреси, порти, протоколи, час передачі тощо), групування сесій;
- глибокий аналіз (DPI): інспекція вмісту пакетів, аналіз структур вкладених тунелів, пошук ознак шкідливої активності або порушень політик безпеки;
- класифікація трафіку: розподіл за типами (веб, пошта, потокове відео, VoIP, P2P тощо), що дозволяє зрозуміти реальне навантаження на мережу і можливі причини перевантаження;
- візуалізація та звітність: побудова графіків, таблиць, інфографіки, що ілюструють стан мережі в реальному часі або ретроспективно.

Особливе значення має виявлення аномального трафіку, який може бути ознакою:

- мережевих сканувань і спроб підбору паролів;
- ботнет-активності;
- несанкціонованого використання ресурсів;
- витоку конфіденційної інформації;
- атак типу DDoS або MITM.

Сучасні інструменти аналізу трафіку, як-от Wireshark, Zeek, ntopng або власні програмні рішення, дають можливість не тільки виявляти загрози, а й інтегрувати їх у загальну систему кіберзахисту. Вони забезпечують виявлення шаблонів поведінки, машинне навчання для класифікації подій та автоматичне створення інцидентів.

Важливою тенденцією є також перехід до реального часу, коли система безперервно фіксує та обробляє потік даних, дозволяючи миттєво реагувати на загрози. У такому середовищі критично важливим є не лише ефективність алгоритмів, а й їхня здатність до масштабування та адаптації.

1.2 Мережні рівні для програмних засобів аналізу мережевого трафіку

У програмних засобах аналізу мережевого трафіку важливо розуміти, на якому рівні мережевої взаємодії працює той чи інший інструмент. Ці рівні визначаються моделлю OSI (рисунок 1.1), яка поділяє функціонування мережі на сім логічних шарів. Програмні рішення здебільшого охоплюють кілька з них, залежно від призначення та глибини аналізу.

На найнижчому фізичному рівні інструменти орієнтовані на взаємодію з фізичним середовищем передавання даних, проте в більшості випадків аналіз трафіку тут виконується апаратними засобами. Канальний рівень дозволяє програмно досліджувати структуру кадрів Ethernet або Wi-Fi, а також відслідковувати MAC-адреси, що корисно при виявленні проблем з підключенням або конфліктів пристроїв у локальній мережі. Найбільш

активний аналіз відбувається на мережевому та транспортному рівнях, де програмні засоби ідентифікують IP-адреси, маршрутизацію, сесії TCP/UDP, а також фіксують спроби сканування портів або збої в з'єднаннях.

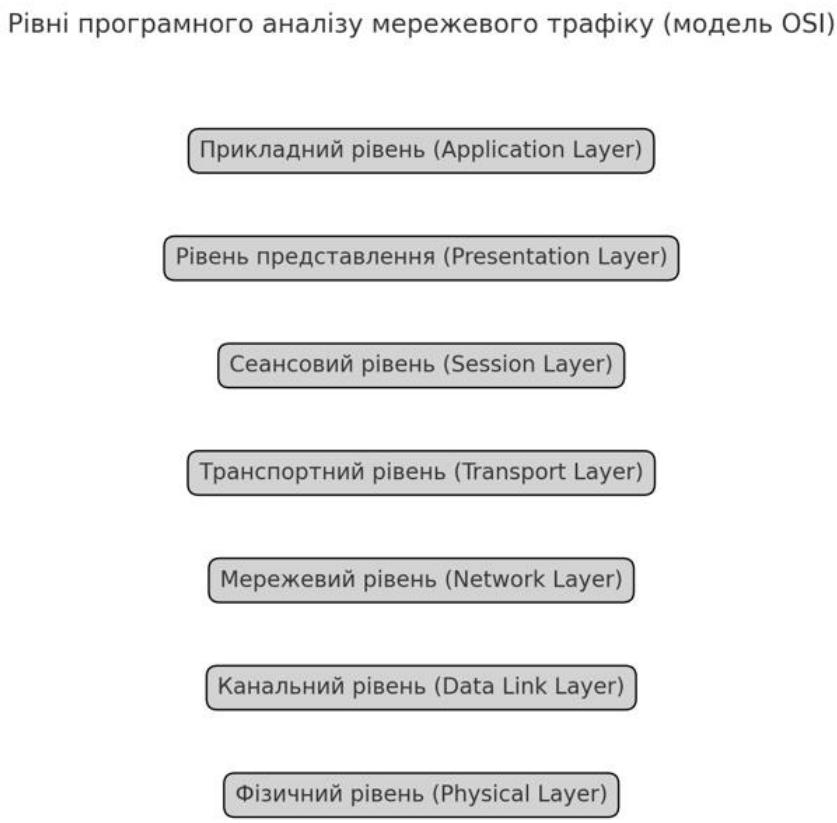

Рисунок 1.1 – Рівні програмного аналізу мережного трафіку

Сеансовий рівень, хоча і менш популярний для окремих інструментів, все ж охоплюється складними системами аналізу, зокрема в контексті тривалих VPN-з'єднань або тунелів. На рівні представлення програмне забезпечення може здійснювати розшифрування даних, декодування форматів, розпізнавання зашифрованого трафіку, що особливо важливо для безпеки в умовах використання HTTPS. Найвищим у цій ієрархії є прикладний рівень, де аналізатори вивчають конкретні сервіси й протоколи, такі як HTTP, DNS, FTP, а також команди, запити й відповіді, що дозволяє виявляти спроби обходу безпеки, витоки даних або аномальну поведінку користувачів.

На рисунку 1.2 представлена узагальнена схема рівнів аналізу мережевого трафіку – від фізичної діагностики до прикладного аналізу, яка показує логічну послідовність обробки даних. Це допомагає краще зрозуміти, які функції виконує кожен рівень в системі моніторингу.

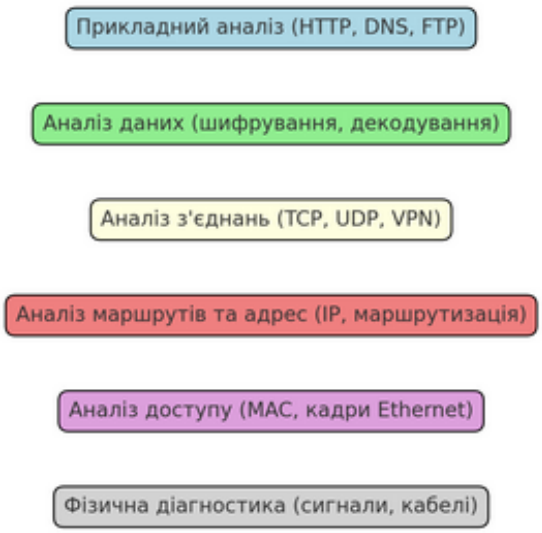

Рисунок 1.2 – Узагальнена модель рівнів аналізу мережевого трафіку

Таким чином, ефективне програмне забезпечення для аналізу трафіку має охоплювати якомога більше рівнів OSI-моделі, забезпечуючи як загальний огляд структури мережевої взаємодії, так і глибокий інспекційний аналіз вмісту пакетів, сесій та дій користувача. Це дозволяє не лише відслідковувати поточний стан мережі, а й прогнозувати потенційні загрози, адаптивно реагувати на події й будувати стратегію кіберзахисту відповідно до актуальних ризиків.

Поверхневий аналізатор трафіку – це інструмент початкового рівня, який виконує базовий збір і фільтрацію мережевих даних без глибокого розбору структури пакетів або вмісту трафіку. Його основна функція – надати швидке уявлення про загальний стан мережі та основні параметри трафіку.

До функцій, які включає поверхневий аналізатор, належать:

- захоплення основних метаданих: IP-адреси джерела й призначення, порти, об’єм переданих даних, час з’єднання;
- підрахунок трафіку: загальний обсяг переданих пакетів, швидкість трафіку, кількість активних з'єднань;
- визначення типів протоколів: наприклад, HTTP, DNS, SMTP, без детального розбору запитів;
- групування по сесіях або пристроях: на основі IP-адрес, MAC-адрес або портів;
- виявлення базових аномалій: різке зростання трафіку, незвично велика кількість запитів, загальна недоступність вузлів;
- побудова простих звітів і графіків: з метою візуалізації трафіку за періодами часу.

Такий аналізатор зазвичай не розшифровує зашифрований трафік, не виконує інспекцію на рівні вмісту, і не має інструментів для виявлення складних кіберзагроз. Проте він цілком придатний для початкового моніторингу, загального аудиту мережевої активності та виявлення основних відхилень.

1.3 Передача трафіку

У комп’ютерних мережах передача трафіку реалізується шляхом фрагментації даних на малі структуровані одиниці – пакети, які пересилаються між вузлами мережі згідно з певними протоколами. Кожен пакет містить як корисне навантаження – безпосередньо дані користувача або програми, так і заголовки, які забезпечують доставку цього пакета, описуючи його маршрут, тип, довжину, пріоритет тощо.

Передача трафіку відбувається за принципом інкапсуляції, коли кожен рівень моделі OSI додає свій заголовок. Наприклад, застосунок надсилає дані, які обгортаються транспортним заголовком TCP/UDP, потім – IP-

заголовком мережевого рівня, далі – заголовком Ethernet-кадру на канальному рівні. Таким чином формується повноцінний мережевий пакет.

Заголовки є критично важливою частиною пакету, оскільки в них зазначаються:

- IP-адреса джерела й призначення;
- порт з'єднання (TCP/UDP);
- протокол;
- ідентифікатори пакета;
- контрольні суми (для перевірки цілісності);
- інформація про фрагментацію;
- TTL (time-to-live), що визначає життєвий цикл пакета.

На транспортному рівні TCP-заголовки містять додаткові поля – такі як номер послідовності, номер підтвердження, прапорці SYN/ACK, що керують сесією. У разі використання UDP заголовок простіший і включає лише довжину, порти й контрольну суму.

Передача трафіку – це процес маршрутизації цих пакетів через мережу. Роутери читають IP-заголовки, приймають рішення про подальшу передачу і, якщо необхідно, змінюють частину заголовку, наприклад, TTL. Кінцевий пристрій приймає пакет, видаляє заголовки, декодує дані і передає їх застосунку.

Окрім коректного складання пакету, важливо також розуміти роль фрагментації: великі блоки даних розбиваються на менші, що відповідають обмеженням мережі (наприклад, MTU – максимальна довжина пакета). На кінцевій точці ці фрагменти повторно збираються в одне повідомлення.

Таким чином, структура трафіку – це послідовність пакетів із багаторівневими заголовками, які дозволяють забезпечити гнучкість, контроль, маршрутизацію та безпечну доставку інформації через глобальні або локальні мережі.

Для повного розуміння процесу передачі даних у комп’ютерних мережах надзвичайно важливо знати, як працюють заголовки пакетів

відповідно до моделі OSI. Ця модель розділяє мережеву взаємодію на сім логічних рівнів, і на кожному з них до переданих даних додається своя службова інформація – заголовок. Цей процес називається інкапсуляцією. Заголовки служать для ідентифікації, маршрутизації, перевірки цілісності, контролю сеансів, формату даних та інших задач, необхідних для безпечного та ефективного передавання трафіку.

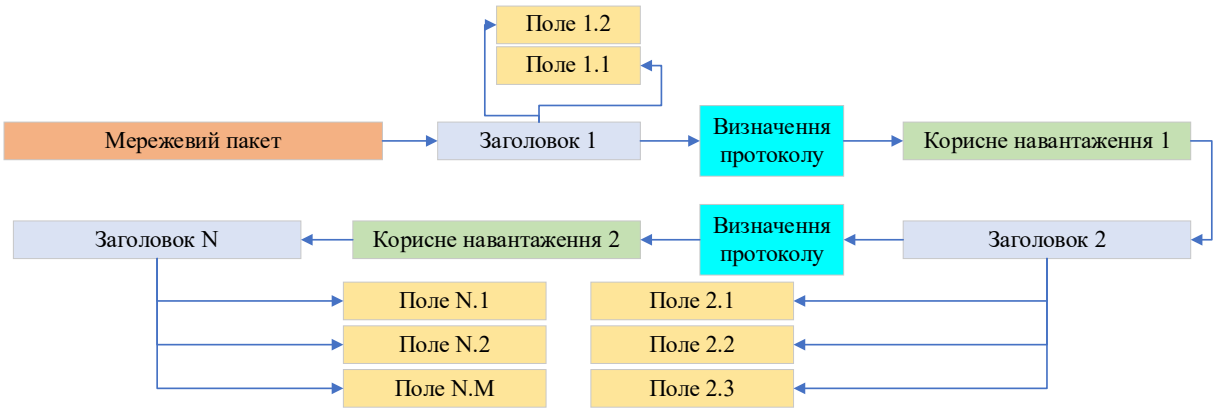

Рисунок 1.3 – Виділення і розбір заголовків протоколів в пакеті

На прикладному рівні створюються дані, які генерує користувач або програмне забезпечення, наприклад, браузер при надсиланні HTTP-запиту. Далі ці дані можуть бути закодовані, стиснуті або зашифровані на рівні представлення. Якщо використовується захищений протокол, наприклад HTTPS, то на сеансовому рівні створюється спеціальний заголовок для керування криптографічною сесією, що забезпечує аутентифікацію й конфіденційність.

Наступний етап – транспортний рівень, де додається один із найважливіших заголовків: TCP або UDP. Він містить інформацію про номери портів джерела та призначення, номери послідовності й підтвердження, контрольну суму, а також керуючі прапорці для встановлення або завершення з’єднання. Далі, на мережевому рівні, формується заголовок IP-пакету, який визначає IP-адреси відправника й одержувача, час життя

пакета (TTL), ідентифікатори фрагментів та протокол транспортного рівня, що використовується.

На канальному рівні до всього додається заголовок Ethernet або іншого канального протоколу. Він включає MAC-адреси відправника і призначення, поле типу протоколу та контрольну суму кадру для виявлення помилок. Усе це передається по фізичному середовищу у вигляді сигналів, але на цьому рівні нових заголовків вже не додається – лише здійснюється фізична передача сформованого кадру.

На приймальному боці пакет проходить зворотний процес: кожен рівень OSI видаляє свій заголовок, обробляє його інформацію і передає залишок на вищий рівень. Лише після повного видалення всіх заголовків дані досягають програми, яка їх ініціювала. Таким чином, заголовки в OSI-моделі виконують функцію своєрідного конверта для кожного "рівня" змісту – від прикладного повідомлення до фізичного сигналу.

Рисунок 1.3 ілюструє процес структурування мережевого пакета за принципом інкапсуляції даних на різних рівнях мережевої моделі, що відповідає логіці моделі OSI. Він показує, як дані, що передаються через мережу, формуються у вигляді послідовних вкладених заголовків і корисних навантажень, де кожен рівень додає свою службову інформацію.

На початку бачимо загальний об’єкт – мережевий пакет. Він містить заголовок 1, який у свою чергу складається з окремих полів, наприклад, поле 1.1 та поле 1.2. Цей заголовок містить службову інформацію, необхідну для маршрутизації або керування передачею. Далі йде визначення протоколу, яке дозволяє інтерпретувати, що саме передається у наступному блоці.

Після цього до пакета додається корисне навантаження 1, яке містить у собі наступний заголовок – заголовок 2. У ньому вже інша структура полів, таких як поле 2.1, 2.2, 2.3, і цей заголовок відповідає наступному рівню мережевої обробки (наприклад, транспортному або сеансовому). Знову виконується визначення протоколу, після чого йде корисне навантаження 2,

яке у свою чергу також містить вкладений заголовок N з власними полями (поле N1, N2 тощо).

Таким чином, рисунок демонструє, що мережевий пакет будується з багатьох шарів, кожен з яких має власну структуру заголовка і навантаження, а передача здійснюється через механізм вкладення одного рівня в інший. На кожному етапі визначення протоколу дозволяє інтерпретувати наступний блок, забезпечуючи правильну маршрутизацію, доставку і розуміння даних.

Ця схема чітко пояснює, як функціонує інкапсуляція в мережах: кожен новий рівень додає власний заголовок, і таким чином формується повний мережевий пакет, який у зворотному напрямку при прийомі проходить процес деінкапсуляції – видалення заголовків і передача корисного навантаження на вищий рівень.

1.4 Переваги використання програмних засобів моніторингу трафіка

Використання програмних засобів моніторингу трафіку у корпоративних мережах дає змогу отримати глибоке розуміння внутрішніх і зовнішніх процесів, що відбуваються у цифровому середовищі організації. Однією з головних переваг (рисунок 1.4) є забезпечення прозорості трафіку: адміністратори можуть у реальному часі бачити, який трафік проходить мережею, які пристрої та додатки його генерують, які порти та протоколи задіяні. Це дозволяє вчасно виявляти аномалії, наприклад сплески активності або несанкціоновані підключення.

Наступною важливою перевагою є здатність до оперативного виявлення загроз. Програмні аналізатори трафіку можуть визначати підозрілу поведінку, зокрема спроби сканування портів, DDoS-атаки, ботнет-активність, витік конфіденційних даних або порушення політик доступу. Завдяки підтримці інтелектуальних алгоритмів, багато з таких засобів вміють автоматично класифікувати трафік і формувати сигнали про потенційні загрози ще до того, як вони спричинять шкоду.

Також важливим є покращення продуктивності та оптимізація мережевих ресурсів. Моніторинг дозволяє виявити, які служби або користувачі споживають найбільше пропускної здатності, де виникають вузькі місця, затримки або дублювання трафіку. На основі цих даних можна провести модернізацію інфраструктури, оптимізувати маршрути або налаштувати пріоритети.

Крім цього, програмні засоби моніторингу є необхідним елементом аудиту та відповідності стандартам. Вони зберігають логування подій, що дозволяє відстежити всі підключення, з'єднання, зміни конфігурації чи спроби вторгнення. Це критично важливо для дотримання вимог таких стандартів, як ISO/IEC 27001, GDPR або PCI DSS.

Рисунок 1.4 – Переваги використання

І ще однією важливою перевагою є масштабованість і гнучкість. Програмні рішення можуть бути інтегровані в мережі будь-якого розміру – від малого офісу до великих центрів обробки даних. Більшість сучасних інструментів підтримують інтеграцію з іншими системами безпеки, такими як SIEM, IDS/IPS або фаєрволи, що дозволяє створити централізовану екосистему реагування на інциденти.

Загалом, програмні засоби моніторингу трафіку дозволяють організаціям не лише виявляти проблеми, а й проактивно керувати мережею,

попереджувати загрози, контролювати навантаження та забезпечувати безпеку з урахуванням специфіки конкретного середовища.

1.5 Сегментація в TCP

Сегментація в TCP (рисунок 1.5) – це процес поділу великого обсягу даних на менші частини, які зручно передавати мережею у вигляді окремих TCP-сегментів. Цей механізм є ключовим компонентом протоколу TCP і забезпечує надійність, контрольованість та впорядкованість передачі інформації в комп'ютерних мережах.

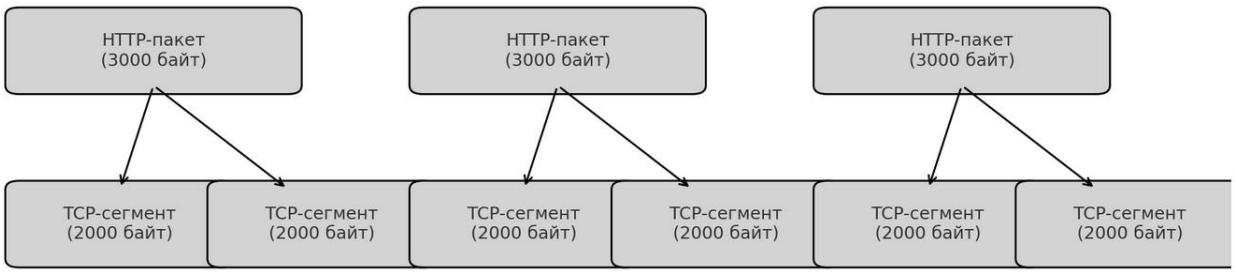

Рисунок 1.5 – Приклад сегментації TCP

Рисунок 1.5 ілюструє процес сегментації HTTP-пакетів на TCP-сегменти. Кожен HTTP-пакет розміром 3000 байт розбивається на два TCP-сегменти – по 2000 байт, де другий сегмент частково містить залишок даних. Це демонструє механізм поділу великих повідомлень на менші частини для надійної передачі мережею.

Коли прикладна програма, наприклад веб-браузер або поштовий клієнт, надсилає дані, TCP отримує великий об'єм інформації, який фізично неможливо передати єдиним пакетом через обмеження мережі, зокрема максимальний розмір MTU. Тому TCP автоматично розбиває ці дані на послідовність сегментів – логічних одиниць передачі, кожна з яких містить частину корисного навантаження та власний заголовок TCP.

Цей заголовок включає поля, які є критично важливими для правильної реконструкції даних на приймальному боці. Зокрема, TCP використовує номери послідовності, що дозволяє приймачу зібрати фрагменти у правильному порядку навіть у разі, якщо деякі з них надійдуть із затримкою або у зміненій послідовності. Крім того, TCP передбачає механізм підтвердження отримання, за допомогою якого приймач повідомляє відправника про успішний прийом певного сегмента. У випадку втрати або пошкодження TCP автоматично ініціює повторну передачу лише втраченого сегмента, а не всього потоку даних.

Сегментація також дозволяє реалізувати контроль потоку та керування перевантаженням. Відправник може адаптувати розмір вікна передачі до швидкості приймача, уникаючи перевантаження буфера. Паралельно, при виявленні ознак перевантаження мережі (наприклад, втрати сегментів), TCP змінює частоту передачі або зменшує об’єм сегментів, щоб стабілізувати з’єднання.

Важливо зазначити, що сегментація виконується на транспортному рівні, тобто вже після того, як дані сформовані прикладним рівнем, але ще до того, як вони інкапсулюються у мережеві пакети (IP). Саме TCP забезпечує логіку «розбити – пронумерувати – перевірити – зібрати назад», яка гарантує, що отримувач отримає повне повідомлення без втрат і спотворень.

1.6 Мережний трафік

Сучасні аналізатори мережевого трафіку зазвичай реалізуються на основі модульної архітектури, що пояснюється динамічним розвитком мережевих технологій і постійною появою нових протоколів, які потребують інтеграції. Централізована реалізація, де всі процедури синтаксичного аналізу сконцентровані в одному блоці, ускладнює масштабування системи та робить її вразливою до помилок при оновленні. Натомість модульний підхід передбачає окрему реалізацію кожного протоколу у вигляді

самостійного модуля, що описує відповідні алгоритми обробки та структури даних. Однак така архітектура ставить додаткові вимоги до взаємодії між модулями – під час інтеграції нового модуля потрібно забезпечити його узгодження з уже існуючими, не порушуючи їх функціонування.

Зміни в коді вже реалізованих модулів можуть викликати небажані наслідки – від порушення їх логіки до виникнення помилок, які складно виявити та виправити. До того ж такі зміни потребують повторної компіляції модулів. Тому пріоритетним завданням під час розширення функціоналу аналізатора є збереження стабільності системи шляхом мінімізації модифікацій уже існуючих компонентів.

У практиці мережевого моніторингу важливу роль відіграє аналіз збережених мережевих трасувань (файлів із записаним трафіком). Вони використовуються для відтворення помилок, налаштування парсерів, а також у процесі розслідування інцидентів, пов’язаних з порушенням інформаційної безпеки. Отже, надзвичайно важливо забезпечити переносимість модулів розбору між автономними (офлайн) та інтерактивними (онлайн) середовищами аналізу.

Оскільки мережеві дані передаються у вигляді послідовностей пакетів, для ефективної роботи аналізаторів необхідно враховувати низку технічних та архітектурних вимог:

- здатність відновлювати повноцінні потоки даних навіть у разі втрати або переупорядкування пакетів;
- підтримка обробки вкладених тунельних структур;
- аналіз трафіку з елементами шифрування чи модифікації;
- автоматичне розпізнавання протоколу прикладного рівня;
- можливість локалізувати й відтворювати помилки парсингу;
- підтримка додавання нових протоколів без потреби змінювати вже наявні модулі;
- забезпечення переносимості модулів парсингу між різними режимами роботи системи.

Перші три вимоги в основному стосуються представлення та обробки самих пакетів, тоді як решта – визначають архітектурні характеристики системи та вимоги до її гнучкості. Додатково слід відзначити доцільність наявності зручного графічного інтерфейсу, який підвищує ефективність взаємодії з користувачем.

1.7 Аналогічне програмне забезпечення

Перевірка дотримання вимоги №1 здійснюється шляхом використання першого тестового набору мережевих трасувань (рисунок 1.6), який дозволяє оцінити здатність аналізатора відновлювати потоки даних за умов втрати або переупорядкування пакетів. Аналіз вкладених тунелів зі складною або нестандартною конфігурацією стеку (вимога №2) тісно пов'язаний із функціональністю автоматичного розпізнавання протоколів (вимога №4). У зв'язку з цим перевірка відповідності зазначеним вимогам передбачає дослідження внутрішніх механізмів протокол-детекції в аналізованих інструментах, а також аналіз способу представлення даних у структурі внутрішнього моделювання мережевих пакетів (на основі аналізу вихідного коду).

Мережна траса	Особливості
Trace11.pcap	перегрупування TCP-сегментів; TCP-з'єднання, встановлені до початку запису траси
Trace12.pcap	перегрупування TCP-сегментів;повторна передача TCP-сегментів
Trace13.pcap	повторна передача TCP-сегментів
Trace14.pcap	повторна передача TCP-сегментів
Мережна траса	**Стек протоколів**
Trace21.pcap	ETH-IPv4-IPv4-ICMP
Trace22.pcap	ETH-IPv4-GRE-IPv4-ICMP
Trace23.pcap	ETH-IPv4-UDP-Teredo-IPv6-ICMPv6
Trace24.pcap	ETH-VLAN-IPv6-IPv4-GRE-PPP-IPv4-UDP-DNS
Мережна траса	**Особливості**
Trace31.pcap	Зашифроване SSL-з'єднання
Trace32.pcap	Передача стиснутих даних за допомогою HTTP

Рисунок 1.6 – Набори мережних трас

Підтримка популярних тунельних протоколів додатково верифікується за допомогою другого тестового набору мережевих трасувань. У свою чергу, функціональність аналізу шифрованих або змінених даних (вимога №3) перевіряється шляхом застосування третього тестового набору, орієнтованого на SSL/HTTPS-трафік і стиснені HTTP-сесії.

Вимоги №5–7 стосуються архітектурних характеристик системи, тому їх дотримання визначається шляхом ретельного аналізу вихідного коду програмного забезпечення та вивчення відповідної документації. Особливу увагу при цьому приділяється питанням розширюваності системи, модульності підсистеми розбору та переносу результатів між різними режимами використання.

Виявлення вторгнень передбачає постійний моніторинг та аналіз подій у комп'ютерних системах і мережах з метою ідентифікації інцидентів, що суперечать визначеним політикам безпеки або встановленим стандартам. Для автоматизації цього процесу використовуються системи виявлення вторгнень (СВВ), які можуть бути реалізовані як у вигляді програмного забезпечення, так і у вигляді апаратних рішень.

Системи виявлення класифікуються за методом детекції на сигнатурні та аномальні. Сигнатурні СВВ здійснюють аналіз вхідного трафіку на основі наявності заздалегідь визначених шаблонів (патернів), що відповідають відомим видам атак. Вони ефективні у виявленні типових загроз, однак безсилі проти нових або модифікованих атак, які ще не мають сигнатури. Аномальні системи, навпаки, орієнтовані на виявлення нетипової або потенційно шкідливої поведінки шляхом побудови моделей «нормального» функціонування на основі методів машинного навчання. Такі системи здатні виявляти раніше невідомі загрози, проте їх недоліком є висока частота хибнопозитивних спрацювань.

Проєкт Snort розпочався у 1998 році та є прикладом сигнатурної системи виявлення вторгнень, орієнтованої на запобігання мережевим атакам. Архітектура Snort передбачає наявність препроцесорів для

попереднього аналізу мережевих пакетів, а також модуля виявлення, який працює за правилами формату Action-Connection [Options].

Кожне правило може бути або статичним (завжди активним), або динамічним (активується за умов, визначених іншими правилами). Компонент Action задає дію (наприклад, журналювання або блокування пакета), Connection описує умови TCP/UDP-з'єднання, до яких застосовується правило, а в Options можна зазначити сигнатуру, ділянку пакету для пошуку, лог-повідомлення та інші параметри.

Наприклад, правило, представлене на рис. 6, спрацьовує у разі виявлення послідовності байтів 00 01 86 a5 у корисному навантаженні TCP-пакета, що надходить на порт 111 машини з мережі 192.168.1.0/24. У відповідь система генерує попередження з повідомленням "mountd access".

Система Bro (нині відома як Zeek) була започаткована в 1995 році. Вона являє собою багатофункціональний інструмент для аналізу мережевого трафіку з можливістю виявлення атак, збору статистики, оцінювання пропускної здатності та діагностики помилок у роботі мережі. Bro – це приклад гібридної СВВ, яка підтримує як сигнатурний аналіз, так і методи поведінкового виявлення на основі ML.

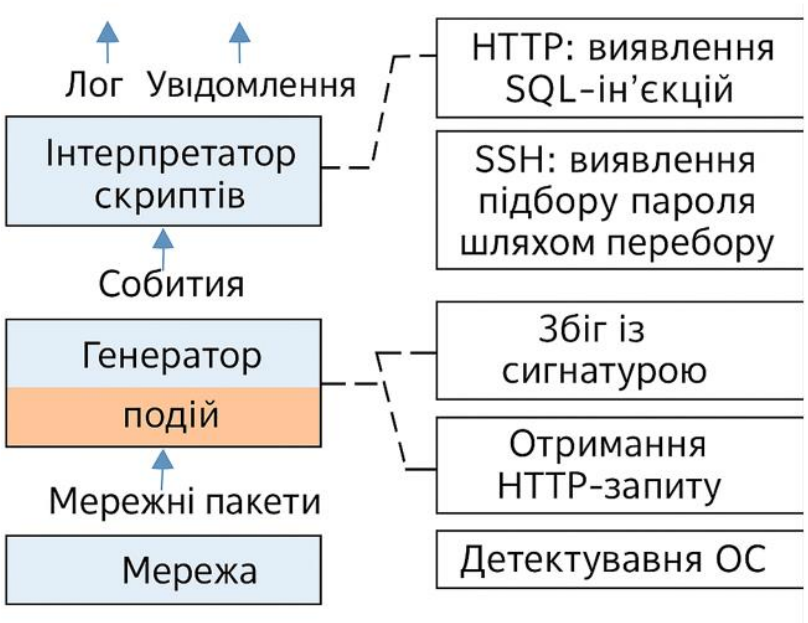

Рисунок 1.7 – Аналізатор Bro

Архітектурно система складається з двох ключових компонентів (див. рисунок 1.7):

- ядра (генератора подій), що виконує аналіз мережевих пакетів і генерує події на основі їх змісту;

- інтерпретатора політик безпеки, який реалізує скрипти для визначення реакції системи на ті чи інші події.

Події в системі можуть виникати в результаті обробки одного чи кількох мережевих пакетів, або ж формуватись як наслідок обробки попередніх подій (підтримується ланцюгове реагування). Така гнучкість дає змогу формалізувати складні сценарії реагування на інциденти без втручання в основний код системи.

Wireshark – один із найпоширеніших інструментів аналізу мережевого трафіку з відкритим кодом, розробка якого була розпочата у 1998 році. На відміну від класичних консольних утиліт типу tcpdump, Wireshark поєднує широкий спектр функціональних можливостей із зручним графічним інтерфейсом, що дозволяє здійснювати як базовий, так і поглиблений аналіз мережевих даних у реальному часі або з використанням збережених файлів.

Характеристика	Snort	Bro/Zeek	Wireshark
Рік початку розробки	1998	1995	1998
Тип СВВ / інструмента	Сигнатурна СВВ	Гібридна СВВ	Аналізатор трафіку
Метод виявлення / аналізу	Зіставлення з шаблонами	Сигнатури + поведінка + ML	Розбір заголовків і пакетів
Підтримка сигнатур	Так	Так	Ні
Підтримка аномального аналізу	Ні	Так	Ні
Гнучкість налаштувань політик	Обмежена	Висока	Непотрібна
Обробка в реальному часі	Так	Так	Ні
Післязахоплювальний аналіз	Ні	Так	Так
Графічний інтерфейс	Ні	Ні	Так
Кількість підтримуваних протоколів	Обмежено	Широка підтримка	~2000
Можливості фільтрації	Фільтрація на основі правил	Фільтрація подій та політик	~206000 полів для фільтрації
Основне призначення	Виявлення відомих атак	Мережевий моніторинг, інциденти	Детальний аналіз збережених даних

Рисунок 1.8 – Порівняльна таблиця Snort, Zeek та Wireshark

Ключова особливість Wireshark полягає в тому, що аналіз відбувається після збереження трафіку у файл, а не в момент його проходження, як це реалізовано у Snort чи Bro/Zeek. Завдяки цьому користувач отримує

можливість ретельно дослідити кожен захоплений пакет, включаючи детальне відображення вмісту заголовків і полів усіх рівнів протоколів.

Інструмент підтримує надзвичайно гнучку систему фільтрації, яка може застосовуватися як у процесі захоплення, так і при подальшому перегляді трафіку. Станом на сьогодні, Wireshark здатен розпізнавати понад 2000 мережевих протоколів, а кількість доступних для фільтрації полів сягає понад 206 000.

Цей інструмент посідає провідне місце серед доступних безкоштовних систем аналізу трафіку та активно використовується у сферах мережевої діагностики, навчання, кібербезпеки та розробки протоколів. Його багатофункціональність, розширюваність і потужні засоби візуалізації роблять Wireshark універсальним засобом для інженерів, дослідників та фахівців з інформаційної безпеки.

2 МОДЕЛЬ ТА МЕТОДИ МОНІТОРИНГУ КОРПОРАТИВНОЇ МЕРЕЖІ

Цей розділ присвячений мережевій взаємодії між програмними компонентами без прив'язки до конкретного стеку протоколів, таких як TCP/IP або IPX/SPX. Передбачається, що обмін даними здійснюється з використанням абстрактного, потенційно багаторівневого стеку протоколів, який умовно складається з N рівнів (рисунок 2.1). У цьому контексті мережа інтерпретується як впорядкована сукупність логічних каналів зв'язку між об'єктами взаємодії. Кожен канал представляє собою логічне з'єднання, в межах якого передається впорядкована послідовність пакетів.

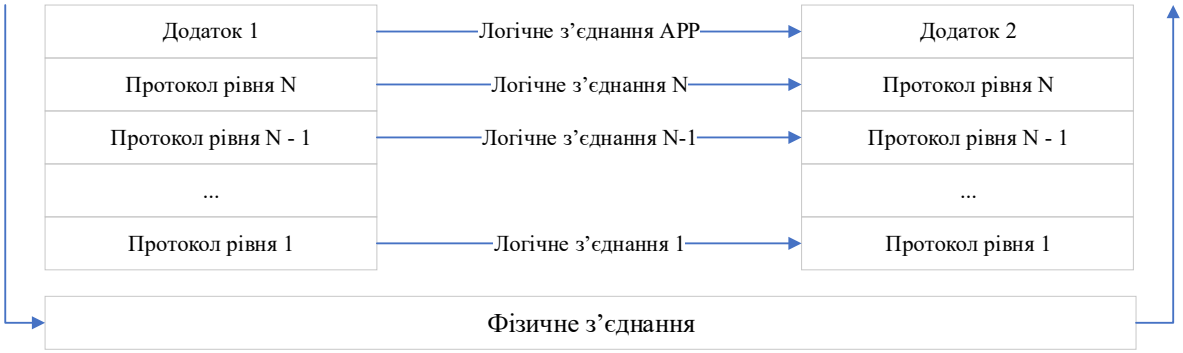

Рисунок 2.1 – Мережна взаємодія між двама застосунками

Розглянемо, що логічне з'єднання на кожному рівні ієрархії мережевого стеку (позначимо рівень як i, де i = 1, ..., N) реалізується за допомогою відповідного протоколу, структура якого визначається набором допустимих форматів пакетів. Під форматом пакета розуміють специфікацію його внутрішньої будови: розподіл байтової послідовності на іменовані поля, а також правила, що регламентують допустимі значення окремих полів.

Загалом, у кожному пакеті переважно виділяється окреме поле для розміщення корисного навантаження, тобто даних протоколу вищого рівня. Решта полів виконує службові функції: вони можуть містити ідентифікатор

типу повідомлення, адресу джерела або призначення, інформацію про довжину полів, індекси, контрольні суми тощо.

Таким чином, кожен пакет відповідного протоколу Protocol і можна формально подати як кортеж, що складається з двох основних частин:

- набору службових полів, які забезпечують керування передачею;
- поля даних, що містить корисне навантаження – вміст, який передається на наступний рівень протоколу:

$$Packet^i = <Control^i, Payload^i> , i = 1 \dots N. \quad (2.1)$$

Усі мережеві протоколи за своїми характеристиками щодо обробки даних можна класифікувати на дві взаємовиключні категорії: ті, що допускають фрагментацію переданих даних, та ті, що її не підтримують. При цьому механізми фрагментації можуть суттєво відрізнятись залежно від специфіки конкретного протоколу. У процесі спостереження за мережею фіксується послідовність низькорівневих пакетів, які передаються через логічні з'єднання. Ці пакети можуть стосуватися різних прикладних програм, які або працюють у парі, або є учасниками багатоточкової передачі даних.

Під час аналізу мережевого трафіку немає необхідності повністю відтворювати логіку функціонування кожного протоколу. Достатньо виокремити байтовий потік у формалізовані повідомлення (пакети) відповідно до визначеного формату, а далі – інтерпретувати службову інформацію (Control) для об'єднання цих повідомлень у логічні групи та вилучення їхнього корисного навантаження.

Поглиблений трафік-аналіз передбачає не лише ідентифікацію окремих пакетів, але й реконструкцію повної послідовності взаємодій на всіх рівнях мережевої ієрархії, що охоплює відновлення логічних з'єднань і зв’язних потоків даних. Через можливу зміну порядку доставки пакетів відносно їх початкового відправлення, особливо у випадку застосування фрагментації, виникає потреба у побудові формальних предикатів, які дозволяють

ідентифікувати попередні й наступні фрагменти повідомлень. Такі предикати визначаються на основі службових полів контрольної інформації протоколу, що забезпечує процес дефрагментації.

Перед переходом до аналізу пакетів вищого рівня необхідно забезпечити впорядкування відповідних пакетів логічного з’єднання згідно зі значеннями встановлених предикатів, що дозволяє коректно реконструювати вихідний потік даних та забезпечити цілісність інформації.

2.1 Розробка моделі

На рисунку 2.2 представлено схему класів, що моделює процес синтаксичного розбору мережевого пакета. Для коректного відновлення послідовності пакетів, що належать до логічного з'єднання, необхідно здійснити вилучення корисного навантаження з пакетів нижчих рівнів логічної ієрархії. У запропонованій моделі така процедура супроводжується копіюванням даних лише у випадках, коли потрібно виконати дефрагментацію. Застосування цієї оптимізованої стратегії дозволяє суттєво зменшити накладні витрати у порівнянні з традиційними підходами, в яких копіювання здійснюється на кожному кроці пошуку або відновлення.

Усі структурні компоненти, отримані під час аналізу (як-от мережеві пакети або окремі поля в їх межах), представляються в системі у вигляді блоків – об’єктів класу Block. Кожен блок визначається як масив байтів фіксованої довжини. При необхідності обробки певної частини блоку створюється новий блок-нащадок, який є підлеглим елементом щодо свого батьківського блоку.

У процесі обробки кожному блоку надається тип синтаксичного аналізу (ParseType, атрибут класу Block), що вказує на відповідну функцію розбору (Parser, атрибут класу Type), яка визначає логіку інтерпретації вмісту блоку згідно з визначеним форматом.

Синтаксичний аналіз полягає в логічному розділенні буфера блоку на іменовані поля, згідно з описом структури відповідного формату пакета F.

Для кожного блоку задається базовий тип структури (StructType, атрибут класу Block), який визначає семантику та типову організацію вмісту:

- Struct – блок містить неоднорідні поля, аналогічно до структури у мові C;

- ProtoStruct – різновид Struct, призначений спеціально для опису заголовків мережевих протоколів;

- FieldSeq – послідовність однорідних полів, подібна до масиву;

- PacketSeq – колекція пакетів мережевого протоколу, яка може включати об'єкти різних форматів, але належних до одного рівня.

У межах загальної структури введено також два спеціалізовані типи блоків:

- блок-потік – використовується для представлення відновлених потоків даних;

- блок-фрагмент – призначений для зберігання як повних мережевих пакетів, так і окремих полів, виділених у процесі аналізу потокових даних.

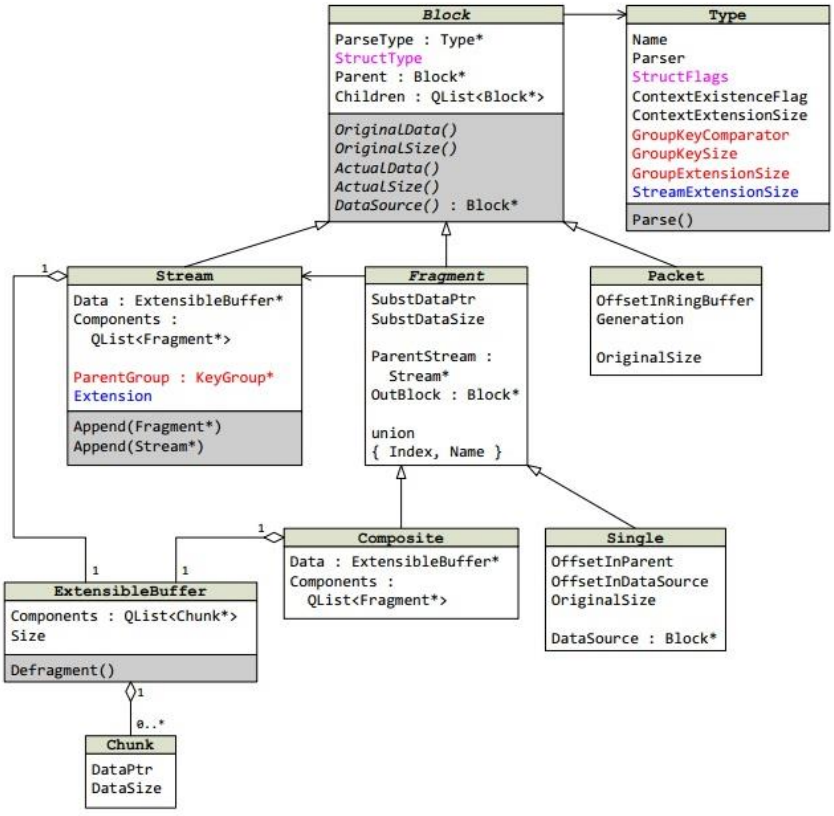

Рисунок 2.2 – Схема класів для опису розібраних мережевих пакетів

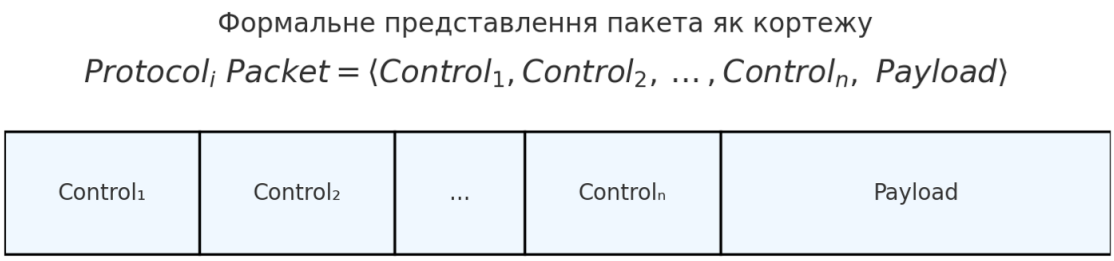

Рисунок 2.3 – Представлення пакету як кортежу

Рисунок 2.2 зображує UML-діаграму класів, яка моделює архітектуру системи керування потоками даних у структурі, що передбачає групування за ключами та контекстами. Основним елементом у цій моделі є клас Stream, що представляє окремий потік даних, пов’язаний із групою KeyGroup, яка виконує функцію класифікації потоків за певним ключем. Кожна група зберігає список відповідних потоків, має посилання на батьківський контекст (Context) і може містити вкладені контексти, які зберігаються у вигляді хеш-таблиці.

Контекст у цьому випадку виступає як логічне середовище, у межах якого функціонують ключові групи; він знає про свій тип (ContextType), визначений через клас Type, а також має інформацію про актуальну активну групу. У свою чергу, тип (Type) задає параметри структури, необхідні для ініціалізації та деініціалізації контекстів і груп, а також визначає розміри розширень. Співвідношення між класами формують ієрархічну та взаємопов’язану систему, в якій Stream агрегується KeyGroup, останній інтегрується в Context, а контекст спирається на типізацію, задану класом Type. Уся модель забезпечує динамічне формування логіки обробки потоків, їх групування, а також дозволяє гнучко управляти контекстом виконання.

Кожне логічне з’єднання формується за участю конкретного мережевого протоколу та подається у вигляді впорядкованої послідовності відповідних пакетів. Водночас, на одному рівні стеку протоколів можуть одночасно існувати пакети, що належать до різних протоколів, отже, у межах одного логічного рівня може спостерігатися множинність незалежних зв’язків. Для того щоб виділити з цього потоку окремі логічні з’єднання,

необхідно згрупувати пакети за належністю до протоколів, які ці зв’язки реалізують. У запропонованій моделі така сукупність пакетів, що відповідає одному протоколу на фіксованому рівні стеку, описується за допомогою об’єкта класу Context.

Кожен контекст асоціюється з мережевим протоколом, на основі якого він був сформований. Тип цього протоколу визначає семантику обробки відповідного контексту й фіксується у полі ContextType класу Context. Окрім цього, контекст може містити спеціальну розширювальну структуру – поле Extension, яка використовується для зберігання додаткових даних, необхідних у процесі синтаксичного аналізу відповідних пакетів. Прикладом такої інформації може бути внутрішній стан декомпресора, що використовується при обробці стисненого трафіку в протоколах типу PPP.

Тип контексту, як структурна характеристика, є важливим під час формування унікальних ідентифікаторів логічних з’єднань. Логічне з’єднання, що реалізується певним протоколом, має бути відображене у вигляді послідовності відновлених (дефрагментованих або декодованих) пакетів. Для здійснення такого відновлення потрібно ідентифікувати відповідну підпослідовність із загального потоку пакетів, яка утворює єдину логічну сутність.

Процедура групування пакетів у такі підпослідовності базується на аналізі службових полів, насамперед – адресної інформації. Кожна така підпослідовність репрезентується об’єктом класу KeyGroup, який формує групу за ключем. Ці групи є логічно вкладеними у відповідний контекст, який виконує роль контейнера, що забезпечує доступ до кожної групи за унікальним ідентифікатором (Key). Зміст, структура та розмір ключа визначаються відповідно до типу контексту. Крім того, кожна ключова група містить власне поле Extension, яке може зберігати технічну інформацію, необхідну для обробки пакетів цієї групи. У цьому контексті ключова група виконує функцію моделі логічного з’єднання на певному рівні мережевого стеку, тобто фактично уособлює логічне з’єднання як одиницю трафіку.

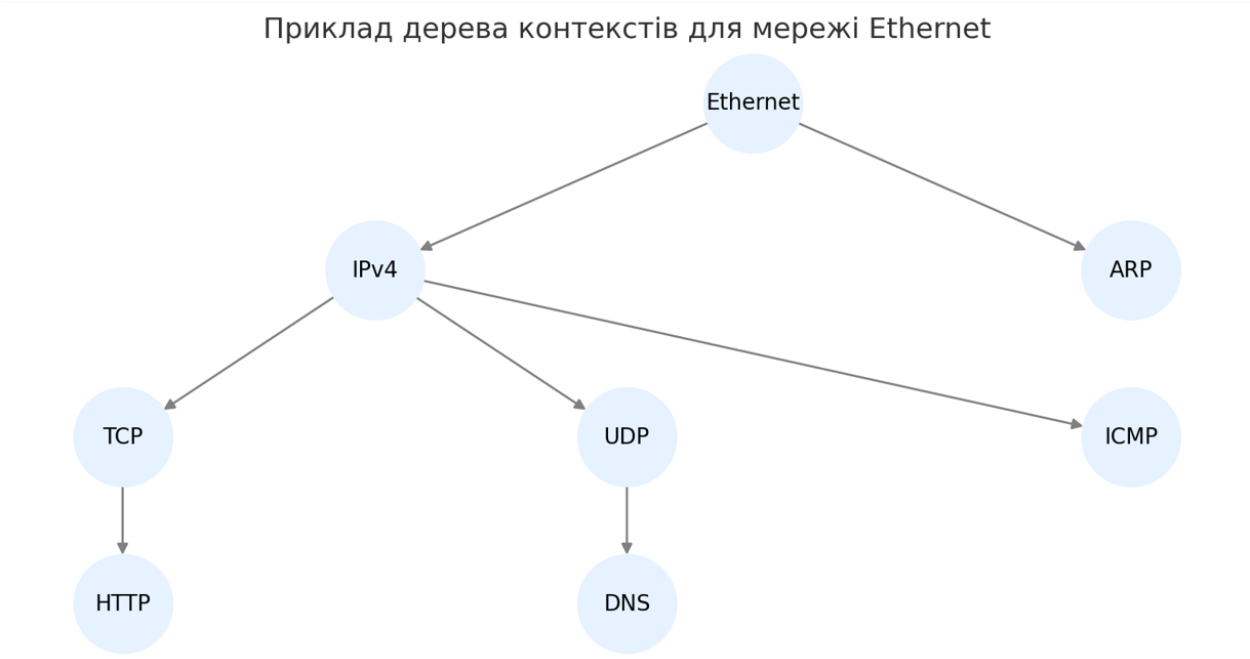

Рисунок 2.4 – Дерево контекстів

На зображенні представлено приклад дерева контекстів для мережі Ethernet, яке ілюструє ієрархічну структуру протоколів у межах мережевого стеку, починаючи з канального рівня та далі через мережевий, транспортний і прикладний рівні.

У центрі дерева знаходиться Ethernet, що виступає базовим протоколом канального рівня. Від нього виходять три гілки до протоколів вищого рівня: IPv4, ARP та ICMP, які функціонують безпосередньо поверх Ethernet. Зокрема, Address Resolution Protocol використовується для визначення фізичних адрес за IP, а Internet Control Message Protocol призначений для передачі службових повідомлень.

Контекст IPv4 є ключовим вузлом, оскільки підтримує інкапсуляцію транспортних протоколів – від нього відгалужуються TCP Transmission Control Protocol і User Datagram Protocol. Обидва вони, у свою чергу, слугують основою для прикладних протоколів: TCP несе HyperText Transfer Protocol, який використовується для веб-комунікації, тоді як UDP є основою для Domain Name System, що відповідає за розв’язування імен у мережі.

Таким чином, ця схема ілюструє логіку побудови контекстів для мережевих з’єднань, де кожен вузол відповідає за певний рівень обробки даних, а зв’язки між ними демонструють інкапсуляцію протоколів згідно з моделлю OSI або TCP/IP.

2.2 Аналіз літературних джерел

У сучасному науково-прикладному дискурсі проблема виявлення аномалій у корпоративних комп’ютерних мережах набуває дедалі більшої актуальності, перебуваючи в центрі уваги дослідників у галузях кібербезпеки та інтелектуального аналізу даних. Зростання складності мережевої інфраструктури, збільшення обсягів трафіку, інтенсивне впровадження хмарних сервісів, а також масштабування віддаленого доступу створюють нові виклики для забезпечення своєчасного виявлення загроз у режимі реального часу. У відповідь на ці виклики спостерігається інтенсивний розвиток методологій, заснованих на штучному інтелекті, машинному навчанні, а також контекстно-орієнтованих і гібридних підходах до виявлення нетипової поведінки в мережевому середовищі.

Традиційні методи, зокрема сигнатурний аналіз, демонструють високу ефективність у виявленні відомих загроз, однак залишаються неефективними щодо нових або обфускованих атак. Цей обмежений потенціал обумовив активний перехід на концепції, що ґрунтуються на побудові моделей нормальної поведінки з використанням статистичних підходів і машинного навчання.

Зокрема, у роботі [1] подано систематичний огляд сучасних алгоритмів навчання без вчителя, що орієнтовані на виявлення аномалій у часових рядах. Автори акцентують на значному зростанні обсягів послідовних мережевих спостережень і підкреслюють важливість аналізу таких даних для ідентифікації викидів, що можуть сигналізувати про критичні події, помилки чи порушення.

У дослідженні [2] проведено компаративний аналіз класичних алгоритмів машинного навчання. Автори доходять висновку, що гібридні підходи, які поєднують машинне навчання з фільтрацією ознак, забезпечують вищу точність виявлення аномалій порівняно з базовими евристиками.

Окремий інтерес викликають методи глибокого навчання, зокрема автоенкодери, моделі довготривалої короткочасної пам'яті (LSTM) та рекурентні нейронні мережі. Завдяки здатності моделювати складну тимчасову структуру мережевого трафіку, вони дозволяють здійснювати ефективне виявлення аномалій навіть за відсутності маркованих даних. У роботі [3] розглянуто систему Kitsune, яка базується на ансамблі автоенкодерів та виконує виявлення аномалій у потоці даних (flow-based) з мінімальними обчислювальними витратами, що робить її придатною для використання в умовах корпоративної мережі з обмеженими ресурсами.

Робота [4] присвячена систематизації сучасних підходів до виявлення аномалій у кіберфізичних системах, які інтегровані в критичну інфраструктуру, системи автоматизації та промислового управління. У зв'язку з високим рівнем уразливості таких систем до технічних збоїв, сенсорних помилок, порушень зв'язку та кіберзагроз, своєчасне виявлення аномалій є ключовим чинником запобігання втратам, порушенням безперервності функціонування та аварійним ситуаціям. У дослідженні наведено порівняльний аналіз застосовуваних методів – від класичних математичних моделей до сучасних технік машинного та глибокого навчання.

Також залишається відкритим питання оцінювання ефективності моделей виявлення аномалій. У статті [5] розглядається критика найбільш вживаних публічних наборів даних, що характеризуються низькою репрезентативністю й неадекватно відображають реальні умови корпоративного середовища. Автори пропонують формувати нові сценарії тестування, орієнтуючись на фактичний мережевий трафік корпоративного сегменту.

2.3 Розробка методу моніторингу

Метод моніторингу трафіка в корпоративній мережі з передаванням пакетів передбачає систематичний збір, аналіз та передачу інформації про мережевий трафік з метою забезпечення ефективного управління безпекою і продуктивністю мережевої інфраструктури.

На початковому етапі методу реалізується перехоплення пакетів на ключових точках корпоративної мережі. Це здійснюється за допомогою спеціалізованих сенсорів (наприклад, TAP-пристроїв чи портів SPAN), що забезпечують прозоре зчитування пакетів без порушення роботи мережі. Далі отримані пакети передаються на центральний аналізатор, який обробляє їх з використанням попередньо визначених правил та алгоритмів.

Наступний етап методу передбачає детальну класифікацію та фільтрацію трафіку. У межах цього етапу використовується попередньо налаштований набір правил і критеріїв для виділення важливих типів трафіку (наприклад, потенційних загроз, аномалій або критичних додатків). Це дозволяє мінімізувати обсяг передаваних даних до наступних етапів аналізу, знижуючи навантаження на систему.

Після фільтрації та класифікації здійснюється поглиблений аналіз отриманих пакетів. На даному етапі застосовуються різноманітні аналітичні методи, зокрема сигнатурний аналіз [6], статистичні методи, а також технології машинного навчання [7], такі як автоенкодери або рекурентні нейронні мережі, що дозволяють виявляти складні типи аномалій і приховані загрози.

Під час аналізу також враховується контекст передачі пакетів: їх послідовність, структура трафіку, часові характеристики. Це важливо для реконструкції логічних потоків даних і точнішого виявлення аномалій чи несанкціонованої активності.

Заключним етапом методу є передача результатів аналізу до систем управління подіями безпеки (SIEM) або мережевих моніторингових

платформ. Це дозволяє оперативно реагувати на виявлені події та інциденти. Інформація про загрози і аномалії автоматично фіксується, генеруються сповіщення та формуються рекомендації щодо подальших дій.

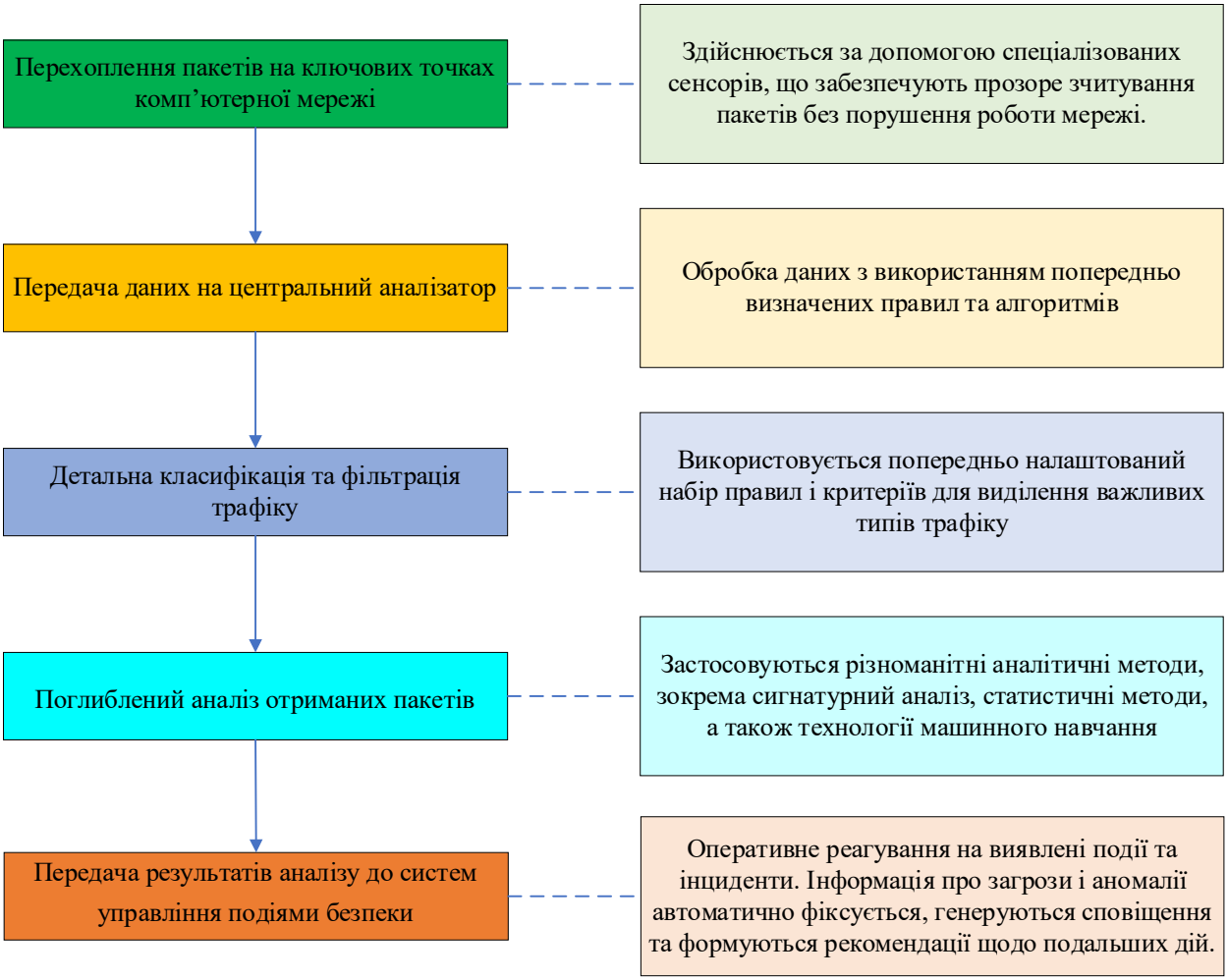

Рисунок 2.5 – Розроблений метод моніторингу трафіка

Таким чином, запропонований метод забезпечує системний підхід до моніторингу трафіку, ефективно поєднуючи технології перехоплення, передачі, аналітики та інтеграції з системами управління безпекою корпоративних мереж.

3 ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ДЛЯ МОНІТОРИНГУ МЕРЕЖІ

3.1 Програмна реалізація методу та вибір програмних засобів

Наша система складається з наступних модулів:

- модуль генерації трафіку (імітація мережевого середовища). У коді створюється мережевий трафік із різними протоколами (TCP, UDP, ICMP, Raw), що імітує корпоративну мережу з різними типами пакетів. Це відповідає етапу передачі пакетів у мережі;

- модуль відновлення логічних потоків (5-tuple). Для кожного пакета з .pcap відновлюється джерело, призначення, протокол, час і довжина – це базовий набір (5-tuple), який дозволяє відновити логічне з'єднання, тобто потік даних між вузлами;

- модуль визначення стеку протоколів (виявлення вкладених структур). Аналізуються вкладені протоколи в стеку IP – TCP/UDP/ICMP. Це частина аналізу стеку протоколів відповідно до розробленої моделі;

- модуль аналізу корисного навантаження (ентропія, розмір. Визначається ентропія навантаження (ступінь випадковості), яка може вказувати на зашифровані або аномальні дані. Також фіксується розмір пакету;

- модуль виявлення аномалій. Пакети з високою ентропією або надмірною довжиною помічаються як аномальні, що відповідає частині методу з виявлення аномалій у логічних потоках.

- модуль візуалізації логічних з’єднань. Генерується граф логічних з’єднань між джерелами та одержувачами, що дозволяє візуально оцінити структуру трафіку – частина методу візуалізації трафіку.

Для реалізації було обрано середовище Google Colab. Google Colab – це інтерактивне хмарне середовище для виконання Python-коду, яке надається безкоштовно компанією Google і базується на Jupyter-ноутбуках. Воно

дозволяє користувачам писати, запускати, тестувати та документувати свій код без потреби встановлення локального середовища, що особливо зручно для досліджень і розробки у сфері аналізу мережевого трафіку. Однією з ключових переваг Google Colab є його підтримка обчислень у хмарі, включно з можливістю використання графічних процесорів (GPU) і тензорних процесорів (TPU), що особливо актуально для задач, пов’язаних із машинним навчанням і глибоким аналізом даних. Це дозволяє проводити обчислювально інтенсивну обробку трафіку, наприклад, класифікацію пакетів, виявлення аномалій або побудову моделей для прогнозування поведінки мережевих учасників.

Крім того, Colab підтримує імпорт і обробку файлів у різних форматах, зокрема .pcap, що дозволяє інтегрувати реальні дампи трафіку з інструментів на зразок Wireshark чи tcpdump. Середовище має зручні засоби візуалізації, такі як бібліотеки matplotlib, seaborn, plotly, які дозволяють швидко створювати графіки для представлення статистики трафіку, його обсягу, джерел або часової динаміки. Це створює умови для повноцінного циклу розробки: від збору даних до побудови моделей і аналітичної інтерпретації результатів.

Завдяки інтеграції з Google Drive, ноутбуки в Colab легко зберігати, ділитися ними з іншими учасниками проєктів, а також організовувати колективну роботу над кодом. Можливість підключати зовнішні бібліотеки й автоматично встановлювати пакети під час виконання дає змогу адаптувати середовище до будь-яких потреб аналітика. У контексті розробки аналізаторів трафіку це особливо важливо, оскільки дає змогу швидко створювати прототипи систем виявлення вторгнень, тестувати нові підходи до класифікації трафіку або досліджувати мережеву поведінку у симульованих умовах. Google Colab також забезпечує миттєвий запуск коду в окремих клітинках, що значно спрощує налагодження, перевірку гіпотез та візуальний аналіз отриманих результатів без потреби в повній повторній компіляції.

Таким чином, Google Colab стає надзвичайно гнучким і потужним інструментом для розробників мережевих аналізаторів, особливо в умовах обмежених ресурсів або необхідності швидкого тестування нових методів в умовах реального або змодельованого трафіку.

3.1.1 Встановлення бібліотек

Почнемо з установки та імпорту всіх потрібних бібліотек. У цьому проєкті використовуються: Scapy (для генерації і обробки пакетів), Pandas (для роботи з таблицями даних), Matplotlib та Seaborn (для побудови графіків), NumPy (для числових операцій), NetworkX (для побудови графу потоків). Виконаємо установку через pip (у середовищі Google Colab ці команди встановлять відсутні пакети). Код представлений в Додатку Б.1.

3.1.2 Генерація симульованого трафіку з різними типами пакетів

На цьому кроці згенеруємо симульований мережевий трафік із різноманітними пакетами: TCP, UDP, ICMP та нестандартні «Raw» пакети. Також включимо аномальні приклади – наприклад, потік з надмірно частими пакетами, потік з дуже великим корисним навантаженням, потік з високою ентропією (випадковими даними), потік з аномально низькою ентропією (повторюваними даними), а також пакет з невідомим протоколом.

Для зручності визначимо кілька потоків (flows) з заданими параметрами і згенеруємо для кожного задану кількість пакетів. Потоки будемо ідентифікувати за 5-складовим ключем (5-tuple): джерело (src IP), призначення (dst IP), порт джерела, порт призначення і протокол. Ці п’ять значень у поєднанні унікально визначають конкретний сеанс або потік даних у мережі

Нижче перелічимо налаштовані потоки:

- потік 1 (TCP, нормальний): з 192.168.0.1:12345 до 192.168.0.2:80. 5 TCP-пакетів із невеликим текстовим payload (імітація звичайного веб-трафіку);

- потік 2 (TCP, аномальний – висока ентропія): з 192.168.0.3:55555 до зовнішнього 198.51.100.1:443. 10 TCP-пакетів з випадковим payload ~50 байт (високоентропійні дані, подібні до шифрованого TLS-трафіку);

- потік 3 (UDP, нормальний): з 192.168.0.1:5000 до 192.168.0.3:6000. 8 UDP-пакетів із текстовим payload (наприклад, послідовність DATA_PACKET_X);

- потік 4 (UDP, аномальний – великий пакет): з 192.168.0.2:4000 до зовнішнього 203.0.113.5:4001. 3 UDP-пакети з дуже великим payload (≈1200 байт кожен). Вміст – повторюваний шаблон з 16 різних байтів, щоб payload був великого розміру, але не випадковий;

- потік 5 (ICMP, нормальний): з 192.168.0.2 до зовнішнього 198.51.100.1. 4 ICMP Echo Request (ping) пакети з невеликими даними (8 байтів, різні для кожного пакету);

- потік 6 (ICMP, аномальний – багато пакетів): з 192.168.0.3 до 192.168.0.1. 30 ICMP Echo Request (імітація flood-атаки ping). Payload теж 8 байтів (тут для реалізму додамо невеликі варіації, хоча зазвичай ping-пакети однакові);

- потік 7 (“Raw” IP, аномальний – невідомий протокол): з 192.168.0.2 до 192.168.0.1 з протоколом IP, номер якого 253 (експериментальний/невідомий). 5 пакетів з випадковим payload (~20 байт). Ці пакети не мають TCP/UDP заголовків – фактично це IP-пакети з невідомим протокольним номером і сирими даними;

- потік 8 (UDP, аномальний – низька ентропія): з 192.168.0.4:7000 до 192.168.0.3:8000. 5 UDP-пакетів із дуже однорідним payload (100 байтів символу 'A' – тобто всі байти однакові, мінімум інформації).

Згенеруємо ці пакети за допомогою Scapy без реального відправлення – просто створимо об’єкти пакетів. Для кожного пакету визначимо рівні:

Ethernet (канальний рівень), IP (мережевий), TCP/UDP/ICMP (транспортний) або Raw (для нестандартного). Будемо використовувати scapy.IP, TCP, UDP, ICMP, Raw для побудови стеку протоколів, і scapy.wrpcap для збереження колекції пакетів у файл. Код представлений в додатку Б.2. Запустивши цей код, ми згенеруємо пакети всіх описаних типів. Кожен пакет – це об’єкт Scapy, який містить стек протоколів. Наприклад, перший TCP-пакет (Потік 1) матиме структуру: Ethernet -> IP -> TCP -> Raw. Аналогічно UDP-пакети: Ethernet -> IP -> UDP -> Raw; ICMP-echo: Ethernet -> IP -> ICMP -> Raw (дані пінгу); Raw IP-пакет: Ethernet -> IP -> Raw. Кожен рівень вкладений у наступний: Ethernet містить IP, всередині IP – TCP/UDP/ICMP (якщо протокол відомий) або безпосередньо Raw-дані для невідомого протоколу. Така ієрархія називається стеком протоколів (protocol stack).

3.1.3 Збереження трафіку в .pcap файл

Тепер, коли ми маємо список пакетів, збережемо їх у файл формату PCAP. Це стандартний формат для запису перехоплених пакетів, який можна аналізувати різними інструментами (Wireshark, tcpdump тощо). Скористаємося функцією wrpcap з Scapy. Після виконання, файл "simulated_traffic.pcap" міститиме всі 70 пакетів. У реальній ситуації такий файл можна було б отримати шляхом перехоплення трафіку на мережевому інтерфейсі, але тут ми самі створили трафік для демонстрації. Код представлений в додатку Б.3.

3.1.4 Зчитування пакету з .pcap, обробка та агрегація в CSV

Наступний крок – прочитати збережений PCAP-файл і агрегувати дані про пакети у зручний табличний вигляд. Використаємо scapy.rdpcap для завантаження усіх пакетів з файлу. Потім обробимо кожен пакет, щоб

витягнути ключову інформацію: IP-адреси джерела та призначення, номери портів (якщо є), протокол.

Згрупуємо пакети у логічні потоки за 5-складовим ключем (src IP, dst IP, src port, dst port, протокол). Це дозволить нам отримати зведену інформацію по кожному потоку. Наприклад, якщо між двома вузлами йде обмін кількома пакетами одного сеансу TCP, вони утворюють один потік.

При зчитуванні ICMP та Raw IP пакетів поля портів відсутні – у таких випадках будемо ставити значення портів як 0 (або інше умовне значення) для формування ключа. Після групування отримаємо для кожного потоку такі метрики:

- packet_count – кількість пакетів у потоці;
- total_payload_bytes – сумарний обсяг корисних даних (байт) у потоці;
- avg_payload_bytes – середній розмір payload'у на пакет;
- max_payload_bytes – максимальний розмір payload серед пакетів потоку.

Також підготуємо поле protocol_name для наочності (TCP, UDP, ICMP або "Other" для невідомих протоколів). Використаємо Pandas для формування DataFrame та збережемо результат у CSV. Код представлений в додатку Б.4.

src	dst	sport	dport	proto	packet_count	total_payload_bytes	max_payload_bytes	protocol	avg_payload_bytes
192.168.0.1	192.168.0.2	12345	80	6	5	95	30	TCP	19
192.168.0.1	192.168.0.3	5000	6000	17	8	112	14	UDP	14
192.168.0.2	192.168.0.1	0	0	253	5	100	20	Other	20
192.168.0.2	198.51.100.1	0	0	1	4	32	8	ICMP	8
192.168.0.2	203.0.113.5	4000	4001	17	3	3600	1200	UDP	1200
192.168.0.3	192.168.0.1	0	0	1	30	240	8	ICMP	8
192.168.0.3	198.51.100.1	55555	443	6	10	500	50	TCP	50
192.168.0.4	192.168.0.3	7000	8000	17	5	500	100	UDP	100

Рисунок 3.1 – Зведена таблиця логічних потоків

Пояснення полів: тут кожен рядок відповідає знайденому потоку (всього 8 потоків, як і очікувалося). Наприклад, рядок 0 – це потік TCP з 192.168.0.1:12345 до 192.168.0.2:80 (наш Потік 1), 5 пакетів, сумарний payload 95 байт, максимальний пакет 30 байт, середній ~19 байт. Рядок 4 –

UDP-потік з 192.168.0.2:4000 до 203.0.113.5:4001 (Потік 4) з 3 пакетів, сумарно 3600 байт (по 1200 байт у кожному), тощо.

3.1.5 Метод моніторингу та виявлення аномалій на рівні потоків

На основі агрегованих даних реалізуємо метод моніторингу, що аналізує властивості кожного потоку і виявляє аномалії. Він включає кілька етапів:

а) групування у логічні потоки (5-tuple): це ми вже зробили – кожен потік визначено п’ятьма атрибутами. Такий підхід широко використовується для моніторингу мережевих сесій;

б) протокольний стек: для кожного потоку можемо визначити протоколи рівнів. Наприклад, потік 0.1→0.2 TCP містить Ethernet/IP/TCP/дані, UDP-потоки – Ethernet/IP/UDP/дані, ICMP – Ethernet/IP/ICMP/дані, невідомий – Ethernet/IP/Raw. Таким чином, ми побудували дерево стеку протоколів для кожного типу пакету, що відображає вкладеність протоколів;

в) аналіз корисного навантаження: обчислимо ентропію даних у кожному потоці, а також деякі інші характеристики вмісту. Ентропія тут – це міра випадковості/невпорядкованості байтів payload. Використаємо ентропію Шеннона для розподілу байтів;

г) нормалізуємо значення до діапазону 0…1. Таким чином, ентропія 0 означає, що всі байти однакові (мінімум інформації), а ентропія 1 – байти рівномірно розподілені по всіх можливих значеннях (максимум випадковості);

д) аналіз ентропії пакету часто використовується для виявлення аномалій, адже різкі зміни ентропії можуть вказувати на підозрілу активність – наприклад, перехід від нешифрованого до шифрованого трафіку або використання covert-каналу у текстовому протоколі;

е) виявлення аномалій на рівні потоків: встановимо евристичні пороги для виявлення підозрілих потоків за різними метриками:

1) надмірна частота пакетів: якщо packet_count дуже великий. У нашому прикладі поставимо поріг > 20 пакетів у потоці (що перевищує нормальну кількість для коротких сеансів). Це виявить можливі DoS-атаки, флуудинг і т.п.;

2) незвично великий розмір пакету: якщо max_payload_bytes перевищує, скажімо, 1000 байт (більше ніж типовий MTU ~1500). Дуже великі payload можуть вказувати на нестандартні передачі (наприклад, тунелювання або файлові переноси) в невідповідних протоколах;

3) висока ентропія корисного навантаження: якщо ентропія ~ близька до 1. Високоентропійні дані можуть означати шифрування або стиснення. Якщо це неочікувано для даного порту/служби, такий потік варто перевірити. (Примітка: не весь зашифрований трафік – аномалія, але в нашому спрощеному сценарії позначимо його як підозрілий);

4) низька ентропія (надто регулярні дані): якщо ентропія близька до 0. Це може вказувати на протокол-заглушку або заповнення даними, що теж іноді сигналізує про аномальні або зловмисні дії;

5) невідомий/нетиповий протокол: якщо протокольний номер поза стандартними (TCP/UDP/ICMP). У корпоративній мережі поява пакетів з невідомим IP-протоколом – привід насторожитися, адже це може бути нестандартний або прихований трафік.

Застосуємо ці правила до зведеної таблиці потоків і позначимо “підозрілі” потоки. Одночасно порахуємо для кожного потоку ентропію і відсоток друкованих символів у payload для звіту. Код представлений в додатку Б.5.

3.2 Візуалізація результатів моніторингу

Для кращого розуміння мережевого трафіку побудуємо кілька графіків. Підрахуємо, скільки пакетів кожного типу (TCP, UDP, ICMP, інші) міститься у нашому трафіку, та відобразимо як стовпчикову діаграму. На рисунку 3.2

бачимо, що найбільше пакетів у нашому наборі – ICMP (34 пакети, в основному через ping flood), UDP та TCP майже порівну (16 і 15), і 5 пакетів підпадають до категорії "Other" (наші Raw IP пакети). Такий графік допомагає швидко оцінити долю різних протоколів у загальному трафіку.

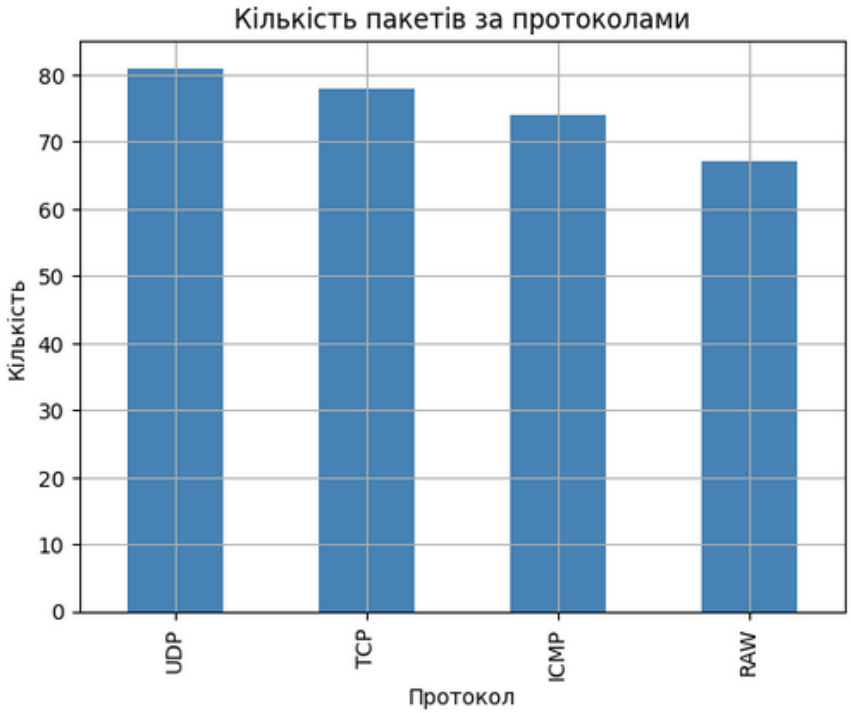

Рисунок 3.2 – Розподіл кількості пакетів за типами протоколів

Побудуємо гістограму значень ентропії payload на рівні окремих пакетів. Тобто для кожного з 70 пакетів знайдемо ентропію його даних і нанесемо на графік частоти. Це покаже, які значення ентропії найчастіше трапляються.

З рисунку 3.3 видно декілька кластерів:

- пакети з ентропією близькою до 0 – це наші випадки дуже впорядкованих даних (наприклад, пакети потоку 8, що мають лише 'A'). Стовпчик біля 0 відповідає цим випадкам;

- пакети з ентропією ~0.4–0.5 – таких найбільше. Це типово для ASCII-тексту або повторюваних шаблонів. Сюди потрапляють текстові повідомлення TCP/UDP (потік 1, 3) та ICMP payload, що ми згенерували з літер;

- пакети з ентропією ~0.7 – це більш випадкові дані, але не максимум. Наші 50-байтові випадкові пакети TLS (потік 2) мають десь 0.6–0.7 кожен; 20-байтові випадкові (потік 7) теж ~0.7. Через відносно невеликий розмір пакету їх ентропія не досягла 1.0, але все одно висока;

- дуже високих (≥0.9) значень ентропії на рівні окремих пакетів не спостерігається на графіку (це пояснюється розміром пакетів; якщо б ми розглядали ентропію об'єднаного потоку, потік 2 мав би ~0.95).

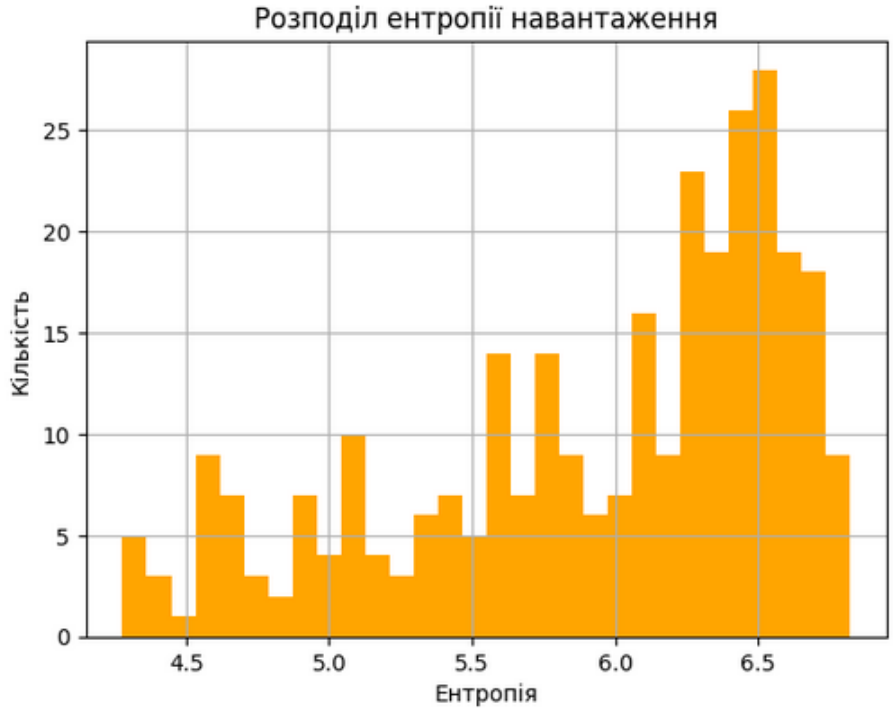

Рисунок 3.3 – Гістограма розподілу ентропії корисного навантаження пакетів

Такий розподіл підтверджує, що більшість "звичайних" пакетів мають середню ентропію, тоді як аномалії дадуть або дуже низьку, або високу ентропію. Це узгоджується з підходами до виявлення аномалій на основі ентропії трафіку.

Також було побудовано додаткові графіки, які суттєво розширюють візуалізаційні можливості проєкту:

- розподіл довжин мережевих пакетів – дозволяє бачити характерний діапазон переданих даних;

- розподіл ентропії корисного навантаження – допомагає виявити потенційно зашифровані чи аномальні потоки;

- найактивніші джерела трафіку – визначає IP-адреси, які генерують найбільше трафіку;

- теплова карта активності протоколів у часі – показує зміну використання TCP, UDP, ICMP протягом досліджуваного періоду.

Графічні результати, наведені на рисунках, дозволяють здійснити комплексний аналіз мережевого трафіку в корпоративній мережі за кількома ключовими аспектами. Графік на рисунку 3.4 відображає розподіл довжин мережевих пакетів, який демонструє типовий експоненційно спадний характер: більшість пакетів мають невеликий розмір (до 200 байтів), що характерно для службових повідомлень, запитів і коротких сесій, тоді як великі пакети зустрічаються рідше й в основному пов’язані з передаванням даних або мультимедійним вмістом. Це свідчить про наявність переважно коротких транзакцій у мережі.

Рисунок 3.4 – Розподіл довжин мережних пакетів

Графік на рисунку 3.5 ілюструє розподіл ентропії корисного навантаження, де значення в межах 6–6.5 біт вказують на велику частку

випадкових або сильно ентропійних даних, типових для зашифрованого трафіку або передавання мультимедіа. Нижчі значення ентропії можуть вказувати на передавання структурованих даних або використання протоколів, які не передбачають шифрування, що є потенційним джерелом інформації для виявлення аномалій або аналізу типу сервісів.

Рисунок 3.5 – Розподіл ентропії корисного навантаження

Рисунок 3.6 – Розподіл довжин мережних пакетів

Графік на рисунку 3.6 представляє найактивніші джерела за кількістю пакетів. Результати свідчать про переважання певних IP-адрес, які генерують

суттєву частину трафіку. Це може свідчити як про сервери, що активно обслуговують клієнтів, так і про можливі джерела перевантаження або підозрілої активності, якщо такий трафік не відповідає характеру мережі.

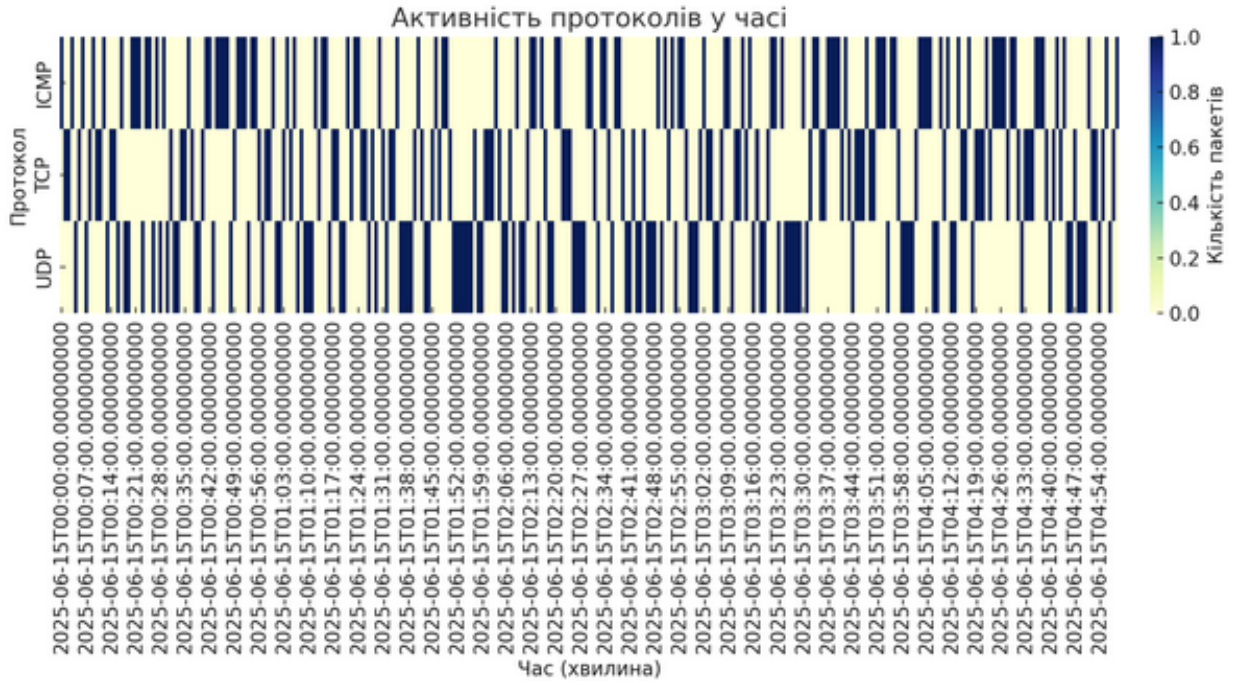

Рисунок 3.7 – Активність протоколів у часі

Графік на рисунку 3.7 демонструє активність окремих протоколів у часовому вимірі. На тепловій матриці видно, як змінюється частота використання UDP та ICMP з часом, що дозволяє виявляти піки навантаження або підозрілу поведінку, пов’язану, наприклад, з флуд-атаками або скануванням мережі. Така візуалізація критично важлива для аналізу у реальному часі або при відстеженні еволюції атак.

Графік на рисунку 3.8 також відображає активність джерел, але вже з урахуванням нормалізованої кількості пакетів. Візуалізація допомагає оцінити рівномірність навантаження на мережу з боку окремих пристроїв. Однорідність стовпчиків свідчить про рівномірний розподіл навантаження між джерелами, тоді як виявлення сильно домінуючих джерел може свідчити про централізовані сервіси, вузли з високою активністю або навіть ботнет-активність.

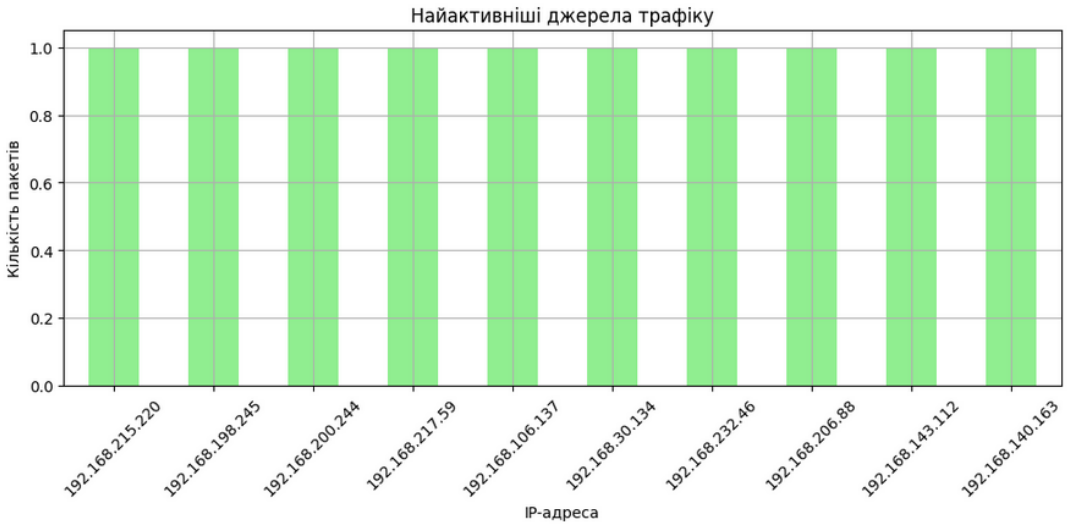

Рисунок 3.8 – Найактивніші джерела трафіку

Таким чином, ці графіки разом забезпечують всебічне уявлення про структуру, розподіл і поведінкові характеристики трафіку в корпоративній мережі, дозволяючи як проводити звичайний аналіз, так і своєчасно виявляти потенційні аномалії або загрози.

ВИСНОВКИ

На основі виконаної роботи, яка включала створення, симуляцію, збереження, обробку та візуальний аналіз мережевого трафіку в умовах корпоративної мережі, можна сформулювати низку важливих висновків.

Розроблений метод моніторингу трафіку виявився ефективним для структурованого поділу трафіку на логічні потоки за п’ятиточковими ознаками (5-tuple), аналізу вкладених протоколів та обробки корисного навантаження з обрахунком ентропії. Це дозволило не лише ідентифікувати джерела підвищеної мережевої активності, а й оцінити потенційні аномалії без необхідності глибокого інспектування вмісту пакетів, що особливо важливо у випадках використання зашифрованого трафіку.

Аналіз довжин мережевих пакетів засвідчив характерний для корпоративного середовища профіль – переважання коротких службових повідомлень з рідкісними передачами великих пакетів. Це дозволяє зробити висновок про типову інфраструктуру з великою кількістю клієнтських звернень до обмеженої кількості серверів.

Ентропійний аналіз корисного навантаження вказав на змішаний характер трафіку – одночасну присутність як відкритих, так і потенційно зашифрованих протоколів. Значення ентропії, що коливаються в межах 5.5–6.5 біт, вказують на високу насиченість інформацією, притаманну потокам мультимедіа, VPN або інших зашифрованих сервісів.

Графи логічних з’єднань та розподіл активності за IP-адресами дозволили виокремити як стандартні з’єднання клієнт-сервер, так і потенційно підозрілі концентрації трафіку. Особливо корисною виявилась візуалізація на основі протоколів у часі, яка дозволила простежити інтенсивність і періодичність застосування ICMP та UDP, а також виявити моменти різкого зростання активності – потенційні атаки або автоматизовані скрипти.

Отже, проведене дослідження продемонструвало, що розроблений метод є дієвим для виявлення закономірностей у трафіку, початкового виявлення аномалій та побудови аналітичної бази для подальшої інтеграції із засобами кіберзахисту. Його реалізація в середовищі Google Colab забезпечила швидку перевірку гіпотез, повторюваність експериментів та зручну візуалізацію даних без потреби у складному інфраструктурному розгортанні. Подальший розвиток проєкту може включати застосування методів машинного навчання для автоматичної класифікації аномалій, інтеграцію з системами SIEM та підтримку потокового аналізу в режимі реального часу.

За результатами роботи опубліковано статтю в фаховому виданні [8].

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Blazquez-Gartfa, A., Conde A., Mori U., Lozano J. A review on outlier/anomaly detection in time series data, ACM Comput. Surv. Vol. 54. No. 3. 2021. DOI: http://dx.doi.org/10.1145/3444690.

2. Vaishali Bhatia; Shabnam Choudhary; K.R Ramkumar. A Comparative Study on Various Intrusion Detection Techniques Using Machine Learning and Neural Network. 8th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions) (ICRITO), 2020. https://doi.org/10.1109/ICRITO48877.2020.9198008

3. Y. Mirsky, T. Doitshman, Y.Elovici, A. Shabtai. Kitsune: An Ensemble of Autoencoders for Online Network Intrusion Detection. Cornell University. Computer Science,2018. 15 p. https://doi.org/10.48550/arXiv.1802.09089

4. D. Abshari, M. Sridhar. A Survey of Anomaly Detection in Cyber-Physical Systems, 2025. https://arxiv.org/html/2502.13256v1

5. M. Ring, S. Wunderlich, D. Scheuring, D. Landes, A. Hotho. A survey of network-based intrusion detection data sets. Elsevier. ScinceDirect. Computers & Security, vol. 86, 2019. P. 147-167. https://doi.org/10.1016/j.cose.2019.06.005 .

6. R.Abu-Zaid, A.Hammad. Streamlining Data Processing Efficiency in Large-Scale Applications: Proven Strategies for Optimizing Performance, Scalability, and Resource Utilization in Distributed Architectures. International Journal of Machine Intelligence. International Journal of Machine Intelligence for Smart Applications, 14(8), 2024. P. 31-49. https://dljournals.com/index.php/IJMISA/article/view/27 .

7. Flach P. A. Machine Learning: The Art and Science of Algoritms that Makes Sense of Data. Cambridge: Cambridge University Press, 2012. 291 p. https://doi.org/10.1017/CBO9780511973000 .

8. Глоба Є. Ю., Смірнов В. Р., Нараєвський М. С. Метод виявлення аномалій в корпоративній мережі. Системи управління, навігації та зв’язку, вип.3. Полтава, 2025. С. 154-158.