Міністерство освіти і науки України

Національний технічний університет
«Дніпровська політехніка»

Навчально-науковий
інститут електроенергетики
(інститут)

Факультет інформаційних технологій
(факультет)
Кафедра інформаційних технологій та комп'ютерної інженерії
(повна назва)

ПОЯСНЮВАЛЬНА ЗАПИСКА
кваліфікаційної роботи ступеня бакалавра

здобувача Гаврон Олександр Олександрович
(ПІБ)

академічної групи 123-21-1
(шифр)

спеціальності 123 Комп'ютерна інженерія
(код і назва спеціальності)

за освітньо-професійною програмою Комп'ютерна інженерія
(офіційна назва)

на тему "Віддалене перехоплення мережевого трафіку як інструмент забезпечення інформаційної безпеки в корпоративних мережах"
(назва за наказом ректора)

Керівники	**Прізвище, ініціали**	**Оцінка за шкалою**		**Підпис**
		рейтинговою	**інституційною**	
кваліфікаційної роботи	доц. Селівьорстова Т.В.			
загального розділу	доц. Селівьорстова Т.В.			
спеціального розділу	доц. Селівьорстова Т.В.			
Рецензент				
Нормоконтролер	проф. Цвіркун Л.І.			

Дніпро
2025

ЗАТВЕРДЖЕНО:
завідувач кафедри
інформаційних технологій
та комп'ютерної інженерії
(повна назва)
_____ Гнатушенко В.В.
(підпис) (прізвище, ініціали)

"25" січня 2025 року

ЗАВДАННЯ
на кваліфікаційну роботу
ступеня бакалавр

здобувача Гаврон О.О. **академічної групи** 123-21-1
(прізвище та ініціали) (шифр)

спеціальності 123 Комп'ютерна інженерія

за освітньо-професійною програмою Комп'ютерна інженерія
(офіційна назва)

на тему " Віддалене перехоплення мережевого трафіку як інструмент забезпечення інформаційної безпеки в корпоративних мережах "

затверджену наказом ректора НТУ «Дніпровська політехніка» від 05.05.2025 № 336-с

Розділ	Зміст	Термін виконання
Загальний розділ	На основі матеріалів виробничих практик, інших науково-технічних джерел показати актуальність завдання, сформулювати мету та задачі виконання кваліфікаційної роботи	10.02.2025
Спеціальний розділ	Сформулювати найменування й призначення системи, висунути технічні вимоги до нього	15.03.2025
	Розробити архітектуру системи, обґрунтування технічних характеристик.	20.04.2025
	Реалізувати програмний інструмент, використовуючи відповідні інструменти програмування та враховуючи принципи розробки надійних та ефективних програмних систем, що є частиною комп'ютерної інженерії	07.05.2025
	Протестувати розроблений програмний інструмент на предмет її функціональності, продуктивності обміну даними в реальному часі та стійкості до можливих збоїв	31.05.2025

Завдання видано _____ доц. Селівьорстова Т.В.
(підпис керівника) (прізвище, ініціали)

Дата видачі 25.01.2025

Дата подання до екзаменаційної комісії _____

Прийнято до виконання _____ Гаврон О.О.

РЕФЕРАТ

Пояснювальна записка: 63 с., 40 рис., 3 табл., 1 додаток, 11 джерел.

Об'єкт роботи – мережевий трафік в корпоративних мережах та процеси його передачі.

Предмет роботи – методи та інструменти віддаленого перехоплення мережевого трафіку з використанням протоколу RPCAP для забезпечення інформаційної безпеки та адміністрування корпоративних мереж.

Метою роботи є розробка та дослідження методики віддаленого перехоплення мережевого трафіку за допомогою протоколу RPCAP як інструменту для забезпечення інформаційної безпеки та ефективного адміністрування в корпоративних мережах.

Завдання:

1. Проаналізувати та обґрунтувати необхідність віддаленого перехоплення трафіку для інформаційної безпеки та адміністрування в сучасних корпоративних мережах.

2. Дослідити та порівняти існуючі методи та технології перехоплення мережевого трафіку, зокрема віддалені способи, такі як дзеркалювання портів та використання демона RPCAPD.

3. Вивчити протокол RPCAP та оцінити його ефективність як інструменту для віддаленого перехоплення трафіку.

4. Розробити та описати методику віддаленого перехоплення мережевого трафіку в корпоративних мережах.

5. Спроектувати та реалізувати програмний інструмент з графічним інтерфейсом на Python для демонстрації та практичного застосування розробленої методики віддаленого перехоплення трафіку за допомогою RPCAP.

6. Провести тестування розробленого інструменту та методики, проаналізувати отримані результати та сформулювати висновки щодо ефективності та доцільності їхнього застосування.

Ключові слова: RPCAP, мережевий трафік, адміністрування мереж.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАК, ОДИНИЦЬ ТА ТЕРМІНІВ

API – (Application Programming Interface) Інтерфейс програмування застосунків

BPF – (Berkeley Packet Filter) Фільтр пакетів Берклі

BSD – (Berkeley Software Distribution) Розповсюдження програмного забезпечення Берклі

CPU – (Central Processing Unit) Центральний процесор

DDoS – (Distributed Denial of Service) Розподілена відмова в обслуговуванні

GUI – (Graphical User Interface) Графічний інтерфейс користувача

GRE – (Generic Routing Encapsulation) Узагальнене інкапсулювання маршрутизації

IDS – (Intrusion Detection System) Система виявлення вторгнень

IPS – (Intrusion Prevention System) Система запобігання вторгненням

IP – (Internet Protocol) Інтернет-протокол

ЛОМ – Локальна обчислювальна мережа (Еквівалент LAN)

ОС – Операційна система

ПК – Персональний комп'ютер

ПЗ – Програмне забезпечення

RAM – (Random Access Memory) Оперативна пам'ять

RPCAPD – (Remote PCAP Daemon) Віддалений демон PCAP (протокол віддаленого захоплення пакетів)

SIEM – (Security Information and Event Management) Управління інформаційною безпекою та подіями

TCP – (Transmission Control Protocol) Протокол керування передачею

UDP – (User Datagram Protocol) Протокол дейтаграм користувача

UNIX – Операційна система UNIX

VPN – (Virtual Private Network) Віртуальна приватна мережа

ВСТУП

У сучасному світі автоматизація різних сфер стала повсюдним явищем, і передача інформації через канали зв'язку не є винятком. Ми спостерігаємо експоненціальне зростання кількості локальних мереж, що свідчить про їхню стрімку популярність.

Популярність Intranet зумовлена наявністю простого у використанні програмного забезпечення, зручністю обміну корпоративною інформацією, відпрацьованою технологією міжмережевого обміну та великою кількістю доступних інформаційних матеріалів. Однак, стрімкий ріст мереж призводить до зменшення контролю над ними, що виходить за межі початково встановлених параметрів передачі даних у окремих корпоративних мережах. Паралельно з розвитком технологій зростає й кількість потенційних мережевих атак, що посилює потребу в моніторингу для їх запобігання. Спостереження за мережею, зокрема, ґрунтується на аналізі інформації, що передається в ній. Виникає нагальна потреба в аналізі даних, які проходять через мережеві пристрої, оскільки саме вони є потенційними вразливими місцями. Отже, з'являється потреба в отриманні цих даних для подальшого аналізу.

Наразі існує безліч методів моніторингу мереж. Деякі з них активно застосовуються в інформаційних інфраструктурах, інші ж мають значний потенціал, але використовуються не повною мірою або ще недостатньо вивчені. У науковій сфері бракує робіт, присвячених методам перехоплення трафіку. Особливо відчутний дефіцит повноцінно описаних методик віддаленого перехоплення, а також відсутня оцінка ефективності їхнього використання [1].

Метою даної кваліфікаційної роботи є розробка віддаленого перехоплення трафіку. Ця методика буде являти собою готовий алгоритм, сукупність прийомів та способів отримання даних, а також порядок їх застосування, спрямованих на моніторинг трафіку та аналіз інформації, що передається в корпоративній мережі. Крім того, вона буде набором завдань для забезпечення віддаленого перехоплення в корпоративній мережі.

Об'єктом дослідження є протоколи, системи, інструменти та методи, що забезпечують перехоплення даних у мережі.

Предметом дослідження є ефективність використання існуючих об'єктів дослідження.

Методи дослідження включають аналіз, моделювання інфраструктури, теорію масового обслуговування та теорію обмежень.

Для досягнення поставленої мети в рамках цієї роботи необхідно вирішити наступні основні завдання:

- провести аналіз існуючих робіт та підтвердити актуальність теми;
- розглянути наявні підходи до моніторингу віддаленого трафіку;
- оцінити ефективність способів перехоплення трафіку;
- розробити методику для віддаленого перехоплення трафіку.

1 ЗАГАЛЬНИЙ РОЗДІЛ

1.1 Аналіз та систематизація інформації про наявні дослідження у сфері перехоплення даних

Аналіз та систематизація інформації про дослідження дасть змогу виявити наявні методи перехоплення трафіку, визначити їхні відмінності, а головне — підтвердити актуальність теми та ступінь її розробленості. Для цього потрібно ознайомитися з поточними дослідженнями в цій галузі. Результати дослідження робіт представлені в таблиці 1.1, де описано переваги та недоліки кожної роботи, а також міститься їхній короткий опис.

Таблиця 1.1 – Аналіз наявних досліджень

Назва роботи	Переваги	Недоліки	Висновок
Перехоплення та аналіз мережевого трафіку за допомогою «Wireshark» [1]	1. Наведено методи перехоплення трафіку. 2. Є приклад роботи з Wireshark.	3. Наведені методи перехоплення трафіку не описані прикладами, про них немає докладної інформації. 4. Не розглядаються інші сніфери та не порівнюються між собою. 5. Не розглядаються плюси та мінуси можливості перехоплення	Коротко описується робота зі сніфером «Wireshark».

		трафіку.	
Технологія перехоплення та аналізу трафіку в бездротовій Wi-Fi мережі [2]	1. Добре описується спосіб перехоплення бездротового трафіку на Windows та Linux.	1. Не розглянуто загалом існуючі методи. 2. Не описано, для чого потрібен перехоплення.	Опис методу перехоплення трафіку в бездротових мережах з використанням сніфера та адаптера.
Аналіз мережевого трафіку корпоративної мережі за допомогою програмного забезпечення Wireshark [5]	1. Розглядаються способи перехоплення трафіку. 2. Детально описується робота зі сніфером.	1. Не розглянуто способи перехоплення. 2. Не описано методи використання аналізаторів трафіку.	Описується робота зі сніфером Wireshark.
Перехоплення даних по мережі [3]	1. Розглядається класифікація. 2. Розглядає можливі джерела загроз.	1. Поверхневе пояснення. 2. Немає порівняння. 3. Немає прикладу використання.	Введення у вивчення перехоплення трафіку.
Перехоплення та аналіз мережевого трафіку за допомогою бібліотеки PCAP [4]	1. Описується створення програми, яка може стати основою для сніфера.	1. Не розглядаються способи перехоплення трафіку.	Створення консольної програми, що демонструє роботу бібліотеки PCAP.

Пасивний перехоплення трафіку [5]	1. Розглядаються способи реалізації перехоплення трафіку в мережі. 2. Проводиться порівняння способів реалізації. 3. Повністю описується реалізація одного зі способів.	1. Неактуальна інформація.	Ця стаття розповідає про те, як за допомогою комп'ютера, обладнаного двома мережевими картами, організувати повністю пасивний перехоплення мережевого трафіку Fast Ethernet.

1.2 Класифікація методів перехоплення трафіку

Перш ніж перейти до питання розробки методики віддаленого перехоплення трафіку, потрібно ознайомитися з основними поняттями, пов'язаними з темою роботи, а також з технологічними рішеннями, взаємодія з якими стане невід'ємною частиною проєкту. Структурними поняттями будуть системи та технології, що пов'язані з процесом перехоплення даних у мережах.

Існує кілька способів перехоплення трафіку. Перший – збір через конкретні хости мережевої інфраструктури. Зазвичай це пристрої рівня L3, наприклад, міжмережеві екрани, комутатори, комп'ютери. Плюси цього способу – швидка та легка реалізація. Мінуси – обмеження одним пристроєм та низька продуктивність. На рисунку 1.1 зображено приклад отримання інформації на конкретних хостах [2].

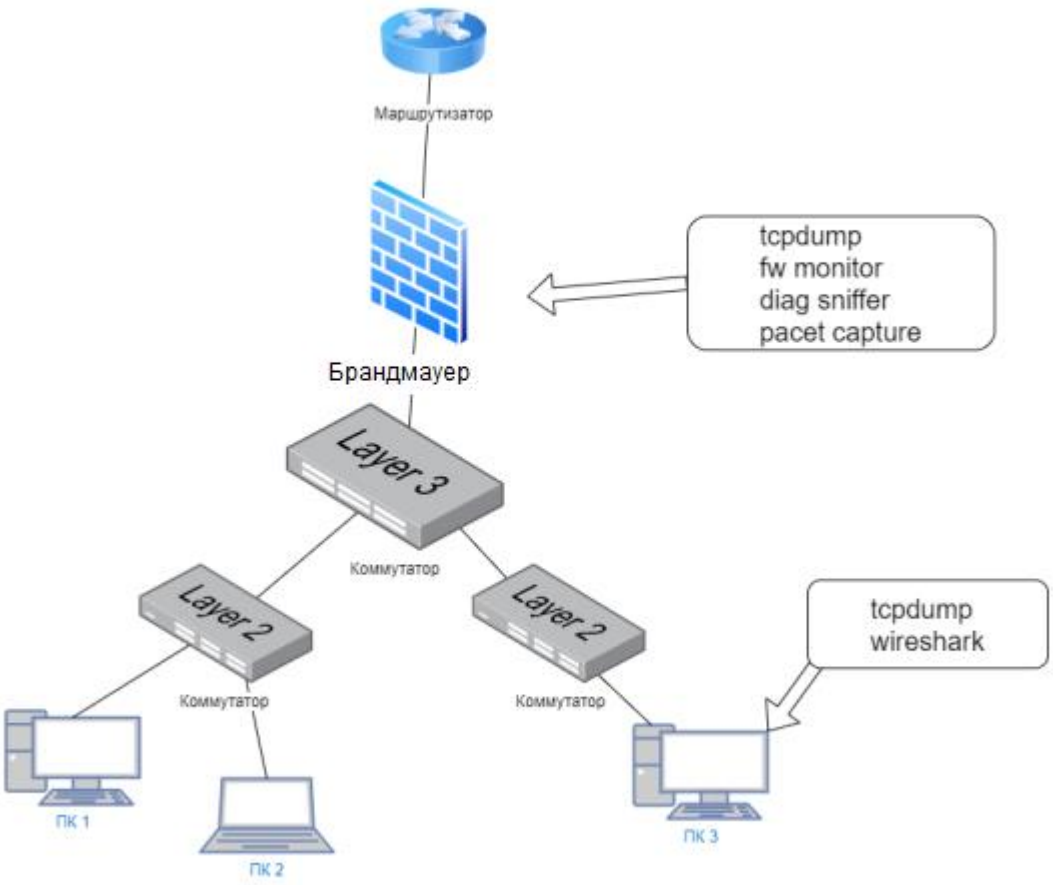

Рисунок 1.1 – Приклад мережі зі способом отримання інформації на конкретних хостах

Другий спосіб — це дзеркалювання трафіку, або дзеркалювання порту. У цьому випадку трафік збирається з фізичних лінків. Вихідний порт копіює потік даних, що передається між клієнтом і портом призначення. Потім скопійований потік даних надсилається на пристрій моніторингу.

При дзеркалюванні портів кожен пакет копіюється і проходить через інтерфейс до пристрою моніторингу. Цей метод має один мінус – великий обсяг передачі трафіку. Водночас, його плюсом є те, що не потрібно змінювати топологію мережі. На рисунку 1.2 зображено приклад мережі із застосуванням дзеркалювання трафіку. Як бачимо, з маршрутизатора та комутатора третього рівня, зображених на рисунку 1.2, дані надходять на окремий пристрій моніторингу з встановленими системами IDS, SIEM, IPS та іншими.

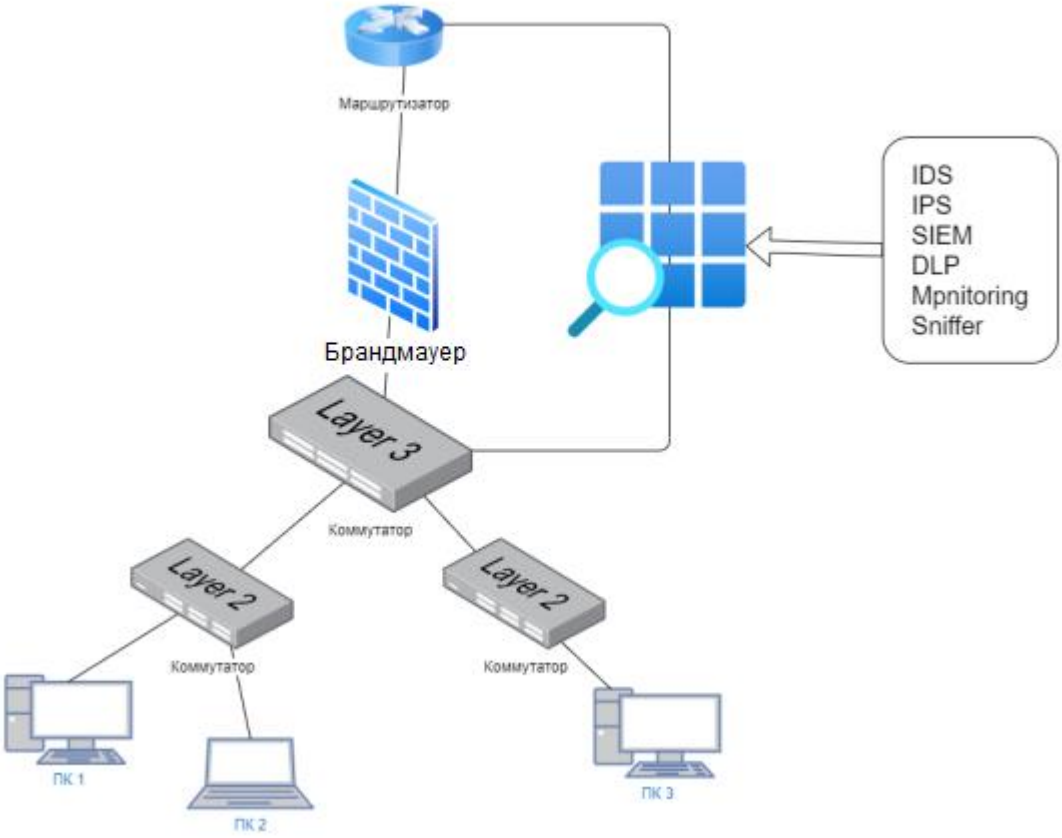

Рисунок 1.2 – Приклад мережі із застосуванням дзеркалювання трафіку

Наприклад, сучасні IDS/IPS (системи виявлення/запобігання вторгнень) працюють з мережевим трафіком, виявляючи в ньому мережеві атаки. Відмінність між IDS та IPS полягає лише в подальших діях, які виконує система після виявлення загрози [3]:

– IDS (Intrusion Detection System) може лише сигналізувати про факт виявлення;

– IPS (Intrusion Prevention System) — розривати з'єднання, в якому було виявлено атаку.

Відповідно, схеми інтеграції IDS та IPS в інфраструктуру організації також виглядають по-різному:

– IDS працює виключно з копією трафіку, що проходить через мережеве обладнання (трафік перенаправляється за технологією SPAN або RSPAN), або віртуальні машини (за технологією ERSPAN);

– IPS же встановлюється безпосередньо на шляху самого трафіку.

Велика кількість керованих мережевих комутаторів дає змогу дублювати трафік від одного або кількох портів чи VLAN на окремо взятий порт. Дзеркалювання портів використовується на мережевому комутаторі або маршрутизаторі для надсилання копії мережевих пакетів, що надходять на вихідний порт, на порт призначення. Коли дзеркалювання портів увімкнене, пакети можна відстежувати та аналізувати.

Дзеркалювання портів широко застосовується, наприклад, мережеві інженери можуть використовувати його для аналізу та відлагодження даних або діагностики помилок у своїх мережах, не впливаючи на можливості обробки пакетів мережевих пристроїв. А Міністерство культури та громадської безпеки може збирати пов'язані дані з дзеркалювання портів для аналізу поведінки мережі, щоб забезпечити здорове мережеве середовище.

Локальне дзеркалювання портів є найпростішою формою дзеркалювання. Усі вихідні порти розташовані на тому ж мережевому пристрої, що й порт призначення. Як показано на рисунку 1.3, дзеркалювання локального порту дозволяє мережевому комутатору переслати копію пакета з вихідного порту (Eth 1/1) на порт призначення (Eth 1/2). Потім пристрій моніторингу, підключений до порту призначення, може відстежувати та аналізувати пакет.

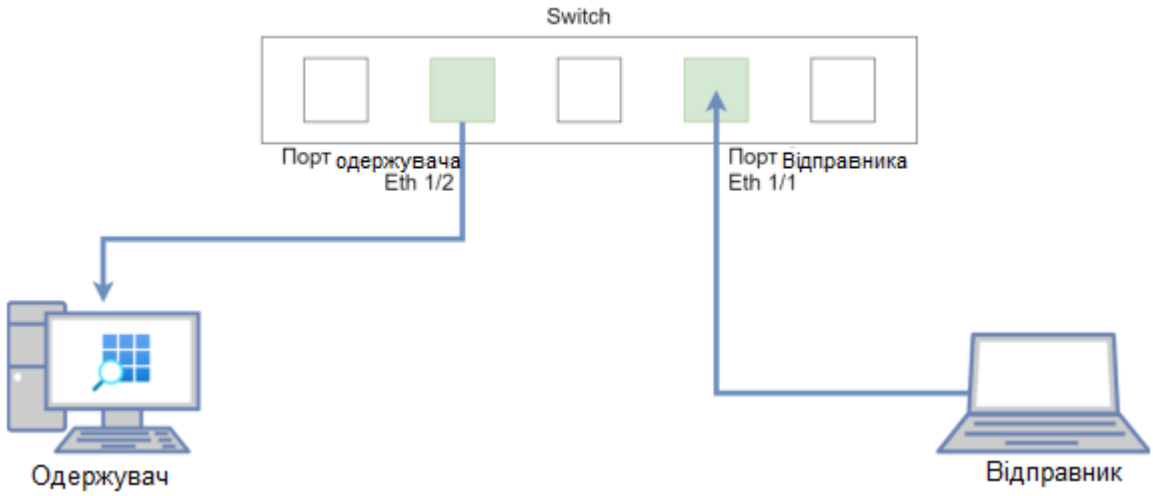

Рисунок 1.3 – Дзеркалювання локального порту

Прикладом такого механізму є SPAN (Switch Port Analyzer), який використовується в обладнанні Cisco Systems. Це ефективна та високопродуктивна система моніторингу трафіку. SPAN застосовується для усунення проблем з підключенням та розрахунку використання й продуктивності мережі. Він дзеркально відображає трафік від одного або кількох інтерфейсів комутатора до одного або кількох інтерфейсів на тому ж комутаторі.

Що стосується дзеркалювання віддалених портів, то це відбувається, коли вихідні порти та порти призначення не знаходяться на одному пристрої. Вихідний порт пересилає копію пакета на порт призначення через з'єднання висхідної лінії зв'язку, як показано на рисунку 1.4. Отже, дзеркалювання віддалених портів дозволяє здійснювати моніторинг та аналіз даних на різних пристроях [4].

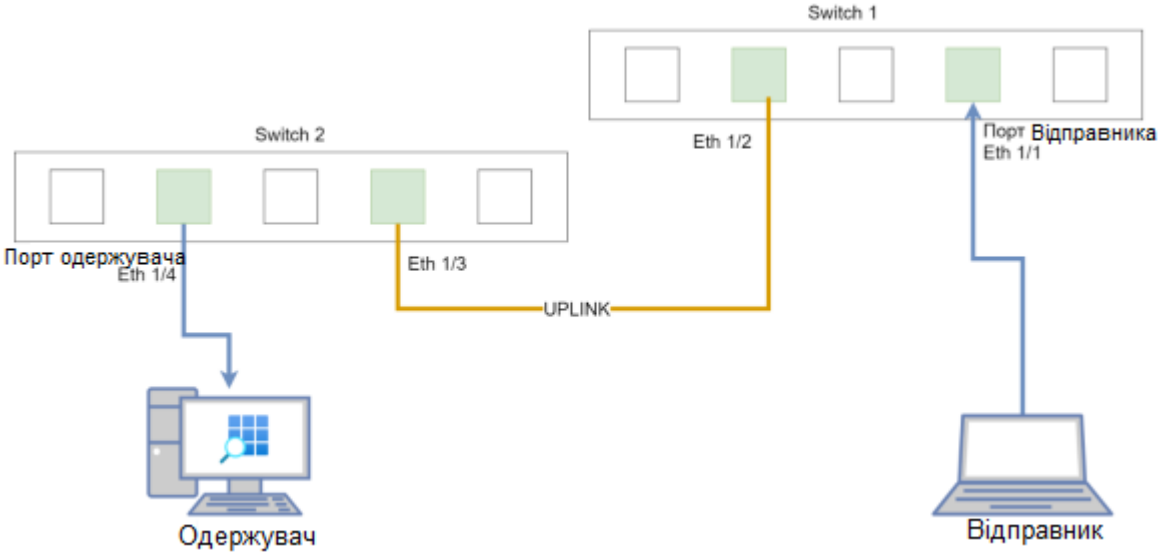

Рисунок 1.4 – Дзеркалювання віддалених портів

Прикладом такого механізму є RSPAN (Remote Switch Port Analyzer), який використовується в обладнанні Cisco Systems (рис.1.5). Він дозволяє здійснювати моніторинг трафіку на фізично віддалених портах через мережу комутаторів. Джерелом трафіку при налаштуванні RSPAN є вихідні порти (source ports), що входять до спеціального RSPAN VLAN, який задається окремо для кожної RSPAN-сесії. Потім фрейми, що передаються в цій сесії, проходять процедуру тегування за

протоколом 802.1q для передачі через інші пристрої, що знаходяться в мережі. Діставшись до комутатора, який містить порт-отримувач для даної RSPAN-сесії, трафік просто дзеркалюється на вказаний порт-отримувач.

Таблиця 1.2 – Комутатори Catalyst, що підтримують функції SPAN та RSPAN

Комутатори Catalyst	**Підтримка SPAN**	**Підтримка RSPAN**
Комутатори серії Catalyst Express 500	Так	Ні
Комутатори серії Catalyst 6500/6000	Так	Так
Комутатори серії Catalyst 5500/5000	Так	Ні
Комутатори серії Catalyst 4900	Так	Так
Комутатори серії Catalyst 4500/4000	Так	Так
Комутатори серії Catalyst 3750 Metro	Так	Так
Комутатори серії Catalyst 3750	Так	Так
Комутатори серії Catalyst 3560	Так	Так
Комутатори серії Catalyst 3550	Так	Так
Комутатори серії Catalyst 3500 XL	Так	Ні
Комутатори серії Catalyst 2970	Так	Так
Комутатори серії Catalyst 2960	Так	Так
Комутатори серії Catalyst 2955	Так	Так
Комутатори серії Catalyst 2950	Так	Так
Комутатори серії Catalyst 2940	Так	Ні
Комутатори серії Catalyst 2948G-L3	Так	Так
Комутатори серії Catalyst 2900XL	Так	Ні
Комутатори серії Catalyst 1900	Так	Ні

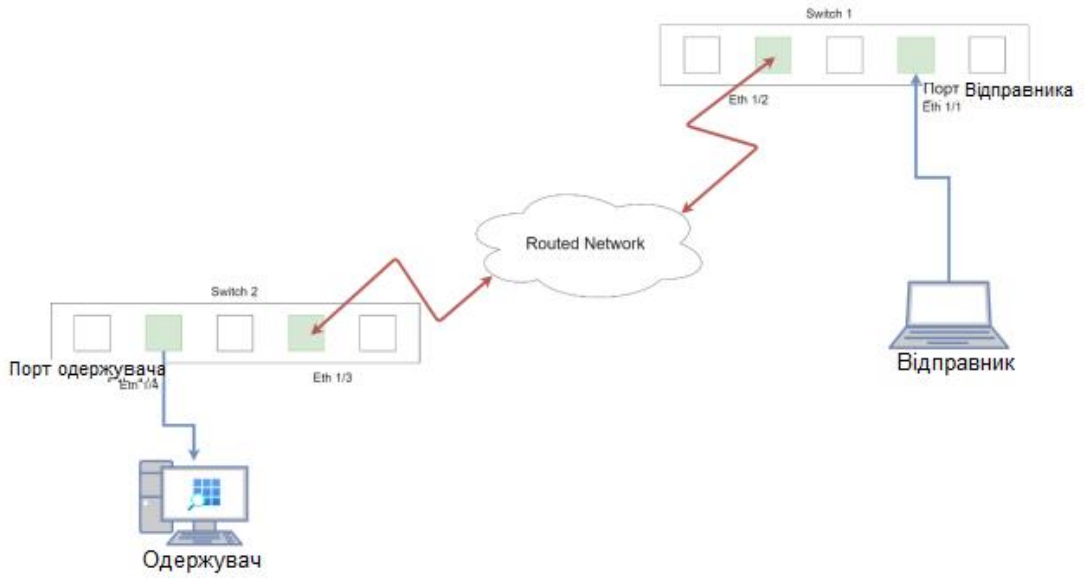

Рисунок 1.5 – Дзеркалювання віддаленого інтерфейсу через маршрутизовану мережу

Останній виявлений метод базується на використанні системних резидентних процесів або демонів. Ці методи застосовуються, наприклад, у SIEM-системі на основі процесу syslog. Ще одним резидентним процесом є rpcapd, створений з використанням бібліотеки libpcap [5].

Syslog – це стандарт відправлення та реєстрації повідомлень про події, що відбуваються в системі, тобто стандарт створення журналів подій, який використовується в комп'ютерних мережах, що працюють за протоколом IP. Терміном "syslog" називають як стандартизований мережевий протокол syslog, так і програмне забезпечення (додаток, бібліотеку), що займається відправленням та отриманням системних повідомлень. При цьому варто зазначити, що журнал подій — це стандартний спосіб запису та централізованого зберігання інформації про важливі програмні та апаратні події для додатків та операційної системи. Наприклад, запис у журналі подій може містити:

– завантаження CPU однієї з робочих станцій користувачів зросло до 100%;
– логування входу в систему Windows;
– підключення Flash-накопичувача до робочої станції користувача;
– логування виходу з системи або часу роботи користувача.

У цій роботі syslog не буде розглядатися, оскільки це розробка для відправлення та отримання системних повідомлень. Процесом взаємодії з мережевим трафіком займається програмний інструмент, що працює в середовищі ОС і дозволяє взаємодіяти з драйверами мережевих інтерфейсів пристрою. Для цього потрібен віддалений демон RPCAPD, який виконує захоплення та надсилає дані назад, і локальний клієнт, який надсилає відповідні команди та отримує захоплені дані.

Програмне забезпечення RPCAPD працює на базі протоколу RPCAP (Remote Packet Capture), який призначений для моніторингу мережевого трафіку та захоплення пакетів, що надходять на віддалений пристрій у мережі, для контролю та аналізу переданих потоків даних. Перехоплення трафіку здійснюється копіюванням переданих пакетів, після чого, застосовуючи протокол передачі RPCAP, вони

передаються на потрібний віддалений хост (рис.1.6).

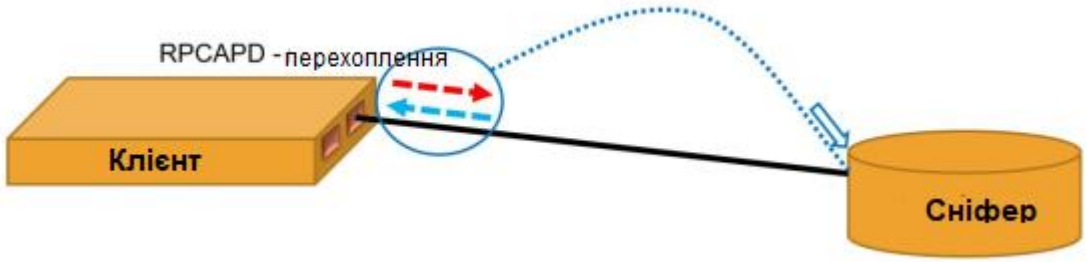

Рисунок 1.6 – Схема перехоплення трафіку з використанням RPCAPD

Протокол RPCAP передбачає взаємодію віддаленого пристрою та програми аналізу мережевих даних (аналізатора пакетів) за схемою клієнт-сервер. На віддаленому пристрої запускається демон RPCAP, який приймає запити на з'єднання від клієнтських програм, проводить автентифікацію та починає обслуговування авторизованих клієнтів: "прослуховує" мережу та передає запитувані пакети клієнту для обробки та аналізу.

1.3 Обґрунтування вибраного напрямку вирішення задачі

З огляду на експоненційне зростання локальних мереж, зростання кількості кібератак та необхідність моніторингу мережевого трафіку для забезпечення безпеки, розробка ефективної методики віддаленого перехоплення є вкрай важливою. Існуючі дослідження мають значні прогалини у цьому аспекті, зокрема, відсутність повноцінних описів методик та оцінки їхньої ефективності.

Дзеркалювання трафіку (SPAN, RSPAN, ERSPAN) є ефективним способом, але часто залежить від апаратних можливостей комутаторів та може бути обмеженим у масштабі. Методи, засновані на системних резидентних процесах, таких як rpcapd, пропонують більшу гнучкість, оскільки вони працюють на рівні операційної системи. Це дозволяє перехоплювати трафік з будь-якого пристрою, на якому можна запустити демон, незалежно від його мережевого обладнання.

Протокол RPCAP спеціально розроблений для віддаленого моніторингу та захоплення пакетів. Це ключова перевага для корпоративних мереж, де

централізований контроль та аналіз трафіку з віддалених точок є необхідністю. Адміністратори можуть отримувати доступ до трафіку з різних робочих станцій, серверів або інших пристроїв, не виходячи з одного місця.

Методика дозволить не тільки перехоплювати трафік, але й здійснювати його фільтрацію та подальший аналіз. Це критично важливо для виявлення аномалій, оцінки безпеки мережі та діагностики проблем без перевантаження системи зайвими даними.

Можливість віддаленого перехоплення трафіку має великий потенціал для підготовки лабораторних робіт та навчання фахівців у галузі мережевої безпеки, зокрема, щодо бездротових мереж.

Діаграми послідовності (рис.1.7) є надзвичайно корисними для візуалізації логіки використання системи, документування сценаріїв взаємодії у складних системах, виявлення потенційних проблем або вузьких місць у потоці повідомлень, проектування API та інтерфейсів взаємодії між модулями, а також для розуміння взаємодії між компонентами мікросервісної архітектури. Вони допомагають розробникам та архітекторам чітко уявити, як об'єкти системи співпрацюють для досягнення певної мети, забезпечуючи краще розуміння динамічної поведінки системи.

Учасники (Actors/Lifelines) — це горизонтальні прямокутники у верхній частині діаграми. Вони представляють окремі екземпляри об'єктів, системи або користувачів, що беруть участь у взаємодії. Від кожного учасника вниз тягнеться вертикальна лінія, яка називається лінією життя (lifeline). Вона показує часову вісь існування та активності учасника. Лінії життя (Lifelines) — це вертикальні пунктирні лінії, що йдуть вниз від кожного учасника. Вони символізують проміжок часу, протягом якого об'єкт існує або бере участь у послідовності.

Повідомлення (Messages) — це горизонтальні стрілки, що з'єднують лінії життя. Вони показують обмін інформацією або виклик операцій між учасниками. Залежно від типу стрілки та її зовнішнього вигляду, повідомлення можуть бути: синхронними (суцільна ліріях заповненою стрілкою, що вказує на блокуючий виклик, тобто відправник чекає відповіді); асинхронними (суцільна лінія з

відкритою стрілкою, що вказує на неблокуючий виклик, тобто відправник не чекає відповіді); повідомленнями-відповідями (Reply messages) (пунктирна лінія з відкритою стрілкою, що показує повернення керування або результату); повідомленнями створення (Create messages) (стрілка, спрямована до об'єкта, що тільки створюється); повідомленнями знищення (Destroy messages) (стрілка, що закінчується символом "X" на лінії життя, вказуючи на знищення об'єкта).

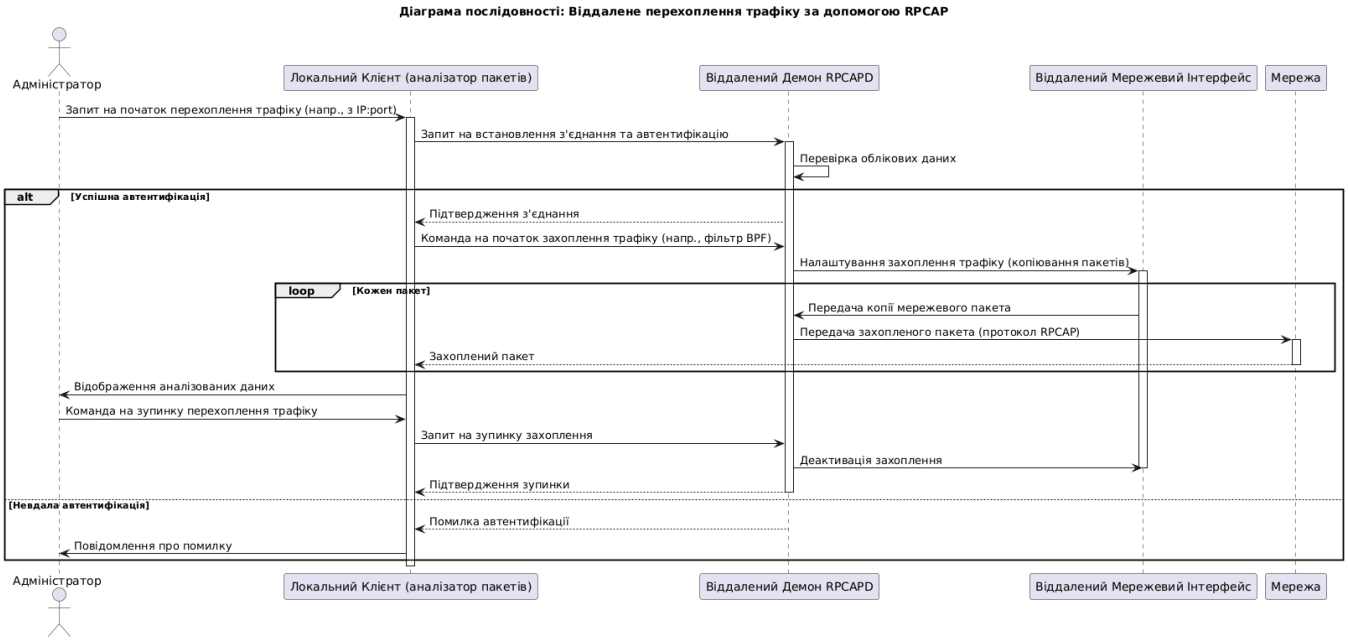

Рисунок 1.7 – Діаграма послідовності

Фрагменти виконання (Activation/Execution Occurrences) — це вертикальні прямокутники, що розміщуються на лініях життя. Вони позначають період часу, протягом якого об'єкт активно виконує операцію або відповідає на повідомлення.

Комбіновані фрагменти (Combined Fragments) — це рамки, що охоплюють частину діаграми та використовуються для моделювання складних логічних структур. До них належать: alt (Alternative) для моделювання умовних розгалужень (як if-else); opt (Option) для необов'язкових послідовностей (як if); loop (Loop) для циклічного виконання послідовностей; par (Parallel) для паралельного виконання декількох послідовностей; ref (Refinement) для посилання на іншу діаграму послідовності.

2 СПЕЦІАЛЬНИЙ РОЗДІЛ

2.1 Дослідження ефективності підходу перехоплення трафіку з застосуванням протоколу RPCAP

Для дослідження віддаленого перехоплення трафіку ми будемо використовувати демон RPCAPD. Наша тестова конфігурація складається з двох робочих станцій, з'єднаних між собою каналом зв'язку. Ми використовуємо пристрій з ОС Windows 10 як фізичну машину, а на ній запускаємо віртуальну машину (ВМ) з ОС Ubuntu 21.04. Обидві машини взаємодіятимуть через віртуальний комутатор, як показано на рис.2.1 [6].

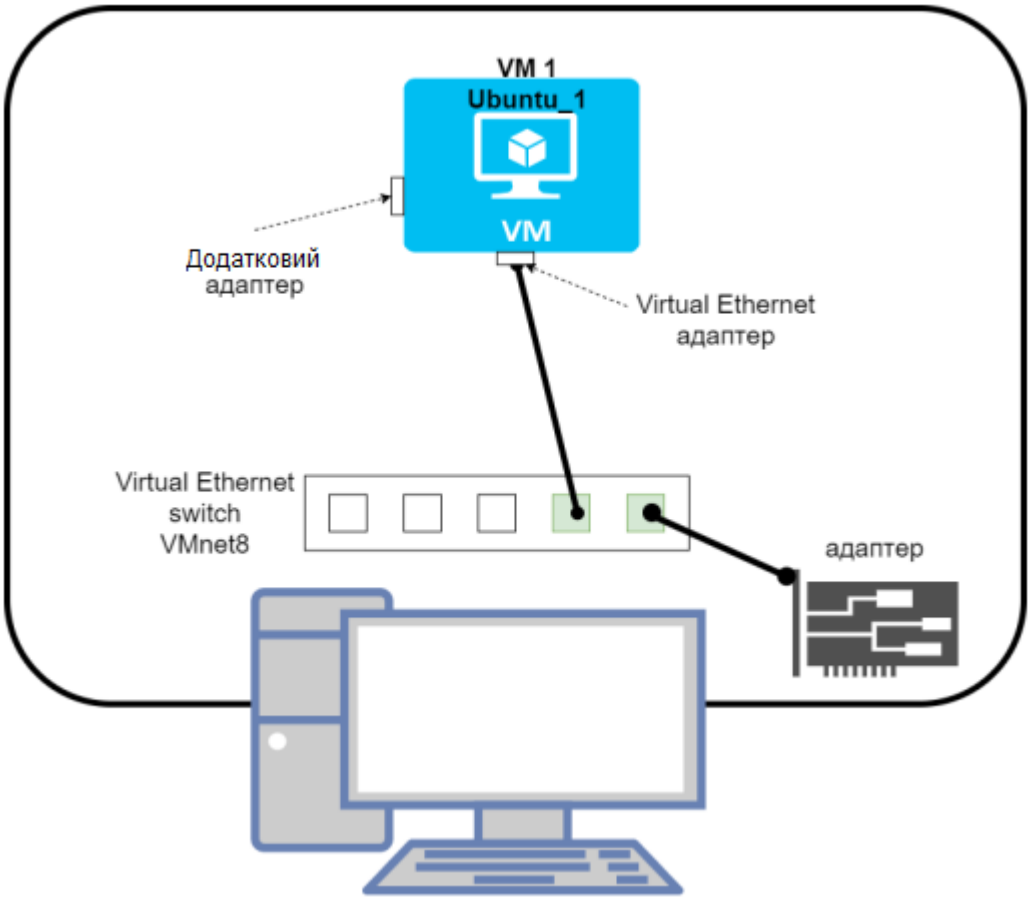

Рисунок 2.1 – Взаємодія компонентів мережі для дослідження RPCAPD

Етапи дослідження резидентного процесу RPCAPD описано нижче.

Налаштування серверної сторони: На фізичній машині (з Windows 10)

потрібно встановити програмне забезпечення для віртуалізації, наприклад, VMware Workstation. Ця машина виконуватиме роль серверної частини системи.

Створення клієнтської сторони: Необхідно створити віртуальну машину з ОС Ubuntu 21.04 (або іншою обраною ОС). В нашому дослідженні ми обрали Ubuntu. Ця ВМ буде виконувати функції клієнтської частини системи.

Встановлення бібліотеки libpcap на Ubuntu: Завантажте та встановіть бібліотеку libpcap на ВМ з Ubuntu. Libpcap є кросплатформною бібліотекою з відкритим вихідним кодом (її версія для Windows називається WinPcap). Відомі сніфери, такі як tcpdump та Wireshark, використовують цю бібліотеку для своєї роботи.

Встановлення демона віддаленого захоплення (rpcapd): Оскільки дослідження проводиться на UNIX-подібній системі, встановлення демона вимагає окремих кроків. Для початку роботи виконайте наступні команди в терміналі Ubuntu:

cd libpcap – перейдіть до каталогу бібліотеки.

./configure --enable-remote – ця команда шукає необхідні бібліотеки та заголовні файли для компіляції (для програм, написаних на C/C++ та подібних мовах), а також налаштовує спеціальні параметри. Якщо все знайдено, вона створить Makefiles – файли, необхідні для збірки програми.

make – це найважливіша команда, яка запускає процедуру компіляції додатка з вихідного коду. Вона використовує файли Makefiles, де детально описано процес збірки. Результатом успішного виконання команди буде зібрана програма в поточній директорії.

make install – ця команда безпосередньо встановлює додаток у директорію, вказану на етапі конфігурування.

Додавання додаткового мережевого адаптера на клієнті: На клієнті (ВМ з Ubuntu) додайте додатковий мережевий адаптер. У нашому випадку це буде адаптер бездротової мережі, через який буде передаватися трафік (див. Рисунок 7).

Запуск демона RPCAPD на ВМ: На ВМ запустіть демон RPCAPD командою: sudo rpcapd -n.

Запуск сніфера на серверній стороні: На стороні сервера (на Windows)

запустіть сніфер Wireshark [7]. Почніть сеанс перехоплення із запитом на отримання даних з додаткового адаптера через віртуальний адаптер (див. Рисунок 2.2). Також запустіть сеанс перехоплення трафіку на фізичному адаптері локальної машини. Це необхідно для того, щоб вивчити обмін інформацією та встановлення з'єднання між клієнтом і сервером.

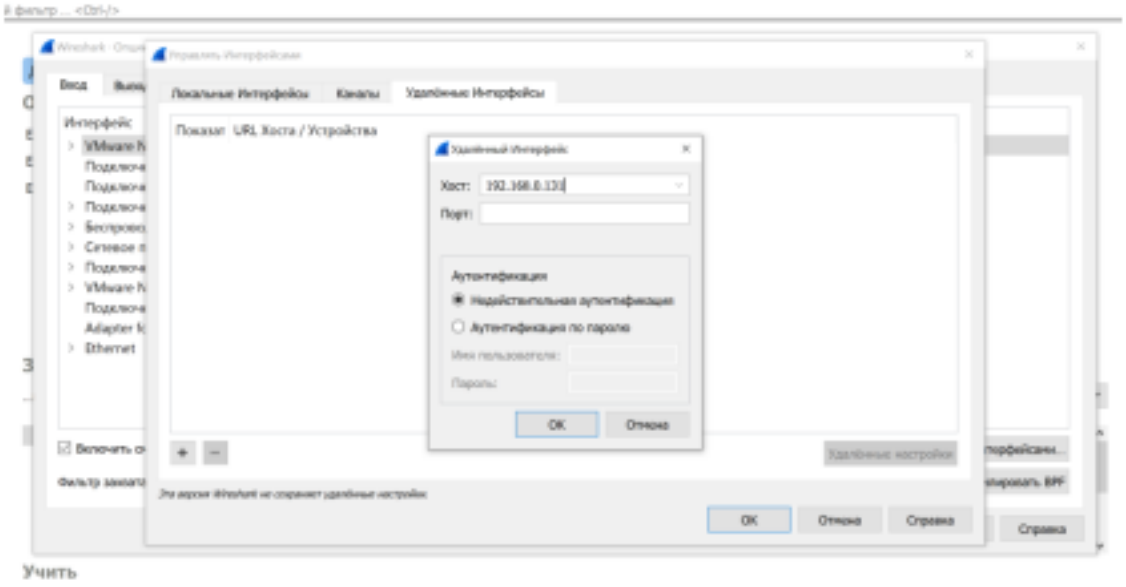

Рисунок 2.2 – Взаємодія компонентів мережі для дослідження RPCAPD

У підсумку, ми отримаємо можливість онлайн-перехоплення трафіку на серверній стороні безпосередньо з віддаленого інтерфейсу віртуальної машини (ВМ).

Варто зазначити, що під час запуску демона RPCAPD на клієнтській стороні можна використовувати різні прапори (опції), представлені в Таблиці 3. Їх можна комбінувати у форматі: rpcapd [-b <адреса>] [-p <порт>] [-6] [-l <список-хостів>] [-a <хост, порт>] [-n] [-v] [-d] [-s <файл>] [-f <файл>].

Наприклад, робочий рядок може виглядати так: rpcapd -n -p 5002. У цьому прикладі демон налаштовується для підключення без автентифікації за допомогою прапора -n та використання порту 5002 за допомогою прапора -p.

Таблиця 2.1 – Опис прапорів команди для запуску демона RPCAPD

Прапор	Опис
-b <адреса>	Встановлює адресу (числову або літерну), до якої демон має прив'язатися. За замовчуванням: прив'язується до всіх локальних адрес IPv4 та IPv6.
-p <порт>	Встановлює порт, до якого демон має бути прив'язаний. За замовчуванням: прив'язується до порту 2002.
-4	Прив'язується тільки до IPv4-адрес. За замовчуванням: використовуються очікуючі сокети IPv4 та IPv6.
-l <файл-списку-хостів>	Визначає файл, в якому зберігається список хостів, яким дозволено підключатися до цього демона (якщо їх більше одного, файл зберігає їх по одному в рядку). Рекомендується використовувати літерні імена (замість числових), щоб уникнути проблем з різними сімействами адрес (IPv4 та IPv6).
-n	Дозволяє NULL-автентифікацію (зазвичай використовується з -l, що гарантує, що лише дозволені хости можуть підключатися до демона). За замовчуванням: вимагається механізм автентифікації за іменем користувача та паролем.
-a <хост, порт>	Змушує демон працювати в активному режимі та підключатися до <хосту> через порт <порт>. Це не виключає того, що демон все ще може приймати пасивні з'єднання.
-v	Змушує демон працювати тільки в активному режимі. За замовчуванням: демон завжди приймає активні з'єднання, навіть якщо вказано ключ -a.
-d	Змушує демон працювати у фоновому режимі, тобто як демон (тільки UNIX) або як служба (тільки Win32). Попередження (Win32): цей перемикач надається автоматично, коли WinPcap встановлює цей демон у служби Win32 (панель керування - інструменти адміністрування - служби).
-s <файл>`	Зберігає поточну кон
-f <файл>	Завантажує поточну конфігурацію з файлу; всі перемикачі, вказані в командному рядку, ігноруються, і замість них використовуються налаштування файлу.
-h	Виводить екран довідки.

2.1.1 Формування загальних вимог до системи

У процесі встановлення з'єднання RPCAPD між сервером і клієнтом відбувається обмін пакетами, який називається handshake (рукостискання). Цей процес дозволяє налаштувати зв'язок для подальшого віддаленого перехоплення трафіку. Приклад побудови RPCAPD-сесії показано на Рисунку 2.3 [8].

```
 82 Authentication request
 76 Authentication reply
 89 Open request
 82 Open reply
102 Start capture request
 82 Start capture reply
```

Рисунок 2.3 – Приклад RPCAPD сесії

Модель Handshake. Встановлення TCP-з'єднання між клієнтом і сервером (Рисунок 2.4). Це початковий етап, на якому формується базовий канал зв'язку.

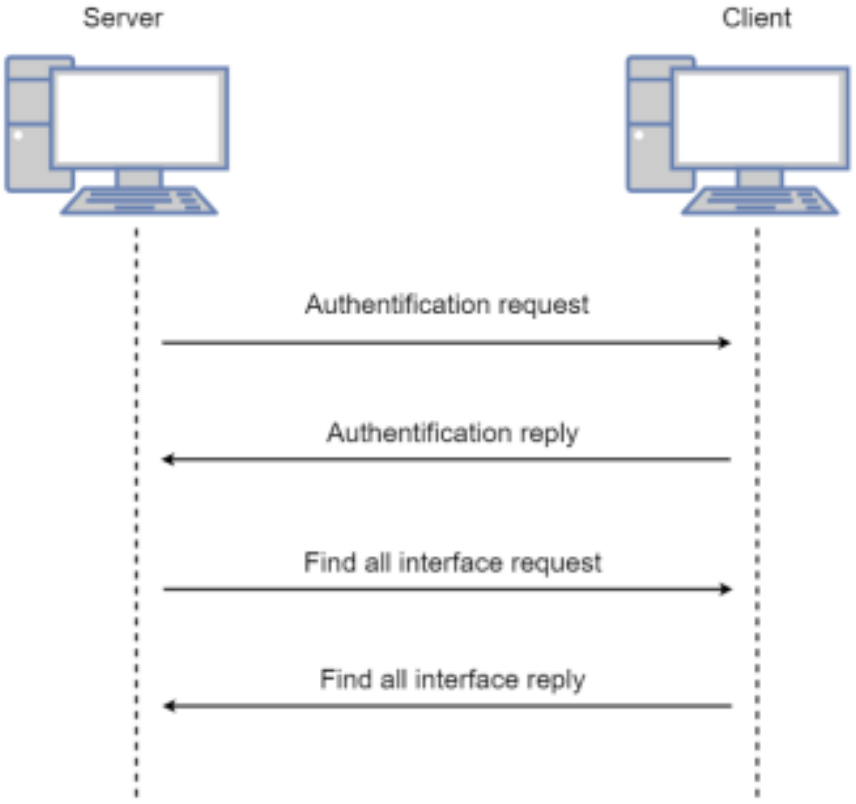

Рисунок 2.4 – Встановлення з'єднання між сервером і клієнтом

Повторне встановлення з'єднань для кожного інтерфейсу окремо (перевірка доступності) між клієнтом і сервером (Рисунок 2.5). Це дозволяє переконатися, що віддалений демон може працювати з усіма необхідними мережевими інтерфейсами на клієнтській стороні.

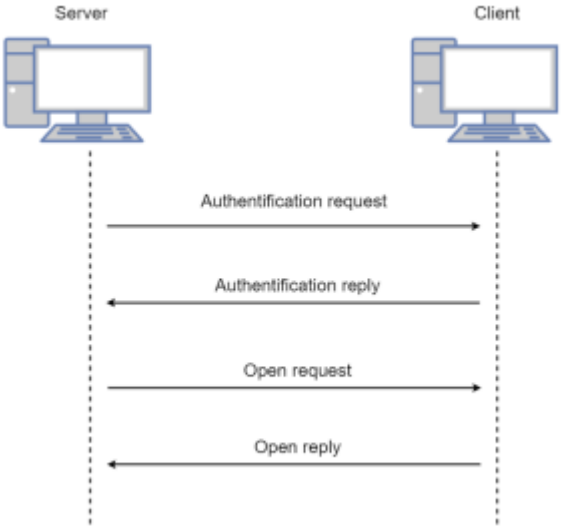

Рисунок 2.5 – Встановлення повторного з'єднання на окремий інтерфейс

Відкриття сесії сервером на вибраному інтерфейсі клієнта (Рисунок 2.6). Після перевірки доступності інтерфейсів сервер ініціює сесію захоплення трафіку на конкретному віддаленому інтерфейсі.

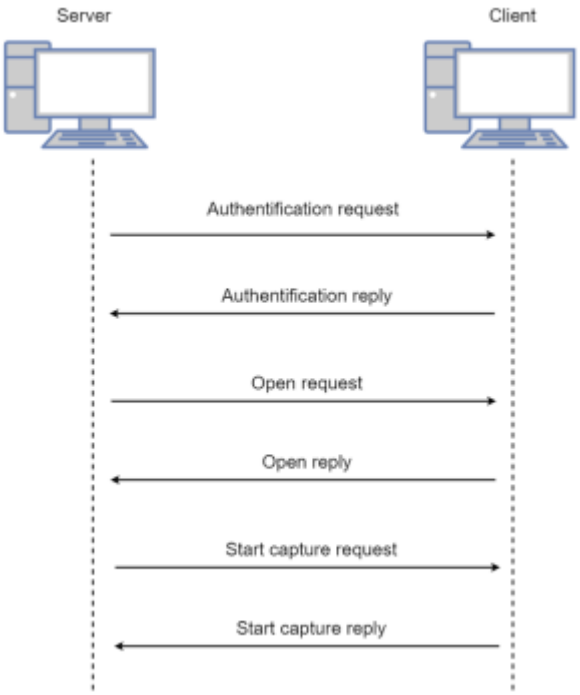

Рисунок 2.6 – Відкриття сесії

Завершення з'єднання з боку клієнта (Рисунок 2.7). Це відбувається після завершення процесу перехоплення трафіку або у випадку помилки.

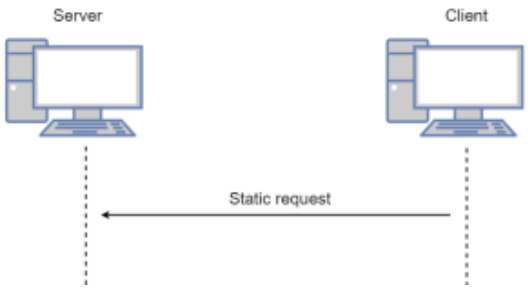

Рисунок 2.6 – Закриття сесії

Authentication request (запит на автентифікацію): Це перший запит від сервера. Рисунок 2.7 показує приклад пакета, коли автентифікація на клієнтській стороні вимкнена. Натомість, Рисунок 2.8 ілюструє пакет, що надсилається, коли на клієнтській стороні увімкнена автентифікація.

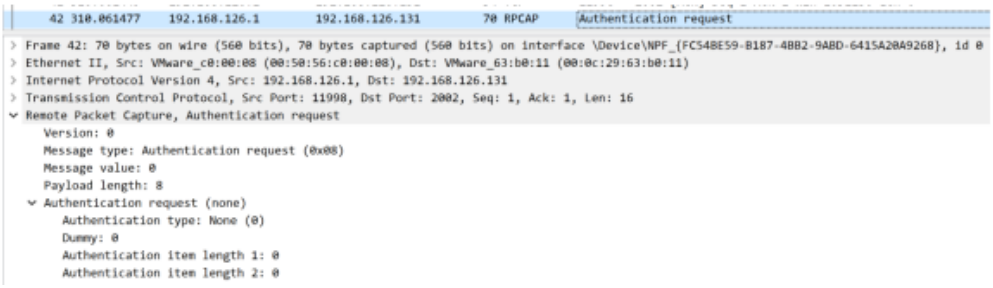

Рисунок 2.7 – Приклад пакетів за відсутності автентифікації на стороні клієнта

Як видно з прикладів, поле заголовка Authentication type змінює значення: 0 означає відсутність автентифікації, а 1 — використання автентифікації. Цей результат також демонструє, що заголовок RPCAP ніяк не шифрує свої дані, а логін і пароль від клієнта можна отримати при перехопленні сесії.

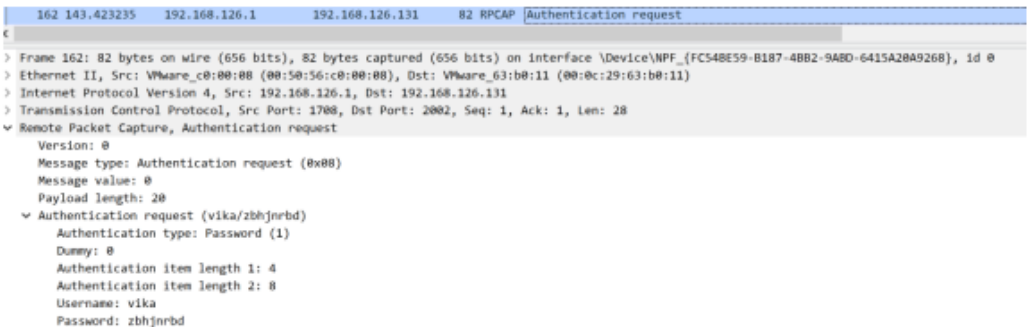

Рисунок 2.8 – Приклад пакета при виборі методу з аутентифікацією на стороні клієнта

Authentication reply (відповідь на запит сервера): Це відповідь клієнта на запит

сервера (Рисунок 2.9). У випадку, коли клієнт використовує автентифікацію, але сервер при реєстрації сесії передає невірний логін або пароль, клієнт відправляє відповідь про помилку автентифікації (Рисунок 2.10).

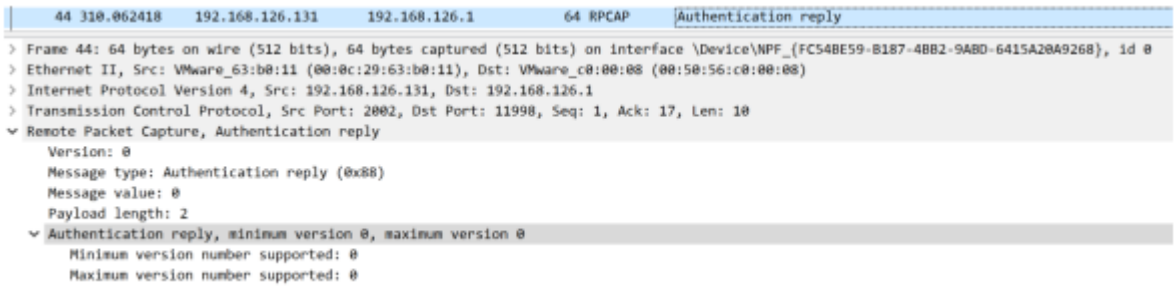

Рисунок 2.9 – Відповідь клієнта на запит сервера

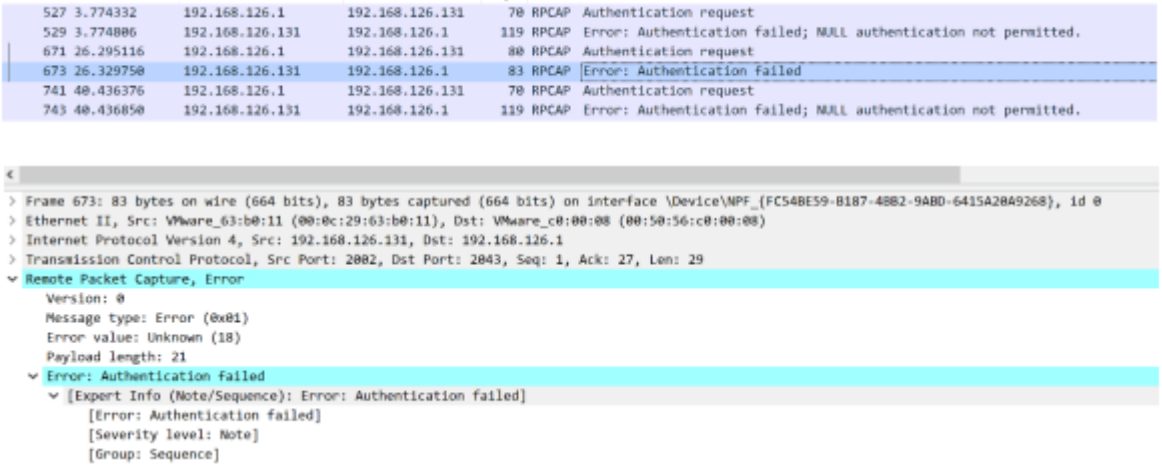

Рисунок 2.10 – Помилка автентифікації

Find all interfaces request (запит на отримання списку всіх інтерфейсів): Сервер надсилає цей запит, щоб отримати повний список доступних мережевих інтерфейсів на клієнтській стороні (Рисунок 2.11).

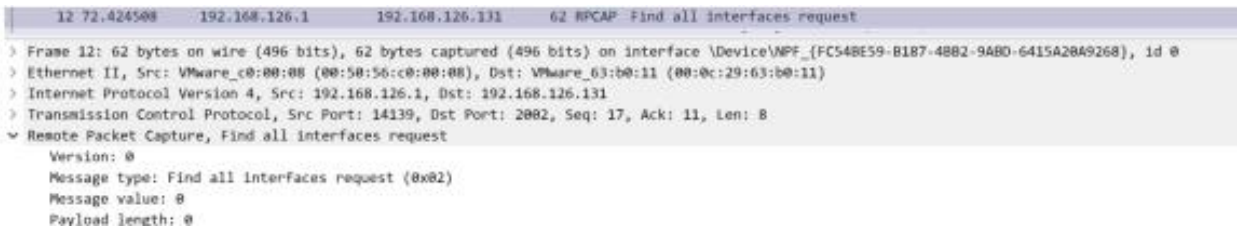

Рисунок 2.11 – Запит на отримання списку доступних інтерфейсів

Find all interface reply (відповідь з інформацією про інтерфейси) – це відповідь

клієнта, що містить інформацію про всі наявні мережеві інтерфейси (Рисунок 2.12).

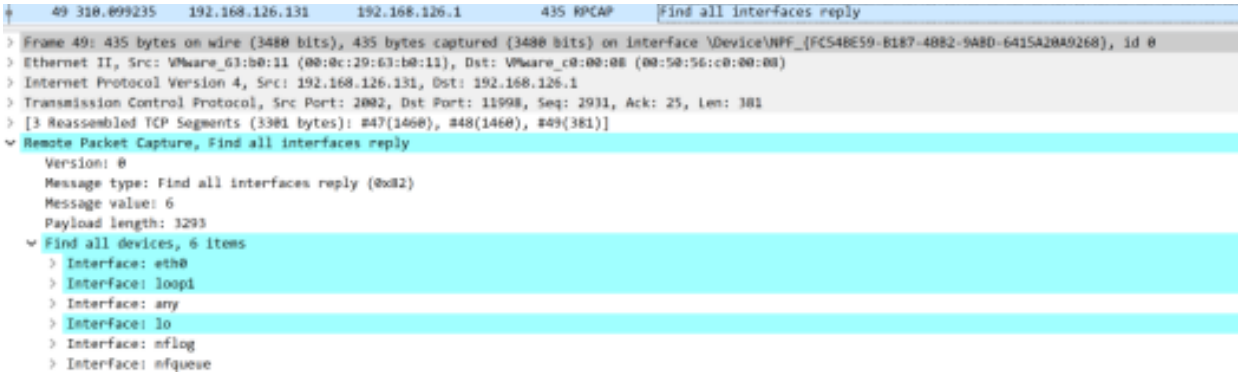

Рисунок 2.12 – Відповідь на запит щодо отримання всіх доступних інтерфейсів

Open request (запит на відкриття сесії) після перевірки доступності інтерфейсів сервер надсилає цей запит для початку перехоплення трафіку на конкретному вибраному інтерфейсі (Рисунок 2.13). Це фактично ініціює створення та налаштування сесії захоплення.

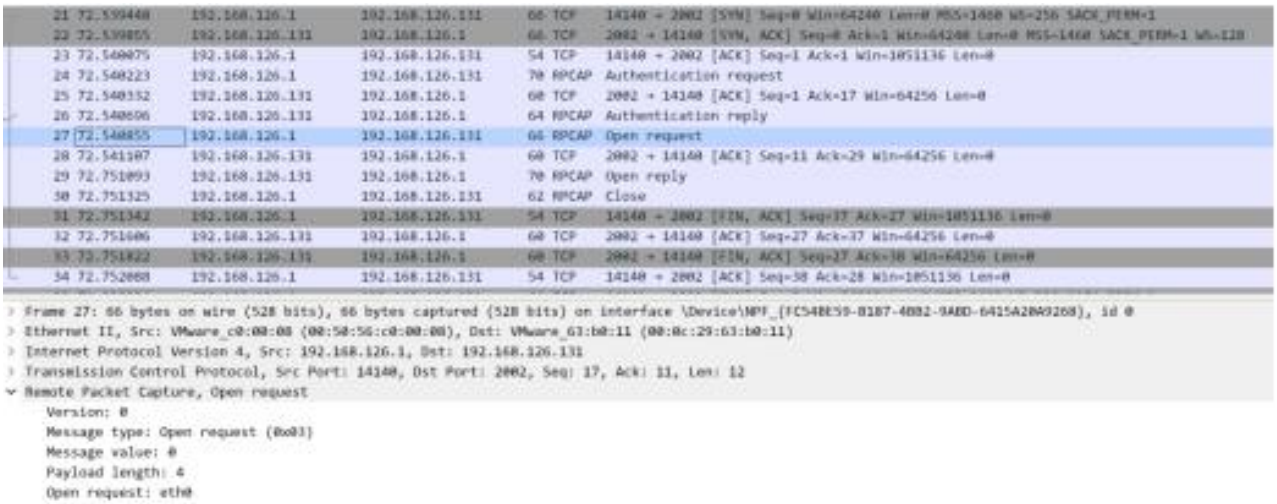

Рисунок 2.13 – Запит на відкриття сесії перехоплення трафіку на конкретному інтерфейсі

Start capture request (запит на початок захоплення). Після відкриття сесії сервер відправляє запит з налаштуваннями захоплення трафіку на віддаленого клієнта (Рисунок 2.14). Цей запит містить правила фільтрації або інші параметри для перехоплення.

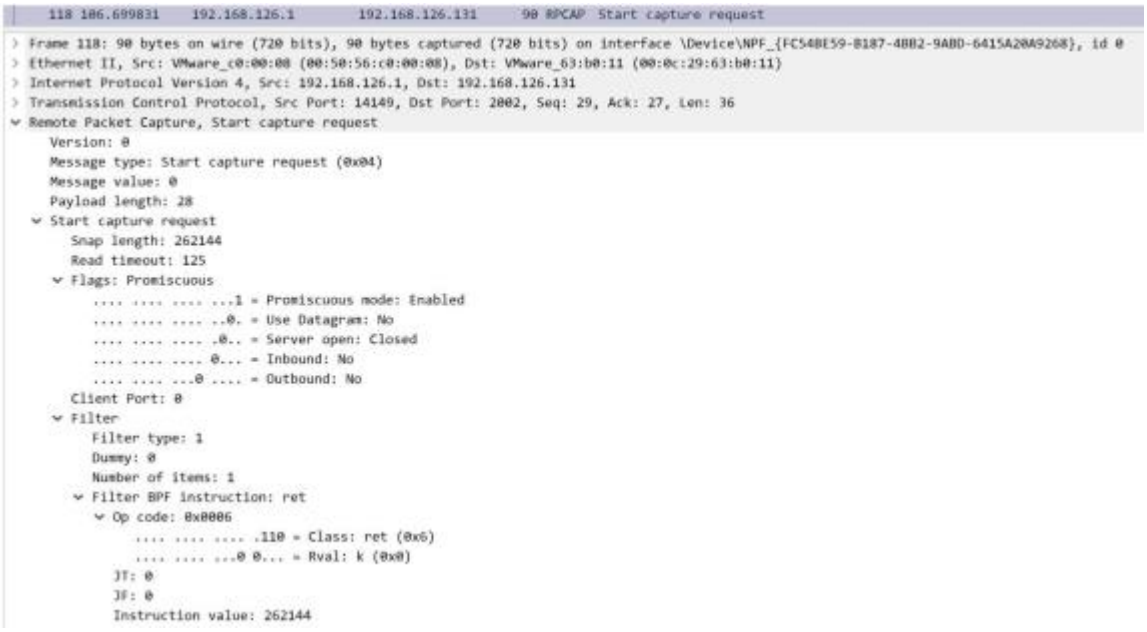

Рисунок 2.14 – Запит з правилами з'єднання

Start capture reply (відповідь на запит налаштувань захоплення). Клієнт надсилає відповідь на запит з налаштуваннями підключення (Рисунок 2.15). У цій відповіді міститься інформація, необхідна для створення нової TCP-сесії, яка буде використовуватися для подальшої передачі захоплених даних.

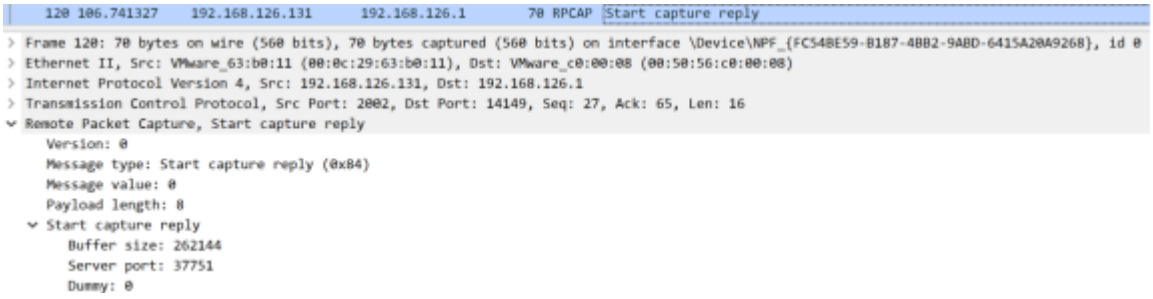

Рисунок 2.15 – Відповідь з правилами з'єднання

Рисунок 2.16 показує приклад ARP-запиту, перехопленого з віддаленого інтерфейсу.

Під час передачі по каналу зв'язку, пакет з віддаленої машини інкапсулюється в заголовок протоколу RPCAP. Після цього поверх накладається заголовок транспортного рівня, наприклад, протоколу TCP, а потім заголовки мережевого, канального та фізичного рівнів. Приклад інкапсуляції даних зображено на Рисунку 2.17.

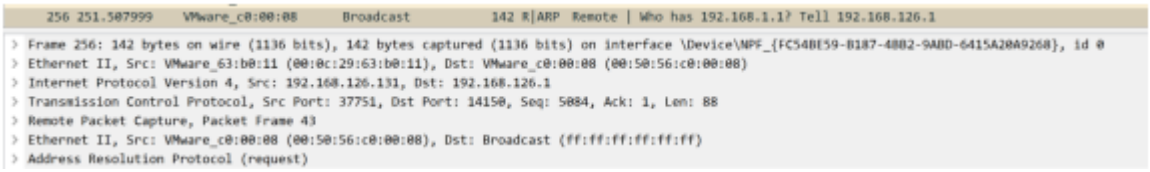

Рисунок 2.16 – Приклад перехопленого з віддаленого інтерфейсу ARP-запиту

Заголовок Remote Packet Capture (RPCAP) може важити максимум близько 28 байт і складається з таких основних частин [9]:

– версія протоколу – 8 біт;
– тип повідомлення – 8 біт;
– значення повідомлення – 16 біт (майже не використовується);
– корисне навантаження – 32 біти.

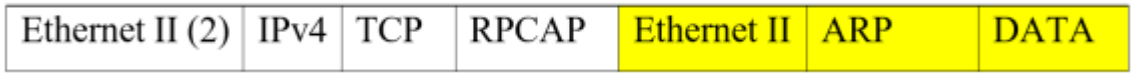

Рисунок 2.17 – Інкапсуляція протоколу RPCAP

2.2 Оцінка ефективності використання системного процесу RPCAP

Для оцінки ефективності використання системного резидентного процесу RPCAPD ми проведемо дослідження на різних операційних системах, включаючи Windows 10, Windows 7, Linux Ubuntu 21.04 та Raspberry OS. У дослідженні буде задіяно сніфер Wireshark та, власне, демон RPCAPD. Крім того, для моніторингу ресурсів системи ми використовуватимемо утиліту top.

Утиліта top та її показники.

Top — це консольна команда, яка відображає список активних процесів у системі та надає детальну інформацію про них. Завдяки цій програмі можна відстежити вплив роботи процесу на CPU пристрою та його ОЗП (оперативну пам'ять).

Результат команди top виводиться у вигляді таблиці з переліком запущених процесів. Перші два стовпці показують номер процесу (PID) та ім'я користувача

(USER), який його запустив. Наступні два стовпці відображають поточний пріоритет процесу (PR) та пріоритет, призначений йому командою NICE (NI). Інші стовпці характеризують безпосередньо рівень споживання ресурсів і розшифровуються так:

– VIRT — віртуальна пам'ять, яку використовує процес;

– RES — фізична пам'ять, зайнята даним процесом;

– SHR — загальний обсяг пам'яті, яку цей процес ділить з іншими;

– S — поточний статус процесу: R — running (виконується); S — sleeping (спить); Z — zombie (зомбі);

– %CPU — відсоток використовуваного часу центрального процесора;

– %MEM — відсоток ОЗП, використовуваної процесом;

– TIME+ — тривалість роботи процесу з моменту запуску;

– COMMAND — назва команди (програми), яка ініціювала процес.

Дослідження проводилися на різних системах. Схема мережі першої експериментальної установки зображена на Рисунку 2.18. На ній показана фізична машина — ПК 1. На ПК 1 використовується програмне забезпечення для віртуалізації, і на ній працюють дві віртуальні машини (ВМ) з встановленою ОС Ubuntu.

ПК 1 та ПК 2 взаємодіють між собою через бездротову мережу за допомогою маршрутизатора, який має вихід до Інтернету. При цьому ВМ 1 та ВМ 2 підключені між собою за методом Host-Only. У цьому режимі створюється віртуальний мережевий адаптер, до якого можна підключити декілька віртуальних машин, об'єднавши їх таким чином у локальну мережу. Доступу до Інтернету немає, проте машини знаходяться в одній мережі, кожна має свою IP-адресу і може взаємодіяти між собою. Доступу до цієї віртуальної мережі з основної системи немає.

На ВМ 1 налаштовано два мережеві адаптери. Один з них описаний вище (Host-Only). Другий підключений до фізичного бездротового адаптера ПК 1 за методом Bridge (міст). За такого підключення віртуальна машина стає повноцінним членом локальної мережі, до якої підключена основна система. ВМ використовує мережевий інтерфейс, щоб отримати адресу від роутера, і стає доступною для інших

пристроїв у мережі, як і основний комп'ютер, за своєю IP-адресою.

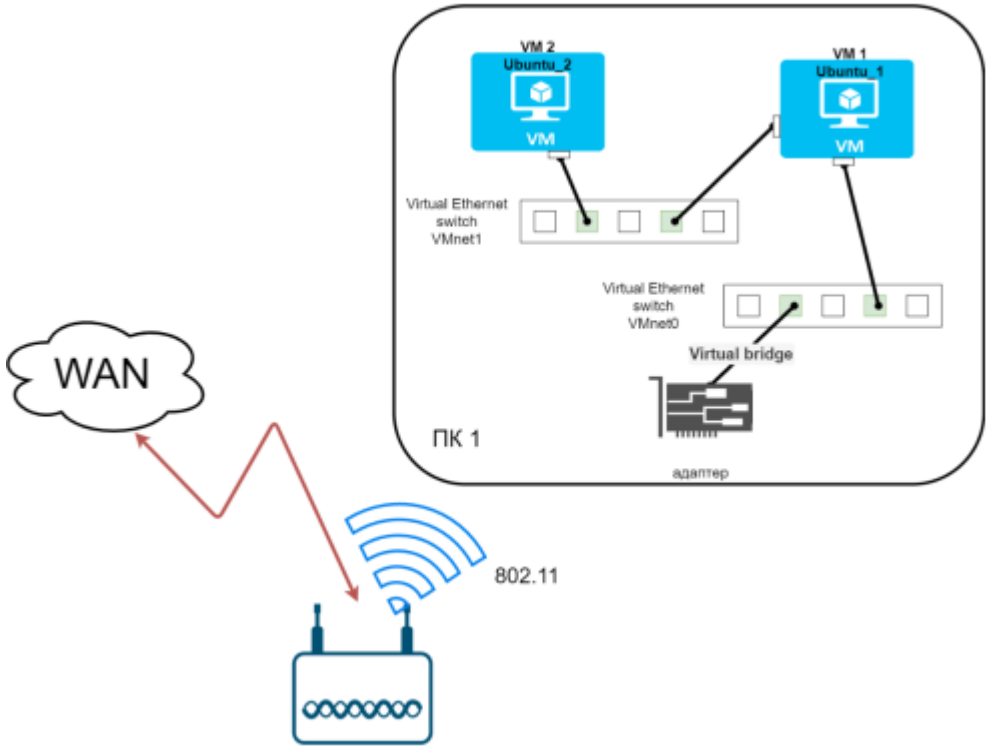

Рисунок 2.18 – Схема мережі першої експериментальної установки

На Рисунку 2.19 зображено другу конфігурацію для наших експериментів. Вона складається з трьох ключових компонентів:

– персональний комп'ютер (ПК) Працює під керуванням операційної системи Windows 10;

– одноплатний комп'ютер Raspberry Pi це малопотужний пристрій, який використовується як віддалений клієнт;

– маршрутизатор бездротової локальної мережі: Забезпечує зв'язок між пристроями та доступ до Інтернету.

Взаємодія компонентів:

– на Raspberry Pi встановлено додатковий фізичний бездротовий адаптер, через який пристрій підключається до маршрутизатора. Це дозволяє Raspberry Pi брати участь у бездротовій мережі;

– крім того, Raspberry Pi підключений до ПК за допомогою дротового

з'єднання. Це створює прямий, стабільний зв'язок між цими двома пристроями;

– ПК також використовує бездротовий канал зв'язку для обміну даними з маршрутизатором та доступу до мережі Інтернет.

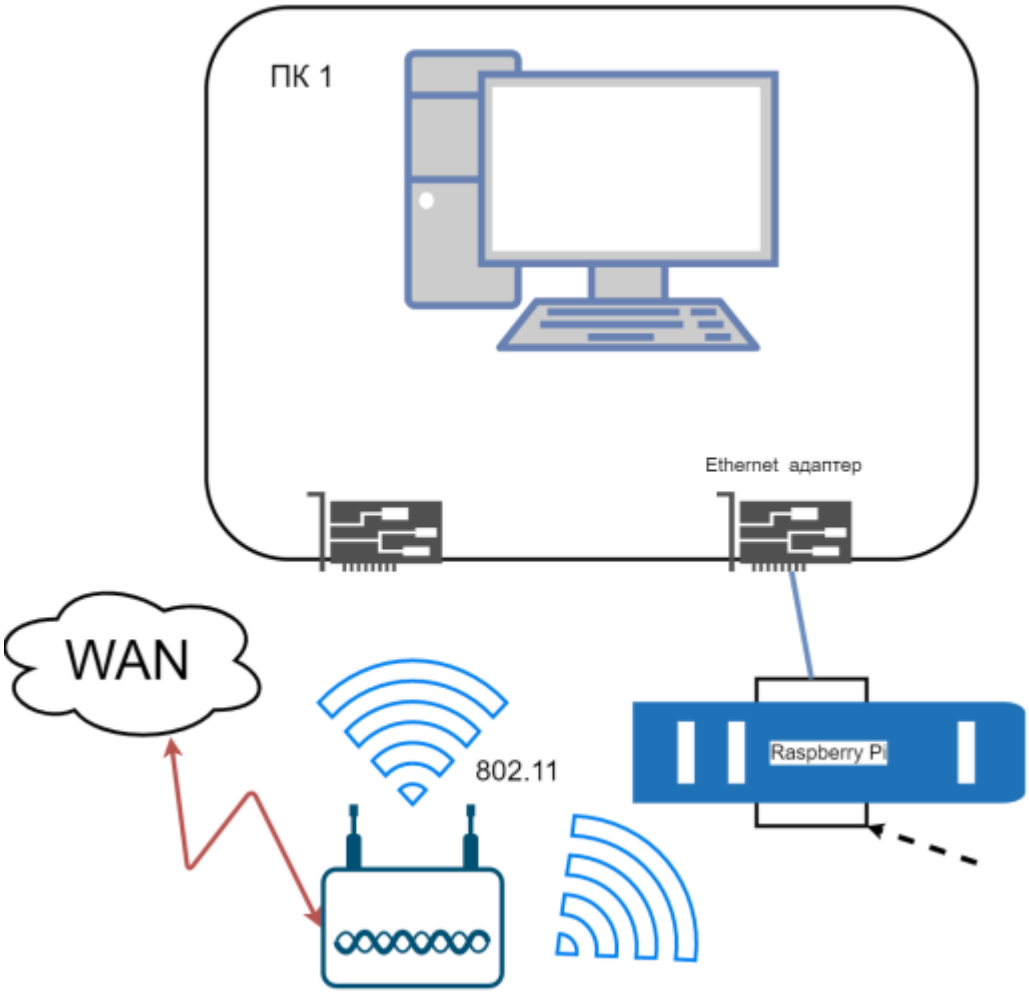

Рисунок 2.19 – Схема мережі другої експериментальної установки

2.2.1 Аналіз трафіку Raspberry Pi під час відтворення відео

Для цього дослідження ми використовуємо другу експериментальну установку, зображену на Рисунку 2.19. Наша мета – проаналізувати процес перехоплення трафіку, коли малопотужний пристрій Raspberry Pi передає захоплені дані на ПК під час перегляду відео низької якості.

Налаштування та сценарій дослідження:

– на ПК встановлюємо сніфер Wireshark. Він виступатиме в ролі сервера

для збору та аналізу трафіку;

– на Raspberry Pi запускаємо демон RPCAPD. Він виконуватиме роль клієнта, що захоплює трафік;

– віддалене перехоплення трафіку здійснюватиметься на бездротовому адаптері Raspberry Pi. Захоплені пакети будуть передаватися на ПК дротовим каналом зв'язку;

– ПК надсилатиме запит до маршрутизатора, який має доступ до Інтернету, з вимогою перегляду відео низької якості на сервісі YouTube.

Відповідно, ПК отримуватиме пакети, що містять дані для відтворення відео, тим самим шляхом. Таким чином, бездротовий канал зв'язку буде активно завантажений в онлайн-режимі.

Перед початком перехоплення трафіку, бездротовий адаптер на Raspberry Pi (зображений на Рисунку 2.20) необхідно перевести в режим «моніторингу» (monitoring mode). Цей режим захоплення даних дозволяє використовувати Wi-Fi адаптер у «нерозбірливому» режимі (promiscuous mode). Це означає, що адаптер може перехоплювати будь-які типи Wi-Fi пакетів: Management (включно з Beacon-пакетами), Data та Control.

```
10110 root      20   0   14688   3872   3688 S   1.0   0.9   0:00.03 rpcapd
10110 root      20   0   14688   3872   3688 S   0.6   0.9   0:00.05 rpcapd
10110 root      20   0   14688   3872   3688 S   4.5   0.9   0:00.19 rpcapd
10110 root      20   0   14688   3872   3688 S   0.6   0.9   0:00.21 rpcapd
10110 root      20   0   14688   3872   3688 S   6.7   0.9   0:00.42 rpcapd
10110 root      20   0   14688   3872   3688 S   0.6   0.9   0:00.44 rpcapd
10110 root      20   0   14688   3872   3688 S   0.6   0.9   0:00.46 rpcapd
```

Рисунок 2.20 – Результат команди top

Отже, перехоплюючи пакети з інтерфейсу бездротового адаптера на Raspberry Pi, ми фактично передаємо дані сканування радіоефіру на ПК.

При запуску утиліти top на Raspberry Pi ми отримуємо табличний вивід на екран (представлений на Рисунку 28), який показує навантаження на центральний процесор (CPU) малопотужного пристрою в момент роботи процесу RPCAPD.

Обробка цих результатів дозволить нам побудувати графік зміни відсотка навантаження на CPU відносно часу роботи процесу RPCAPD та його впливу на

мережеве навантаження (Рисунок 2.21). Це дасть змогу оцінити продуктивність Raspberry Pi під час виконання завдання віддаленого перехоплення трафіку.

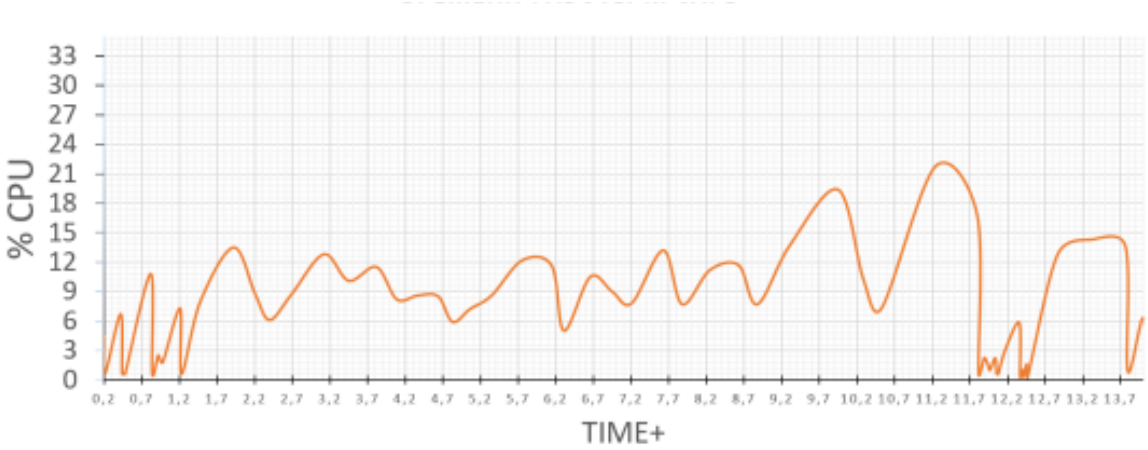

Рисунок 2.21 – Зміна показника відсотка навантаження CPU при відтворенні низької якості зображення

На графіку рис.2.21 чітко видно, як змінювалося навантаження на процесор відповідно до обсягу переданих даних та часу роботи процесу. Під час пікових навантажень бездротова мережа між домашнім маршрутизатором і ПК активно використовувалася для передачі даних, необхідних для відтворення відео.

Протягом тесту середня швидкість прийому пакетів бездротовою мережею становила приблизно 450 Кбіт/с. Водночас, передача даних через дротовий канал зв'язку Ethernet 4 різко зросла під час запуску відео (Рисунок 2.22). На цьому ж рисунку добре простежується залежність між швидкістю передачі даних та зміною відсотка завантаження CPU відносно часу роботи процесу RPCAPD. Піки швидкості передачі даних корелюють з піками навантаження на мережу, а обриси графіків залишаються ідентичними.

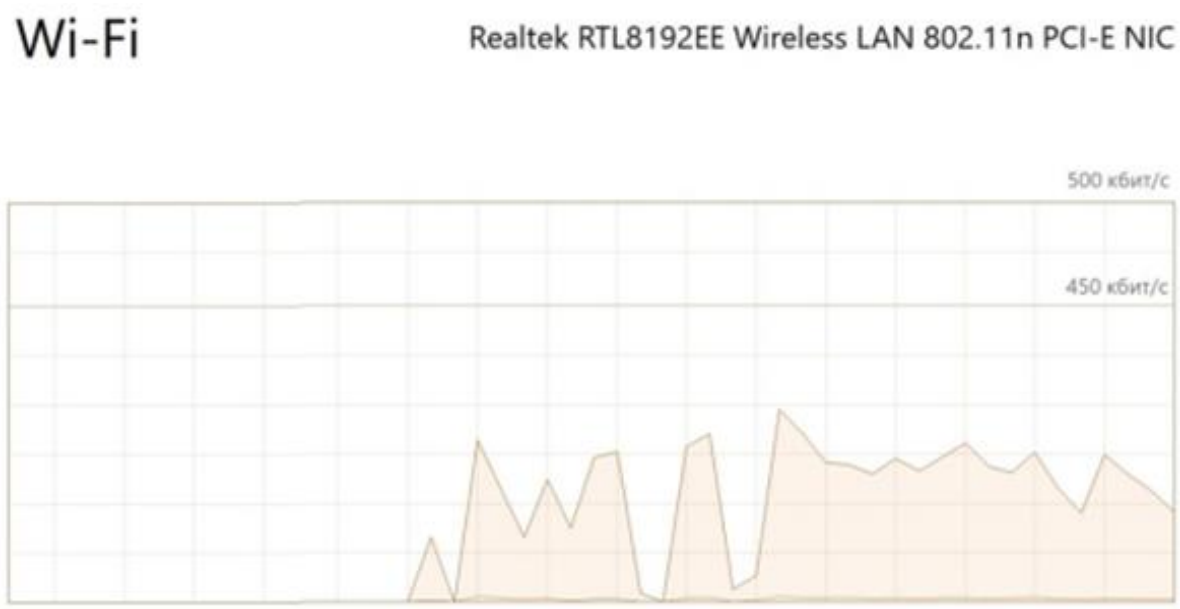

Рисунок 2.22 – Швидкість передачі даних під час відтворення відео низької якості

Виконаємо експеримент другий цього разу для його проведення відтворювалося відео середньої якості (720p).

Передача даних між ПК та маршрутизатором відбувалася за допомогою бездротового каналу зв'язку. На Raspberry Pi було налаштовано інтерфейс бездротової мережі в режимі «Monitoring» (моніторингу). Цей інтерфейс "прослуховував" радіоефір на 13 каналі та перехоплював дані, що передавалися між ПК і маршрутизатором.

Зв'язок між ПК і Raspberry Pi здійснювався за допомогою дротового каналу зв'язку, який використовувався для передачі перехопленого трафіку з бездротового інтерфейсу Raspberry Pi на ПК.

На Рисунку 2.23 зображено графік, що показує, як змінювалися показники відсотка завантаженості CPU відносно часу роботи процесу RPCAPD під час відтворення відео середньої якості.

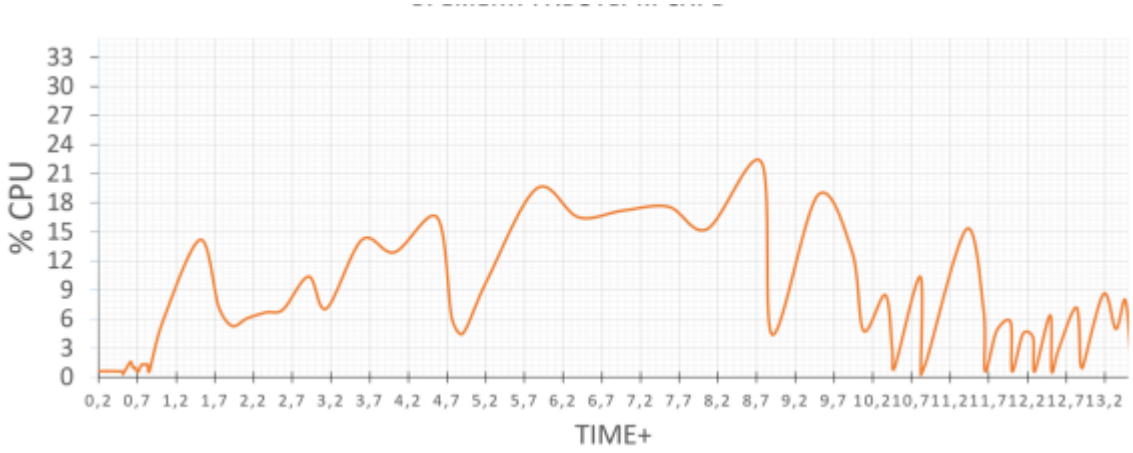

Рисунок 2.23 – Зміна показників відсотка завантаженості CPU відносно часу роботи RPCAPD

Дані для побудови графіка ми брали з виводу утиліти top. Для цього використовувалися значення колонок %CPU (відсоток завантаження процесора) та TIME+ (час роботи процесу). Результат виконання команди представлений на Рисунку 2.24.

```
ypi:~ $ top | grep rpcapd
 9804 root      20   0   14688   3880   3696 S   3.2   0.9   0:00.10 rpcapd
 9804 root      20   0   14688   3880   3696 S  12.0   0.9   0:00.48 rpcapd
 9804 root      20   0   14688   3880   3696 S   0.6   0.9   0:00.50 rpcapd
 9804 root      20   0   14688   3880   3696 S   0.3   0.9   0:00.51 rpcapd
 9804 root      20   0   14688   3880   3696 S   0.6   0.9   0:00.53 rpcapd
 9804 root      20   0   14688   3880   3696 S   1.0   0.9   0:00.56 rpcapd
 9804 root      20   0   14688   3880   3696 S   1.6   0.9   0:00.61 rpcapd
```

Рисунок 2.24 – Результат команди top для відео середньої якості

На графіку чітко видно, як змінилося навантаження на процесор відповідно до обсягу переданих даних і часу роботи процесу. У пікові моменти через бездротову мережу між домашнім маршрутизатором і ПК передавалися дані для відтворення відео.

Під час тестування середня швидкість прийому пакетів бездротовою мережею становила приблизно 7,7 Мбіт/с. Водночас, передача даних по дротовому каналу зв'язку різко зросла під час запуску відео середньої якості (Рисунок 2.25). Також, виходячи з графіків, простежується чітка залежність між швидкістю передачі даних та зміною показника відсотка завантаженості CPU відносно часу роботи процесу RPCAPD. Піки швидкості відповідають пікам навантаження на мережу. Обриси

графіків ідентичні. Варто зазначити, що зі збільшенням якості зображення швидкість переданих даних також зросла.

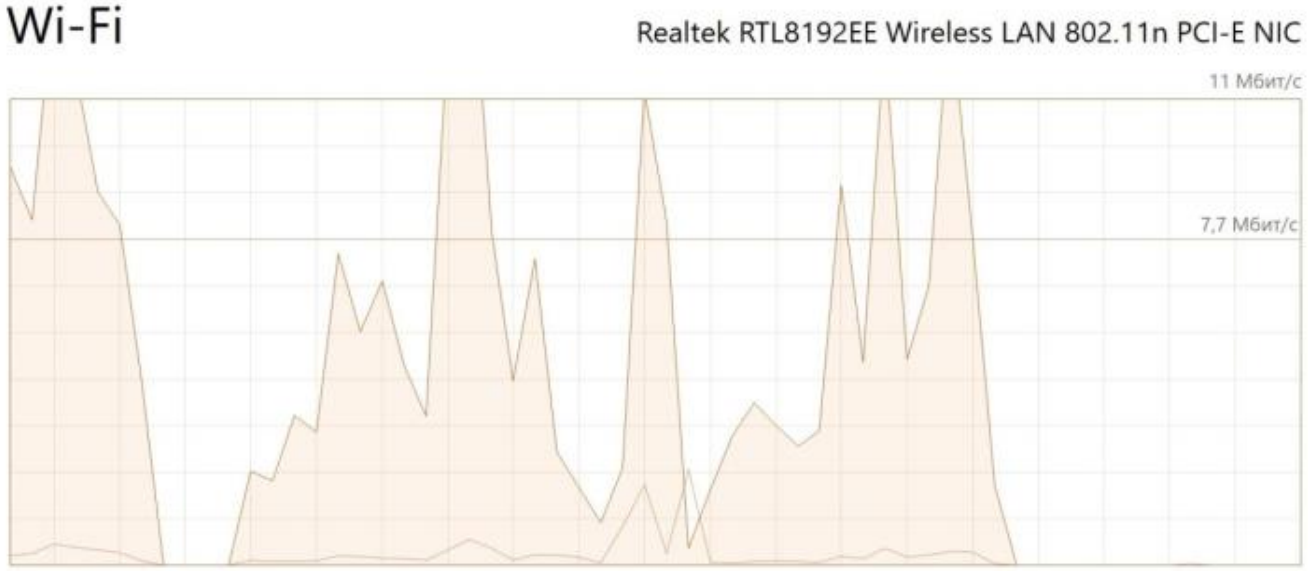

Рисунок 2.25 – Активність відправлення та отримання даних у момент відтворення відео середньої якості

2.3 Оцінка ефективності передачі даних каналом зв'язку

Для оцінки ефективності передачі даних каналом зв'язку необхідно ретельно вивчити передані пакети в мережі. З цією метою під час проведення попередніх досліджень були створені файли формату «.pcap» за допомогою сніферів Wireshark та Tcpdump. Ці файли містять усі захоплені пакети. На Рисунку 2.26 представлений приклад перехопленої пакетної операції, згенерованої за допомогою демона RPCAPD.

Для оцінки ефективності використання методу RPCAPD була розроблена формула, яка складається зі співвідношення обсягу корисного трафіку до обсягу перехоплюваного трафіку (2.1):

$$\text{Ефективність} = \frac{V_{\text{корисного трафіку}}}{V_{\text{перехоплюваного трафіку}}} \quad (2.1)$$

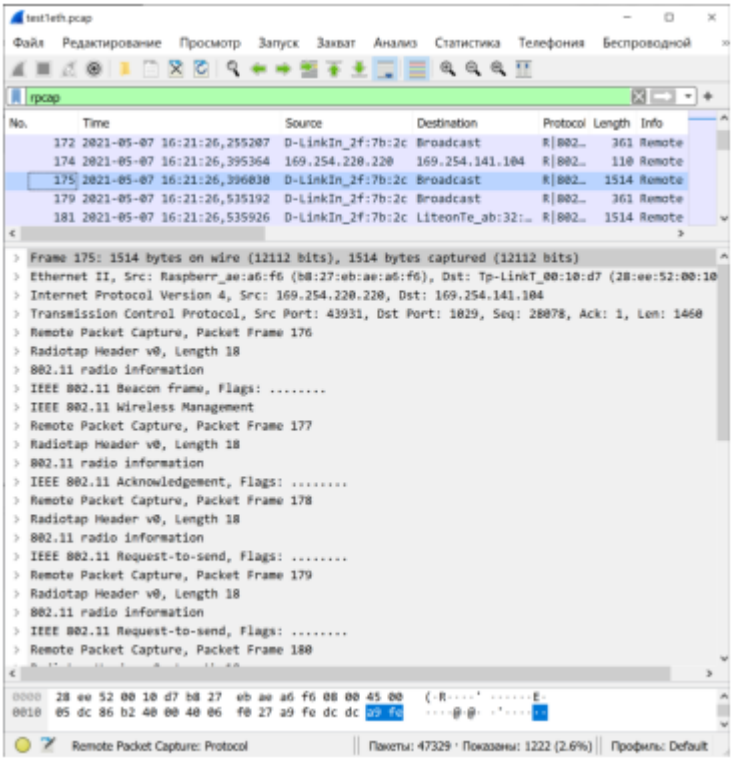

Рисунок 2.26 – Приклад перехопленої пакетної операції

Проаналізовані файли з перехопленими пакетами показали, що віддалений демон виконує пакетні операції. Тобто, він об'єднує перехоплені пакети в один, інкапсулюючи їх для передачі мережею Ethernet (у випадку з цими дослідженнями). Ці заголовки деінкапсулюються на приймаючій стороні, внаслідок чого отриманий пакет розбивається на кілька інших пакетів, що містяться в ньому.

Наприклад, у файлі, отриманому після дослідження перехоплення трафіку з використанням пристрою Raspberry Pi під час перегляду відео низької якості, фрейм №175 містить 10 частин. Заголовок RPCAP займає в кожному фреймі 28 байт.

Таким чином, ми отримаємо, що у випадку, коли демон не використовує пакетні операції, навантаження на мережу збільшується за формулою (2.2):

$$\text{Data} = x + 82 \text{ (байт)} \tag{2.2}$$

де Data — новий отриманий обсяг даних; x — початковий обсяг фрейму; 82 байта — обсяг доданих даних, з яких: 14 байт — Ethernet II; 20 байт — IPv4; 20 байт — TCP; 28 байт — RPCAP.

У випадку, коли використовується пакетна операція, наприклад, з 10 фреймів

різної довжини, навантаження на мережу збільшується за формулою (2.3):

$$\text{Data} = 14 + 20 + 20 + 28 + x_1 + 28 + x_2 + 28 + x_3 + \cdots + 28 + x_{10}\ (\text{байт}) \quad (2.3)$$

де x_i — початковий обсяг фрейму під номером i.

Скоротивши формулу (2.3), отримаємо формулу навантаження на мережу з використанням пакетних операцій (2.4):

$$\text{Data} = 54 + 28 \cdot n + \sum_{i=1}^{n} x_i\ (\text{байт}) \quad (2.4)$$

де n — кількість фреймів у пакетній операції; i — порядковий номер фрейму.

На основі формул (2.2) і (2.4) ми бачимо, що при використанні пакетної операції обсяг даних, що передаються, зменшується приблизно в 0,4 рази порівняно з відправкою тих самих пакетів окремими фреймами.

Однак, при інкапсуляції корисний обсяг збільшується приблизно на 82 байти. Це означає, що ефективність додавання додаткових заголовків при використанні RPCAPD можна розрахувати за формулою (2.5):

$$\text{Ефективність} = \frac{x}{x+82} \quad (2.5)$$

де x – це розмір корисного навантаження (оригінального фрейму).

На Рисунку 2.27 можна побачити приклад фрейму, що містить заголовок ERSPAN. TCP-пакет транспортується, загорнутий у заголовок GRE (Generic Routing Encapsulation), який має тип протоколу ERSPAN. Аналізатор, що знаходиться на кінці ERSPAN-тунелю, побачить вже деінкапсульований трафік, починаючи із заголовка Ethernet II. Довжина заголовка GRE може варіюватися від 4 до 16 байт залежно від увімкнених опцій.

Виходячи з цього прикладу, ми можемо розрахувати довжину заголовка ERSPAN, визначити його ефективність і порівняти з методом RPCAPD. Так, ми отримуємо, що довжина заголовка ERSPAN становить 24 байти. Розрахунки проводилися на основі мінімально можливої довжини кожного заголовка з прикладу та різниці між загальним обсягом (116 байт) і цими значеннями.

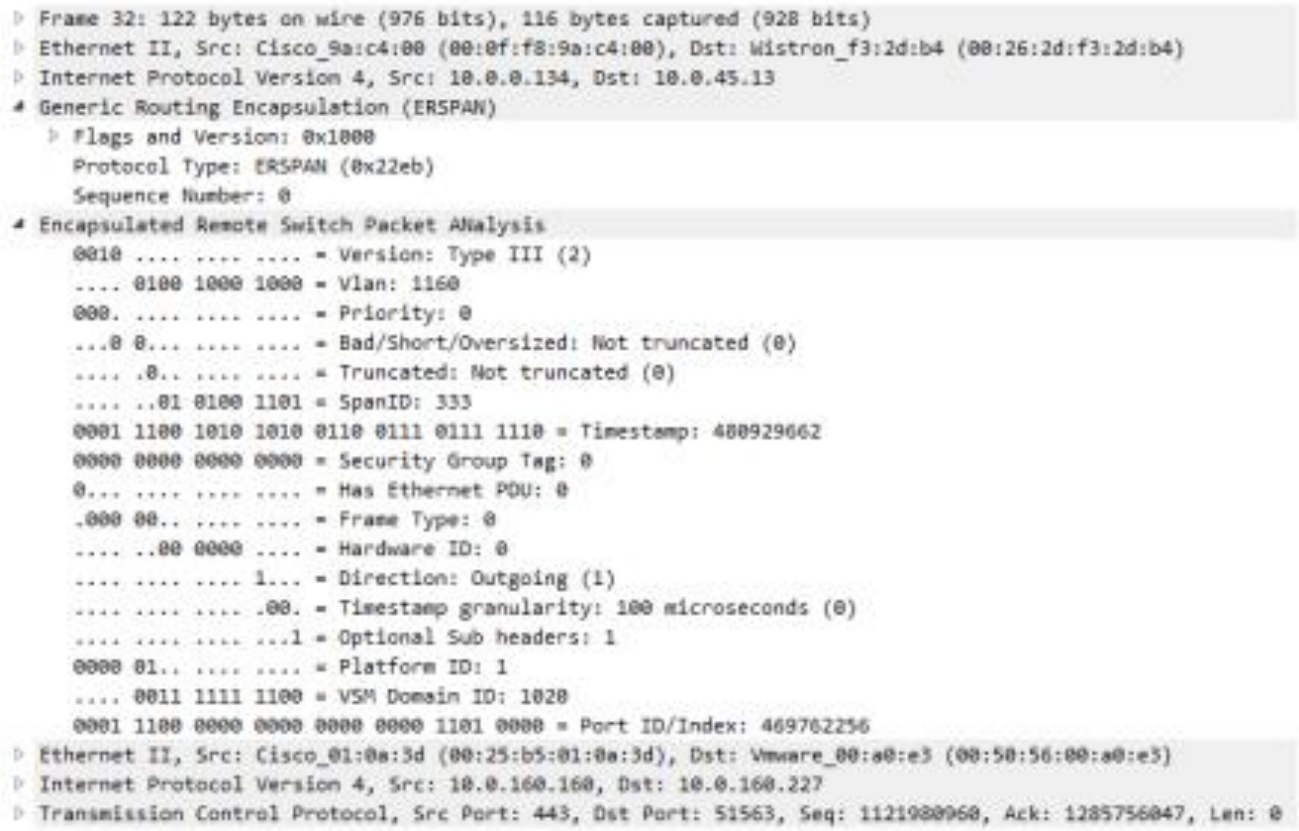

Рисунок 2.27 – Приклад перехопленого пакету заголовку ERSPAN

При таких значеннях корисний обсяг збільшується приблизно на 62 байти. Це дозволяє обчислити ефективність додавання додаткових заголовків при використанні ERSPAN за формулою (2.6):

$$\text{Ефективність} = \frac{x}{x+62} \tag{2.6}$$

Взявши за приклад наявний пакет із заголовком ERSPAN та пакет із заголовком RPCAP, ми побудуємо графік ефективності використання цих методів віддаленого перехоплення, щоб порівняти отримані результати.

На Рисунку 2.28 зображено дві прямі, побудовані за формулами (5) та (6). На осі абсцис (горизонтальна вісь) відображено зміну корисного обсягу, а на осі ординат (вертикальна вісь) — зміну ефективності відносно корисного трафіку. Чим вище значення по осі ординат, тим вищою є ефективність відповідного методу. Це візуально демонструє, який метод є ефективнішим при різних обсягах передаваних даних.

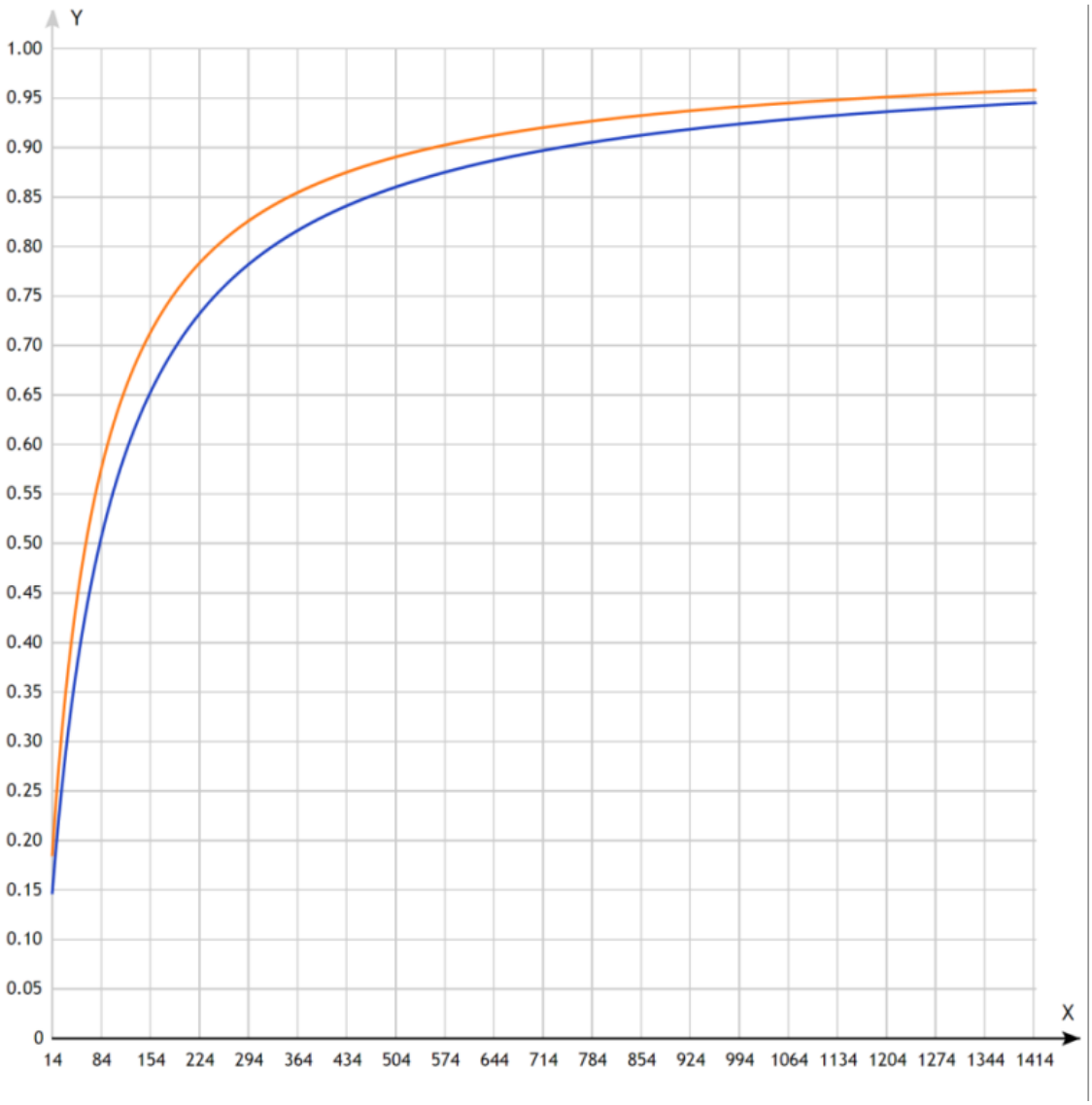

Рисунок 2.28 – Зміна показників ефективності для методу ERSPAN і RPCAP

2.3 Розробка інструменту перехоплення мережевого трафіку

2.3.1 Призначення програми

Адміністрування комп'ютерних мереж передбачає перехід від керування роботою окремих пристроїв до аналізу трафіку на різних мережевих ділянках, керування логічною конфігурацією мережі та її робочими параметрами.

Таким чином, завдання адміністрування можна розділити на дві основні групи: контроль за функціонуванням мережевого обладнання та управління роботою мережі в цілому.

Ключовою метою стає досягнення та підтримка таких параметрів роботи

інформаційної системи, які б найбільш точно відповідали потребам користувача. Для цього необхідно буде оцінити її роботу за такими характеристиками, як: особливості трафіку, протоколи, що використовуються, швидкість відгуку сервера на запити та специфіка сценаріїв використання.

2.3.2 Спосіб перехоплення трафіку та визначення методики

Для методики адміністрування мережі підходять одразу кілька способів перехоплення трафіку. В першу чергу, при побудові мережі з вищезгаданих пристроїв, таких як комутатор і маршрутизатор, варто звернути увагу на здатність конкретного обладнання до можливості дзеркалювання трафіку (port mirroring).

Розглянуті ОС кожної системи побудовані на основі BSD UNIX (Berkeley Software Distribution) — системи розповсюдження програмного забезпечення у вихідних кодах. Особливістю пакетів ПЗ BSD була спеціальна ліцензія BSD, яку коротко можна охарактеризувати так: весь вихідний код — власність BSD, всі правки — власність їхніх авторів.

Тому, у випадку, якщо пристрій не може дзеркалювати трафік, на допомогу приходить бібліотека libpcap спільно з демоном RPCAPD. Варто врахувати, що для цього, можливо, доведеться змінити код під ядро ОС, яка використовується на мережевому пристрої.

У випадку використання перехоплення трафіку на робочих станціях, чудово підходить використання способу перехоплення демоном RPCAPD. Також, якщо адміністратор захоче сканувати радіоефір бездротової мережі, він може виділити для цього окремий пристрій для сканування, який передаватиме інформацію на окремий сервер, що обробляє та аналізує трафік.

Таким чином, виходячи з вищесказаного, отримаємо, що для реалізації методу необхідний окремий сервер, що збирає дані, до якого звертається адміністратор для аналізу отриманих результатів та складання точної картини стану мережі. Приклад локальної обчислювальної мережі (ЛОМ), що складається з робочих станцій, комутаторів та сервера для збору й аналізу даних, представлений на Рисунку 2.29.

На ньому також зображений потік перехоплюваного трафіку, а саме: з пристроїв 1, 2 і 3 перехоплюваний трафік збирається на сервері. Після чого, результат оброблених даних відправляється Адміністратору мережі, щоб виявити нестандартну активність.

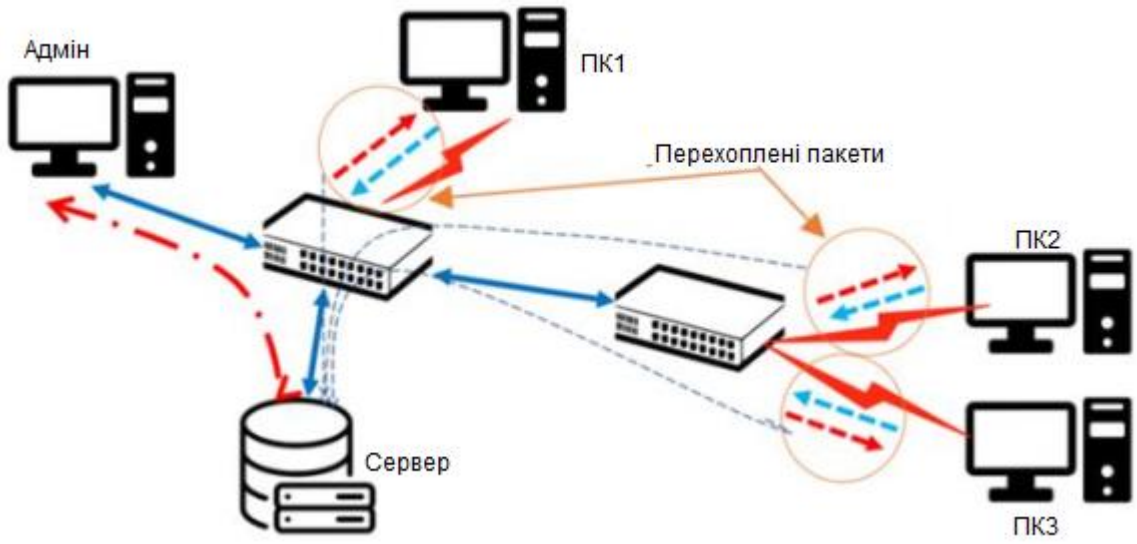

Рисунок 2.29 – Схема прикладу перехоплення трафіку при адмініструванні мережі

Для реалізації перехоплення трафіку адміністратору необхідно виконати наступні кроки:

1. Детально ознайомитися з топологією мережі, розташуванням пристроїв, їхніми ролями та взаємозв'язками.

2. Обрати найбільш підходящі технології (наприклад, дзеркалювання портів, використання RPCAPD) та ідентифікувати конкретні мережеві пристрої (комутатори, маршрутизатори, робочі станції), які будуть джерелами трафіку.

3. Вибрати сервер або робочу станцію, яка буде виконувати функцію збирача трафіку та зберігати перехоплені пакети.

4. Здійснити необхідні налаштування на мережевому обладнанні та пристроях для активації обраних методів перехоплення (наприклад, налаштувати дзеркалювання портів або запустити демон RPCAPD).

5. Активувати процес перехоплення, використовуючи відповідні інструменти (наприклад, Wireshark, спеціально розроблений клієнт).

6. Проаналізувати зібрані дані, виявити аномалії, проблеми з продуктивністю або безпекою, та сформулювати рекомендації щодо оптимізації або усунення виявлених недоліків.

2.3.3 Програмна реалізація

Розробка інструменту для віддаленого перехоплення трафіку за допомогою демона RPCAPD є ключовим кроком у впровадженні методики адміністрування корпоративних мереж. Цей інструмент, розроблений на Python з графічним інтерфейсом, значно спрощує процес моніторингу мережі, надаючи адміністраторам наочний засіб для аналізу трафіку та виявлення аномалій. Архітектура рішення ґрунтується на модульному підході, що забезпечує гнучкість, розширюваність та легкість у підтримці. Основні компоненти інструменту включають main.py, rpc_client.py, packet_analyzer.py, ui_main_window.py (рис.2.30).

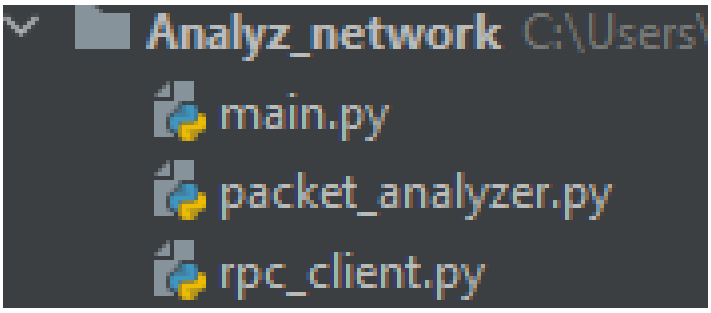

Рисунок 2.30 – Структурна файлів інструменту для перехоплення трафіку

Головний файл програми з GUI main.py відповідає за ініціалізацію графічного інтерфейсу користувача (GUI), координацію взаємодії між іншими модулями та обробку подій користувача. Він побудований на основі фреймворку PySide6 (або PyQt) [10], що дозволяє створювати кросплатформні додатки з професійним виглядом (рис.2.31).

Створює головне вікно програми (MainWindow) та всі його елементи керування (поля введення IP-адреси та порту, кнопки підключення, випадаючий список інтерфейсів, поле для BPF-фільтра, кнопки запуску/зупинки захоплення, текстове поле для виводу пакетів та рядок стану). Дизайн цих елементів може бути

візуально створений за допомогою Qt Designer, а потім імпортований як ui_main_window.py.

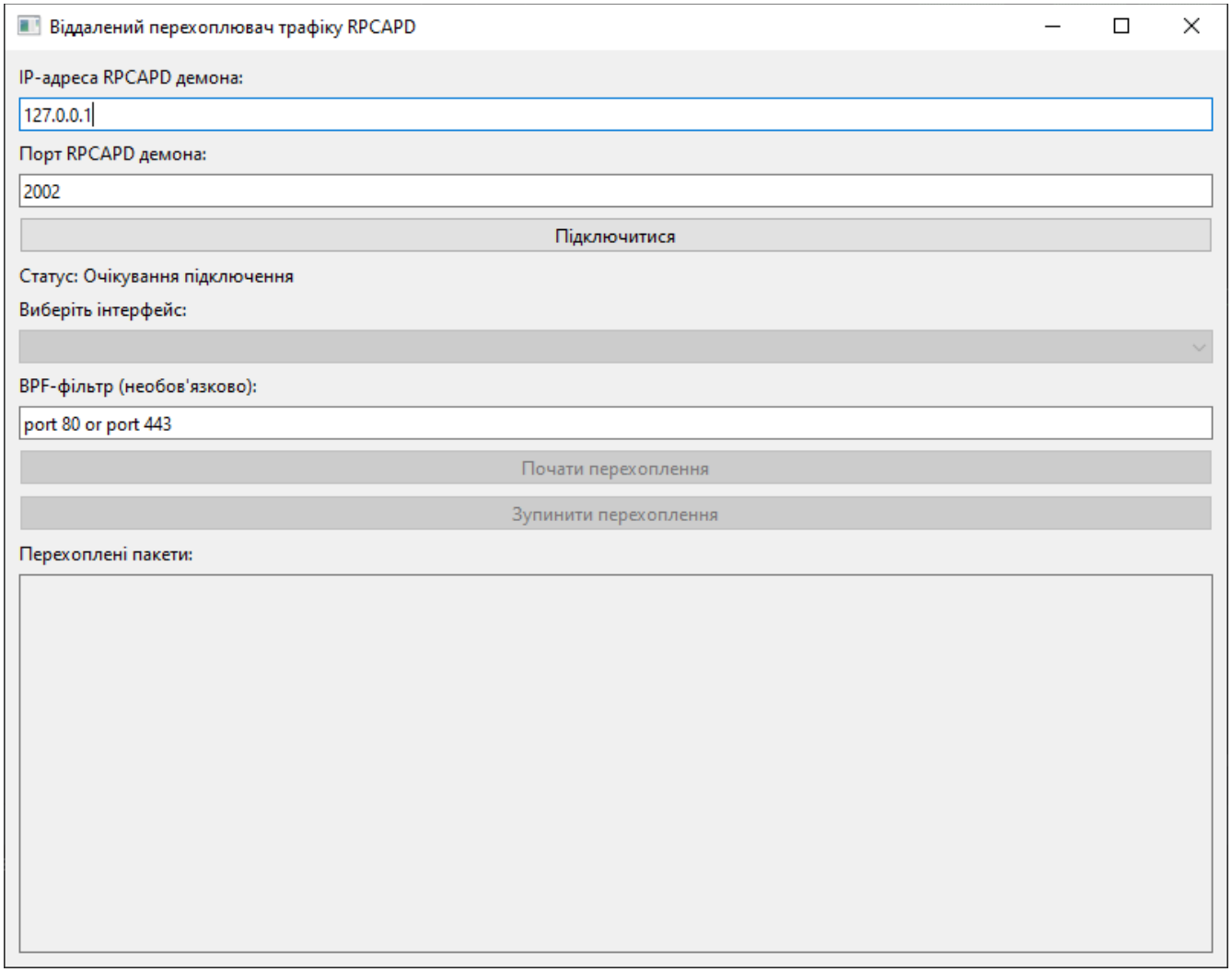

Рисунок 2.31 – Головне вікно інструменту

Обробляє події натискання кнопки "Підключитися". При цьому створюється окремий потік (RPCAPClient), що дозволяє виконувати мережеві операції асинхронно, запобігаючи "зависанню" інтерфейсу. main.py передає RPCAPClient вказані користувачем IP-адресу та порт.

Обробляє натискання кнопок "Почати перехоплення" та "Зупинити перехоплення". При цьому він викликає відповідні методи у RPCAPClient для ініціації або завершення процесу віддаленого захоплення трафіку на вибраному інтерфейсі з застосуванням вказаного BPF-фільтра.

Отримує сигнали від RPCAPClient щодо статусу підключення (наприклад, "Підключено", "Відключено", "Помилка", "Захоплення розпочато/зупинено") та відповідно оновлює рядок стану, доступність кнопок та елементів керування (наприклад, вимикає поля введення IP/порту після підключення, активує вибір інтерфейсу після отримання його списку).

Отримує сирі байти перехоплених пакетів від RPCAPClient. Ці байти передаються до екземпляра PacketAnalyzer для їхнього парсингу та перетворення на зрозумілий текстовий опис. Отриманий опис потім додається до текстового поля "Перехоплені пакети", забезпечуючи візуалізацію трафіку в реальному часі. Для зручності користувача реалізовано автоматичне прокручування донизу.

Реалізує механізм відображення помилок, отриманих від RPCAPClient, за допомогою діалогових вікон (QMessageBox), що інформує користувача про будь-які проблеми з мережевим з'єднанням або демоном RPCAPD.

Модуль для взаємодії з RPCAPD демоном rpc_client.py є важливим модулем, що реалізує логіку взаємодії з віддаленим демоном RPCAPD. Він працює в окремому потоці (QThread) для забезпечення неблокуючої роботи GUI.

Відповідає за ініціалізацію TCP-з'єднання з вказаними IP-адресою та портом віддаленого демона. Реалізація протоколу RPCAP. Це найскладніша частина. Модуль має реалізовувати протокольні взаємодії RPCAP: handshake встановлення початкового з'єднання. Якщо активовано, модуль повинен обмінюватися даними автентифікації (логін/пароль) з демоном. Надсилати запит на отримання списку доступних мережевих інтерфейсів на віддаленому пристрої та парсити відповідь. Відправляти команди демону RPCAPD для початку захоплення трафіку на вибраному інтерфейсі з певним BPF-фільтром, а також команди для зупинки захоплення. У постійному циклі отримувати сирі байти перехоплених пакетів від демона. Використовує сигнали PySide6 (packet_received, status_message, interfaces_found, error_occurred) для асинхронної передачі інформації до головного потоку GUI.

Модуль для обробки та відображення пакетів packet_analyzer.py є відповідальним за перетворення сирих байтів мережевих пакетів, отриманих від

rpc_client.py, на зрозумілий та читабельний формат для відображення в GUI.

Використовує потужні бібліотеки, такі як Scapy (рекомендовано) або Pyshark (потребує встановлення tshark), для розбору сирих байтів на окремі мережеві протоколи (Ethernet, IP, TCP, UDP тощо) та вилучення ключової інформації (IP-адреси джерела/призначення, порти, протоколи, розмір даних). Генерує короткий, але інформативний текстовий опис кожного пакета, включаючи часову мітку, порядковий номер, основні протоколи та адреси. Має механізми для коректної обробки пакетів, які не можуть бути повністю розпізнані, виводячи їхній сирий вигляд або базову інформацію про довжину. Може бути розширений для збереження перехоплених пакетів у стандартний формат .pcap, що дозволить надалі аналізувати їх за допомогою спеціалізованих інструментів, таких як Wireshark.

2.3.4 Опис роботи інструменту віддаленого перехоплювача трафіку

При запуску програми інтерфейс відображає поля для введення IP-адреси та порту RPCAPD демона, які за замовчуванням встановлені на 127.0.0.1 та 2002 відповідно. Рядок статусу показує: "Статус: Очікування підключення". Кнопки "Почати перехоплення" та "Зупинити перехоплення", а також випадаючий список "Виберіть інтерфейс" неактивні, оскільки підключення до демона ще не встановлено.

Після натискання кнопки "Підключитися" (рис.2.32), програма ініціює імітацію підключення до RPCAPD демона. У демонстраційному режимі це відбувається швидко:

– рядок статусу змінюється на "Статус: Знайдено інтерфейсів: 3". Це свідчить про успішну імітацію зв'язку з демоном та отримання від нього списку доступних мережевих інтерфейсів;

– випадаючий список "Виберіть інтерфейс" стає активним і заповнюється імітованими інтерфейсами (на скріншоті видно "eth0", припускаючи, що також доступні "wlan0" та "lo", як це було зазначено в демо-коді);

– кнопка "Почати перехоплення" стає активною, сигналізуючи про

готовність до ініціації захоплення трафіку.

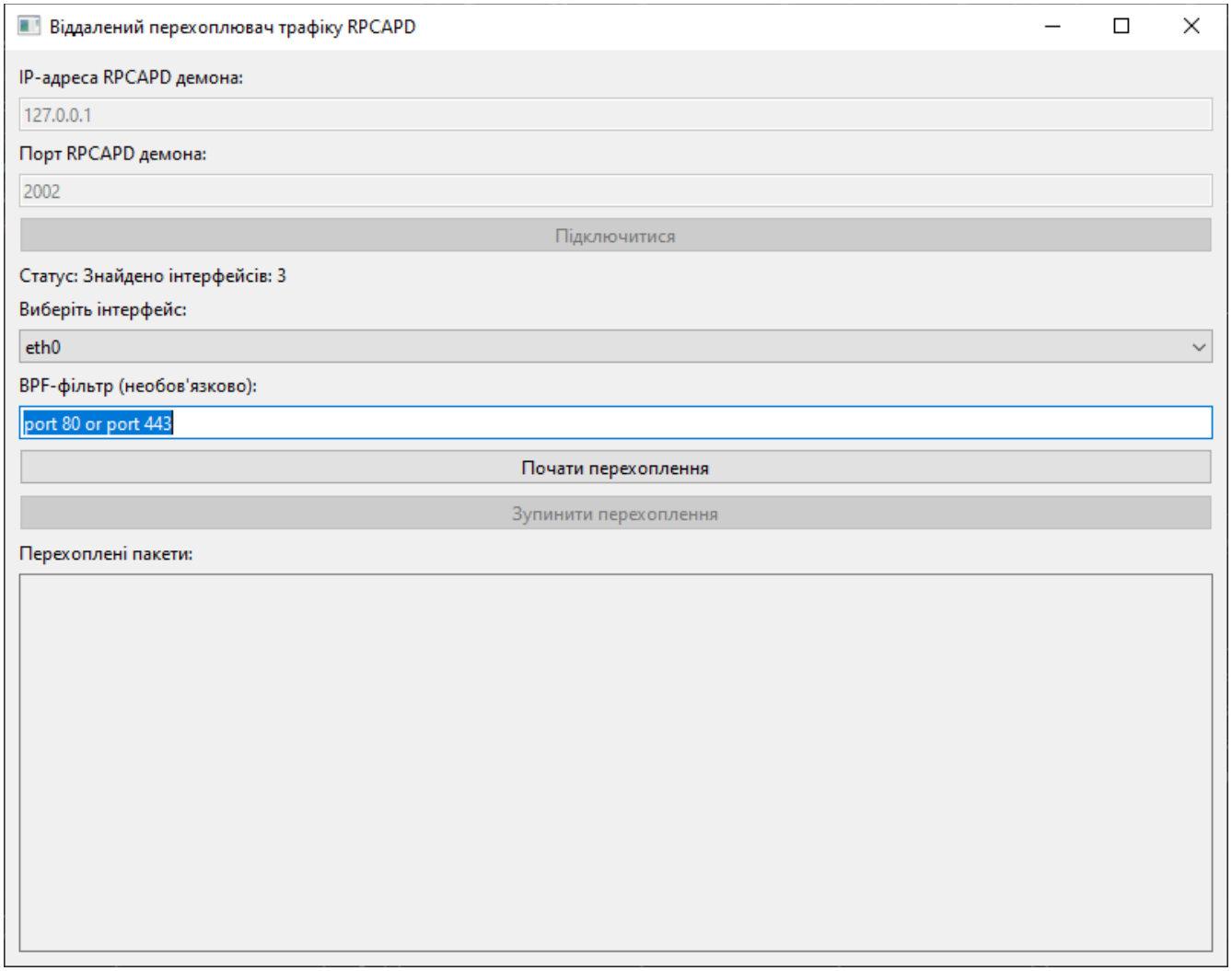

Рисунок 2.32 – Підключення та виявлення інтерфейсів

Коли користувач обирає інтерфейс (наприклад, "eth0") та, можливо, вводить BPF-фільтр (за замовчуванням "port 80 or port 443"), а потім натискає "Почати перехоплення", програма переходить у режим імітації захоплення (рис.2.33):

– рядок статусу змінюється, відображаючи, що імітація захоплення триває. У цьому скріншоті статус уже "Імітація захоплення зупинена", що вказує на те, що захоплення вже відбувалося і було зупинено, але вікно "Перехоплені пакети" демонструє результати, отримані під час цього процесу;

– у великому текстовому полі "Перехоплені пакети" починають з'являтися записи. Кожен запис представляє собою імітований мережевий пакет. Формат запису включає: ч асову мітку: [ЧЧ:ХХ:СС.мс] (наприклад, [23:41:30.958]).

Порядковий номер пакета: №XXX (наприклад, №1).

– короткий опис пакета – це імітовані дані, які відображають тип пакета та частину його сирого вмісту (наприклад, Raw Type:0x2b98 або Raw Type:0x13be). У реальному режимі тут був би більш детальний парсинг, що відображає протоколи (Ethernet, IP, TCP/UDP), джерела та призначення.

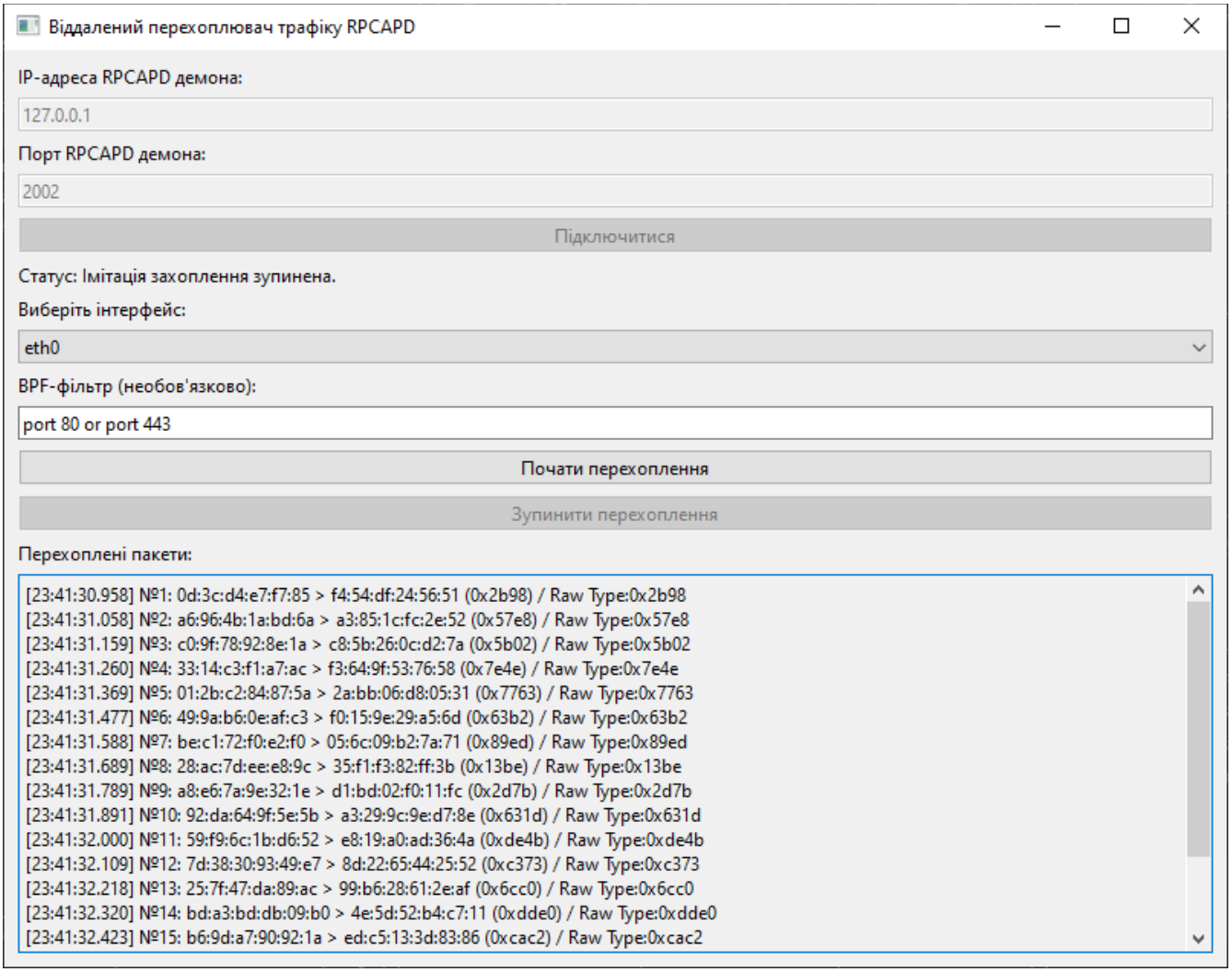

Рисунок 2.33 – Процес захоплення трафіку

Після завершення демонстрації перехоплення (натискання кнопки "Зупинити перехоплення", або у випадку повного відключення) інструмент повертається до початкового стану:

– рядок статусу відображає "Статус: Відключено.", вказуючи на завершення сесії;

– кнопки "Почати перехоплення" та "Зупинити перехоплення" знову стають неактивними;

– поля введення IP-адреси та порту, а також кнопка "Підключитися" знову активуються, дозволяючи розпочати нову сесію. Вікно "Перехоплені пакети" зберігає попередній вивід, якщо його не очищали перед новим підключенням.

ВИСНОВКИ

У рамках виконання випускної кваліфікаційної роботи було успішно вирішено низку раніше поставлених завдань.

На початковому етапі було проаналізовано предметну область досліджуваної теми, за результатами якої обґрунтовано необхідність розробки методики віддаленого перехоплення трафіку в корпоративних мережах. Проаналізовано основні методи перехоплення трафіку в дослідній галузі. Визначено маловивчені способи та проведено різні дослідження, що мали успішний результат. Отримано результати дослідження ефективності використання методу перехоплення трафіку шляхом застосування протоколу RPCAP.

На основі проведеного комплексного аналізу способів перехоплення розроблено таблицю аналізу існуючих методів віддаленого перехоплення трафіку, завдяки якій було описано методику для віддаленого перехоплення в корпоративній мережі.

Були сформовані вимоги до розроблюваної методики. Віддалене перехоплення трафіку вимагає значних обчислювальних ресурсів для зберігання та аналізу великих обсягів даних, а також кваліфікованих спеціалістів для правильної інтерпретації отриманих результатів. Окрім того, зростаюча популярність шифрованого трафіку (HTTPS, VPN) створює додаткові виклики для його ефективного аналізу без компрометації конфіденційності.

Найбільшу ефективність віддалене перехоплення трафіку демонструє у поєднанні з іншими інструментами інформаційної безпеки, такими як системи виявлення вторгнень (IDS), системи запобігання вторгненням (IPS), системи керування інформаційною безпекою та подіями (SIEM), що дозволяє створити багатошарову та комплексну систему захисту.

ПЕРЕЛІК ПОСИЛАНЬ

1. Mehdi, Merouane. (2019). Interception of P2P Traffic in a Campus Network. Revista Română de Informatică și Automatică. 29. 10.33436/v29i2y201902.

2. Oh C, Ha J, Roh H. A Survey on TLS-Encrypted Malware Network Traffic Analysis Applicable to Security Operations Centers. Applied Sciences. 2022; 12(1):155.

3. Goodall, J.R.; Ragan, E.D.; Steed, C.A.; Reed, J.W.; Richardson, G.D.; Huffer, K.M.; Bridges, R.A.; Laska, J.A. Situ: Identifying and Explaining Suspicious Behavior in Networks. IEEE Trans. Vis. Comput. Graph. 2019, 25, 204–214.

4. Axon, L.; AlAhmadi, B.A.; Nurse, J.R.C.; Goldsmith, M.; Creese, S. Data presentation in security operations centres: Exploring the potential for sonification to enhance existing practice. J. Cybersecur. 2020.

5. Shen, M.; Liu, Y.; Zhu, L.; Xu, K.; Du, X.; Guizani, N. Optimizing feature selection for efficient encrypted traffic classification: A systematic approach. IEEE Netw. 2020, 34, 20–27.

6. Conti, M.; Li, Q.Q.; Maragno, A.; Spolaor, R. The Dark Side(-Channel) of Mobile Devices: A Survey on Network Traffic Analysis. IEEE Commun. Surv. Tutor. 2018, 20, 2658–2713.

7. Wireshark Protocol Analyzer [Електронний ресурс] – Режим доступу до ресурсу: http://www.wireshark.org/.

8. Tcpdump [Електронний ресурс] – Режим доступу до ресурсу: http://www.tcpdump.org/.

9. Traffic Interception [Електронний ресурс] – Режим доступу до ресурсу: https://docs.debookee.com/en/latest/mitm.html.

10. Cisco. Cisco Encrypted Traffic Analytics. 2019. [Електронний ресурс] – Режим доступу до ресурсу: https://www.cisco.com/c/en/us/td/docs/ios-xml/ios/sec_data_eta.

11. Scapy [Електронний ресурс] – Режим доступу до ресурсу: https://scapy.net/.

ДОДАТОК А

Текст програми віддаленого перехоплення мережевого трафіку як інструмент забезпечення інформаційної безпеки в корпоративних мережах

Міністерство освіти і науки України
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ
“ДНІПРОВСЬКА ПОЛІТЕХНІКА”

ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ

ВІДДАЛЕНЕ ПЕРЕХОПЛЕННЯ МЕРЕЖЕВОГО ТРАФІКУ ЯК ІНСТРУМЕНТ ЗАБЕЗПЕЧЕННЯ ІНФОРМАЦІЙНОЇ БЕЗПЕКИ В КОРПОРАТИВНИХ МЕРЕЖАХ

Текст програми

804.02070743.25018-01 12 01

Листів 10

2025

АНОТАЦІЯ

Дана програма представляє собою десктопну комп'ютерну систему, що забезпечує віддалене перехоплення та аналіз мережевого трафіку завдяки інтеграції з протоколом RPCAPD.

Програма призначена для збору, обробки та візуалізації даних мережевого трафіку, що дозволяє автоматизувати процеси моніторингу, виявлення аномалій та забезпечення інформаційної безпеки в корпоративних мережах. Система підтримує застосування BPF-фільтрів для цільового захоплення трафіку, забезпечуючи гнучкість у застосуванні.

Програма розроблена з використанням Python та бібліотеки PySide6 для графічного інтерфейсу, що гарантує її кросплатформенність та забезпечує зручний користувацький інтерфейс. Вона може бути використана в різних сферах, таких як адміністрування мереж, аудит безпеки, розслідування інцидентів та навчання у галузі кібербезпеки, де потрібно ефективне віддалене управління та аналіз мережевих даних.

ЗМІСТ

Main.py

```
import sys
from PySide6.QtWidgets import QApplication, QMainWindow, QVBoxLayout, QWidget,
QPushButton, QLineEdit, QLabel, QTextEdit, QComboBox, QMessageBox
from PySide6.QtCore import QThread, Signal # Signal вже імпортований

# Імпорт ваших модулів
from rpc_client import RPCAPClient
from packet_analyzer import PacketAnalyzer

class MainWindow(QMainWindow):
    def __init__(self):
        super().__init__()
        self.setWindowTitle("Віддалений перехоплювач трафіку RPCAPD")
        self.setGeometry(100, 100, 800, 600)

        self.central_widget = QWidget()
        self.setCentralWidget(self.central_widget)
        self.layout = QVBoxLayout(self.central_widget)

        # Налаштування підключення
        self.layout.addWidget(QLabel("IP-адреса RPCAPD демона:"))
        self.ip_input = QLineEdit("127.0.0.1") # Для тестування можна використати
локальний IP
        self.layout.addWidget(self.ip_input)

        self.layout.addWidget(QLabel("Порт RPCAPD демона:"))
        self.port_input = QLineEdit("2002")
        self.layout.addWidget(self.port_input)

        self.connect_button = QPushButton("Підключитися")
        self.connect_button.clicked.connect(self.connect_to_rpcapd)
        self.layout.addWidget(self.connect_button)

        # Статус
        self.status_label = QLabel("Статус: Очікування підключення")
        self.layout.addWidget(self.status_label)

        # Вибір інтерфейсу
        self.layout.addWidget(QLabel("Виберіть інтерфейс:"))
        self.interface_combo = QComboBox()
        self.interface_combo.setEnabled(False) # Вимкнено до підключення
        self.layout.addWidget(self.interface_combo)
```

```
        # Фільтр
        self.layout.addWidget(QLabel("BPF-фільтр (необов'язково):"))
        self.filter_input = QLineEdit("port 80 or port 443")
        self.layout.addWidget(self.filter_input)

        # Кнопки управління захопленням
        self.start_capture_button = QPushButton("Почати перехоплення")
        self.start_capture_button.clicked.connect(self.start_capture)
        self.start_capture_button.setEnabled(False)
        self.layout.addWidget(self.start_capture_button)

        self.stop_capture_button = QPushButton("Зупинити перехоплення")
        self.stop_capture_button.clicked.connect(self.stop_capture)
        self.stop_capture_button.setEnabled(False)
        self.layout.addWidget(self.stop_capture_button)

        # Вивід пакетів
        self.layout.addWidget(QLabel("Перехоплені пакети:"))
        self.packet_output = QTextEdit()
        self.packet_output.setReadOnly(True)
        self.layout.addWidget(self.packet_output)

        self.rpc_client_thread = None
        self.packet_analyzer = PacketAnalyzer() # Ініціалізуємо аналізатор пакетів

    def connect_to_rpcapd(self):
        host = self.ip_input.text()
        port = int(self.port_input.text())

        if self.rpc_client_thread and self.rpc_client_thread.isRunning():
            QMessageBox.warning(self, "Попередження", "Вже відбувається підключення
або захоплення.")
            return

        self.rpc_client_thread = RPCAPClient(host, port) # Використовуємо RPCAPClient
        self.rpc_client_thread.packet_received.connect(self.handle_raw_packet) #
Підключаємо до нового обробника
        self.rpc_client_thread.status_message.connect(self.update_status)
        self.rpc_client_thread.interfaces_found.connect(self.populate_interfaces)
        self.rpc_client_thread.error_occurred.connect(self.display_error) # Додаємо
обробник помилок
        self.rpc_client_thread.start()

        self.connect_button.setEnabled(False)
```

```
        self.ip_input.setEnabled(False)
        self.port_input.setEnabled(False)
        self.packet_output.clear() # Очищаємо вікно виводу при новому підключенні

    def update_status(self, message):
        self.status_label.setText(f"Статус: {message}")
        if "Підключено." in message:
            self.interface_combo.setEnabled(True)
            self.start_capture_button.setEnabled(True)
        if "Відключено." in message or "Помилка" in message: # Додаємо обробку
помилок для розблокування кнопок
            self.connect_button.setEnabled(True)
            self.ip_input.setEnabled(True)
            self.port_input.setEnabled(True)
            self.interface_combo.setEnabled(False)
            self.start_capture_button.setEnabled(False)
            self.stop_capture_button.setEnabled(False)

    def display_error(self, message):
        QMessageBox.critical(self, "Помилка RPCAPD", message)
        self.update_status(f"Помилка: {message}") # Оновлюємо статус

    def populate_interfaces(self, interfaces):
        self.interface_combo.clear()
        if interfaces:
            self.interface_combo.addItems(interfaces)
        else:
            self.interface_combo.addItem("Інтерфейси не знайдено")

    def start_capture(self):
        selected_interface = self.interface_combo.currentText()
        capture_filter = self.filter_input.text()

        if not selected_interface or selected_interface == "Інтерфейси не знайдено":
            QMessageBox.warning(self, "Попередження", "Будь ласка, виберіть мережевий
інтерфейс або дочекайтесь їх завантаження.")
            return

        if self.rpc_client_thread and self.rpc_client_thread.isRunning():
            self.rpc_client_thread.start_remote_capture(selected_interface, capture_filter)
            self.start_capture_button.setEnabled(False)
            self.stop_capture_button.setEnabled(True)
        else:
            QMessageBox.critical(self, "Помилка", "Клієнт RPCAPD не запущено.")
```

```
    def stop_capture(self):
        if self.rpc_client_thread:
            self.rpc_client_thread.stop_capture()
        # Кнопки будуть розблоковані через сигнал update_status після повного відключення
        self.start_capture_button.setEnabled(True) # Можна залишити, якщо хочемо дозволити перезапуск
        self.stop_capture_button.setEnabled(False)


    def handle_raw_packet(self, raw_bytes):
        """
        Обробляє сирі байти пакета, використовуючи PacketAnalyzer,
        та відображає результат у QTextEdit.
        """
        packet_summary = self.packet_analyzer.process_raw_packet(raw_bytes)
        self.packet_output.append(packet_summary)
        # Опціонально: автоматичне прокручування донизу

self.packet_output.verticalScrollBar().setValue(self.packet_output.verticalScrollBar().maximum())

if __name__ == "__main__":
    app = QApplication(sys.argv)
    window = MainWindow()
    window.show()
    sys.exit(app.exec())
```

Packet_analyzer.py

```
from scapy.all import Ether, IP, TCP, UDP, Raw
import datetime

class PacketAnalyzer:
    """
    Клас для аналізу та відображення мережевих пакетів.
    Розбирає сирі байти пакетів та формує зручний для відображення формат.
    Використовує Scapy для парсингу.
    """
    def __init__(self):
        self.packet_count = 0
        # self.capture_file = None # Можна додати функціонал збереження в pcap

    def process_raw_packet(self, raw_bytes):
        """
        Обробляє сирі байти отриманого пакета та повертає його summary.
        """
        self.packet_count += 1
        timestamp = datetime.datetime.now().strftime("%H:%M:%S.%f")[:-3]
        summary_info = ""

        try:
            # Спробуємо розібрати як Ethernet-кадр
            # Якщо сирі байти не схожі на Ethernet, Scapy може видати помилку
            packet = Ether(raw_bytes)
            summary_info = f"[{timestamp}] №{self.packet_count}: {packet.summary()}"

            # Додаємо деталі, якщо пакети розпізнані
            if IP in packet:
```

```
            summary_info += f" {packet[IP].src} -> {packet[IP].dst}"
            if TCP in packet:
                summary_info += f" TCP:{packet[TCP].sport} > {packet[TCP].dport}"
            elif UDP in packet:
                summary_info += f" UDP:{packet[UDP].sport} > {packet[UDP].dport}"
            elif packet.haslayer(Raw): # Якщо є сирі дані
                summary_info += f" DataLen:{len(packet[Raw].load)}"
        else:
            # Якщо це не IP пакет, але Scapy його розпізнав як Ethernet
            summary_info += f" Type:{hex(packet.type)}"

    except Exception:
        # Якщо Scapy не може розібрати пакет як Ethernet,
        # просто виводимо його як послідовність байтів
        summary_info = f"[{timestamp}] №{self.packet_count}: Невідомий пакет (довжина: {len(raw_bytes)} байт) - {raw_bytes[:20].hex()}..."

    return summary_info

# def save_packet_to_pcap(self, raw_bytes, filename="captured_traffic.pcap"):
#     """
#     Зберігає сирі байти пакета у файл формату .pcap.
#     """
#     # Приклад збереження за допомогою Scapy
#     # from scapy.all import wrpcap
#     # wrpcap(filename, raw_bytes, append=True)
#     pass
```