Галицький фаховий коледж імені В’ячеслава Чорновола

відділення комп’ютерних технологій

циклова комісія інформатики та комп’ютерних дисциплін

ДОПУСТИТИ ДО ЗАХИСТУ

Завідувач відділенням

комп’ютерних технологій

Наталія СТЕФУРАК /_____/

підпис

«___» _____ 2024 р.

ПОЯСНЮВАЛЬНА ЗАПИСКА

до кваліфікаційної роботи

освітньо-професійного ступеня «фаховий молодший бакалавр»

зі спеціальності 122 «Комп’ютерні науки»

на тему: «Програмний засіб для аналізу мережевого трафіку на мові Python з використанням Scapy»

Студент групи КН-41	Сергій ЛОБАС	_____
		(підпис)
Керівник роботи	Степан ІВАСЬЄВ	_____
		(підпис)
Консультанти:		
з техніко-економічного обґрунтування	Любов МЕЛЕНЧУК	_____
		(підпис)
нормоконтролер	Наталія КУЛЬЧИНСЬКА	_____
		(підпис)

Тернопіль – 2024

Галицький фаховий коледж імені В’ячеслава Чорновола

відділення комп’ютерних технологій

циклова комісія інформатики та комп’ютерних дисциплін

ЗАТВЕРДЖУЮ

Завідувач відділенням

комп’ютерних технологій

Наталія СТЕФУРАК /_____/

підпис

«___» _____ 2023 р.

ЗАВДАННЯ

на кваліфікаційну роботу

на здобуття освітньо-професійного ступеня «фаховий молодший бакалавр»

студенту Лобасу Сергію Андрійовичу

(прізвище, ім’я та по-батькові студента)

1. Тема роботи Програмний засіб для аналізу мережевого трафіку на мові Python з використанням Scapy затверджено наказом по коледжу від “27” листопада 2023 р., №234а-н
2. Термін здачі студентом завершеної роботи “___” _____ 2024 р.
3. Вихідні дані до роботи: Розробка програмного засобу для аналізу мережевого трафіку за допомогою Python та бібліотеки для роботи з мережами Scapy.
4. Перелік питань, які повинні бути розроблені:

а) основна частина: встановлення та формалізація вимог, проєктування інтерфейсу, програмна реалізація застосунку, тестування.

б) техніко-економічне обґрунтування: аналіз ринку збуту, обґрунтування витрат на проєктування, обґрунтування необхідності розробки.

5. Перелік графічного матеріалу: діаграма послідовностей, діаграма станів, діаграма варіантів використання.
6. Консультанти роботи: Меленчук Л.І.

Розділ	Консультанти	Підпис, дата	
		Завдання видано	Завдання прийнято
з техніко-економічного обґрунтування	Меленчук Л.І. (вчена ступінь, звання П.І.Б. консультанта)		

КАЛЕНДАРНИЙ ПЛАН

виконання кваліфікаційної роботи

№ п/п	Найменування етапу	Терміни	
		початку	завершення
1.	Вибір теми кваліфікаційної роботи.	23.11.2023	01.12.2023
2.	Детальне ознайомлення з предметною областю. Аналіз програмних рішень.	02.04.2024	05.04.2024
3.	Опрацювання теоретичних матеріалів, написання розділу роботи.	06.04.2024	07.04.2024
4.	Формалізація вимог. Аналіз технологій реалізації. Написання розділу роботи.	08.04.2024	15.04.2024
5.	Проєктування та реалізація графічного інтерфейсу. Реалізація основних функцій програми.	18.04.2024	05.05.2024
6.	Тестування програмного засобу.	05.05.2024	16.05.2024
7.	Створення відповідного розділу роботи	16.05.2024	22.05.2024
8.	Обґрунтування вартості проєкту.	26.05.2024	01.05.2024
9.	Оформлення пояснювальної записки.	14.06.2024	14.06.2024
10.	Попередній захист кваліфікаційної роботи.	17.06.2024	17.06.2024
11.	Підготовка до захисту та виправлення помилок.	17.06.2024	24.06.2024
12.	Захист кваліфікаційної роботи.	26.06.2024	26.06.2024

Дата видачі “___” _____ 2023 р. Керівник _____/ Степан Івасьєв

Завдання прийняв до виконання _____/ Сергій Лобас

Реферат

Кваліфікаційна робота. Розробка програмного засобу для аналізу мережевого трафіку на мові Python з використанням Scapy. 77 сторінок, 36 рисунків, 2 додатки.

Об'єкт дослідження –засоби аналізу мережевого трафіку.

Метою кваліфікаційної роботи є розробка програмного засобу для аналізу мережевого трафіку за допомогою технологій Python та бібліотек Scapy, tkinter.

Програмний засіб повинен забезпечувати зручний спосіб взаємодії для користувача, а також оптимізований і точний процес захоплення і аналізу мережевого трафіку.

Крім того, необхідно спроєктувати та реалізувати можливість збереження захопленого трафіку та його імпорту з інших аналізаторів мережевого трафіку, що повинно забезпечити зручність для користувачів.

Для досягнення поставленої мети використано різноманітні технології та інструменти для взаємодії з мережами.

Після завершення процесу розробки було отримано функціональний та готовий до використання програмний засіб.

IP, TCP, UDP, MAC, WI-FI, ETHERNET, HTTP, APPLICATION.

Abstract

Development of a software tool for analyzing network traffic in Python using Scapy. Qualification work. Lobas Serhii Andreyovych. Vyacheslav Chornovol Halych Vocational College, department of computer technologies. Specialty 122 "Computer Sciences", 2024.

Pages - 77, figures - 36, appendices - 2.

The research object is a software tool for network traffic analysis.

The purpose of the work is to develop a software tool for network traffic analysis using Python technologies and Scapy, tkinter libraries.

The software should provide a convenient way of interaction for the user, as well as an optimized and accurate process of capturing and analyzing network traffic.

In addition, it is necessary to design and implement the ability to save captured traffic and import it from other network traffic analyzers, which should ensure convenience for users.

Various technologies and tools for interaction with networks were used to achieve the goal.

After the completion of the development process, a functional and ready-to-use software was obtaine

IP, TCP, UDP, MAC, WI-FI, ETHERNET, HTTP, APPLICATION.

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

IP – Інтернет-протокол

TCP – Протокол керування передачею

UDP – Протокол користувацьких датаграм

DNS – Система доменних імен

HTTP – Протокол передачі гіпертексту

HTTPS – Захищений протокол передачі гіпертексту

FTP – Протокол передачі файлів

SSH – Захищена оболонка

SSL – Рівень захищених сокетів

TLS – Захист транспортного рівня

VPN – Віртуальна приватна мережа

MAC – Контроль доступу до медіа

NAT – Трансляція мережевих адрес

IDS – Система виявлення вторгнень

IPS – Система запобігання вторгнень

SIEM – Управління інформацією та подіями безпеки

Wi-Fi – Бездротова мережа

LAN – Локальна мережа

WAN – Глобальна мережа

DDoS – Розподілене відмовлення в обслуговуванні

MITM – Атака "людина посередині"

Botnet – Мережа ботів

ВСТУП

В сучасності, всесвітня мережа Інтернет щороку набуває в рази все більших масштабів, а також, на неї покладене зобов’язання підтримувати комерційну, економічну, суспільну, культурну, освітню та інші види діяльностей людини. Вже давно не дивина і не таємниця, що Інтернет – один з найкращих засобів для розвитку, навчання та відпочинку, мільярди користувачів, розуміючи це чи ні, привносять щось нове в мережу, ділячись цим з іншими користувачами.

Проте, з усіма позитивними аспектами Інтернету зростають і ризики. Такою проблемою з інтернетом є кіберзлочинність і є однією з найсерйозніших проблем сучасного інформаційного суспільства. Адже необмежений доступ до мережевих ресурсів, величезні обсяги інформації, його швидкий розвиток та доступність роблять Інтернет привабливим середовищем для ведення зловмисницької діяльності через мережу. Окрім кіберзлочинності, що становить серйозну загрозу, існують й інші проблеми, пов'язані з мережевим трафіком. До них належать перевантаження мереж, збої в передачі даних, аномалії в мережевому трафіку, що можуть призводити до зниження якості обслуговування та ефективності роботи мережі.

Мережевий трафік забезпечує передачу даних між користувачами, серверами та різними пристроями в Інтернеті. Аналіз мережевого трафіку стає критично важливим для підтримки стабільної та безпечної роботи мереж, задля виявлення різного роду мережевих аномалій. Кіберзлочинці використовують мережевий трафік для атаки на мережеву інфраструктуру, крадіжки даних, розповсюдження шкідливого програмного забезпечення та інших злочинних дій. Відсутність ефективних засобів моніторингу та аналізу мережевого трафіку може призвести до серйозних наслідків, таких як витік конфіденційної інформації, фінансові втрати, зниження продуктивності систем і навіть повне припинення роботи організацій.

Мета цього проєкту – розробка програмного засобу для аналізу мережевого трафіку, який дозволить ефективно виявляти аномалії, допомогти запобігати кіберзагрозам та забезпечувати безпеку мережевої інфраструктури. Програмний засіб повинен надавати користувачам можливість швидкого доступу до інформації про їх трафік і до аналізу цих даних у реальному часі.

Розробка програмного засобу для аналізу мережевого трафіку має важливе соціальне та освітнє значення. Перш за все, підвищення безпеки мережевої інфраструктури сприятиме захисту персональних даних користувачів, зменшенню кількості кіберзлочинів, запобіганню мережевим аномаліям та підвищенню загальної довіри до Інтернету як до безпечного середовища.

З освітньої точки зору, впровадження такого програмного засобу може стати важливим інструментом для навчання студентів та спеціалістів в галузі кібербезпеки. Вони отримають можливість працювати з сучасними технологіями аналізу трафіку, вивчати методи виявлення та запобігання загрозам, а також навчаться ефективно працювати з мережевими аномаліями. Це сприятиме підготовці висококваліфікованих фахівців, здатних забезпечувати стабільну та безпечну роботу мережевих систем.

Таким чином, розробка та впровадження програмного засобу для аналізу мережевого трафіку є необхідною умовою для забезпечення безпечного та ефективного функціонування сучасної мережевої інфраструктури, а також сприятиме підвищенню загального рівня кібербезпеки у суспільстві.

РОЗДІЛ 1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАВДАНЬ

1.1 Опис предметної області

У сучасному інформаційному середовищі, де мережі стають все більш складними та об'ємними, аналіз мережевого трафіку визначається як критично важливий етап для забезпечення стабільності та ефективності комунікаційних систем, особливо, враховуючи тенденції інформаційного розвитку в суспільстві за останні десятки років, що майже унеможливлює ведення господарської, підприємницької, побутової та інших діяльностей без застосування всесвітньої мережі, також всесвітньо відому як «Інтернет». Такий розвиток, крім користі, також приносить і загрози інформаційній безпеці в мережі, а мінімалізувати ці загрози можна за допомогою аналізу трафіку мережі, щоб можна було визначити актуальність небезпеки, щоб зменшити потенційну шкоду. Для вирішення цих проблем буде реалізований продукт, який би зробив аналіз зручним та ефективним. Завдяки ньому можна буде отримувати детальну інформацію про взаємодію пристроїв у мережі, виявити та вирішити різноманітні проблеми, пов'язані зі з'єднанням та передачею даних, а також фіксувати підозрілу активність з глобальної мережі, виявляти та аналізувати потенційно небезпечний трафік, що допоможе у попередженні можливих загроз та атак на мережу. Аналізатор мережевого трафіку дозволятиме виявляти та моніторити мережеві пакети, що проходять через мережу, що є важливим для забезпечення її стабільності та ефективності. Програмний засіб дозволятиме визначити ефективність передачі даних у мережі та виявляти можливості для її оптимізації. Це важливо для забезпечення швидкодії та надійності передачі інформації для різних веб-застосунків, завантаження та відправки файлів тощо. А також, будуть наявні інструменти для вивчення та аналізу характеристик мережі, які допомагають визначити її працездатність та ефективність.

1.2 Аналіз наявних рішень

Перш ніж розпочати проєктування програмного продукту, варто проаналізувати вже існуючі, або схожі програмні рішення. Аналіз існуючих програмних рішень у сфері аналізу мережевого трафіку є важливим етапом у визначенні необхідних функцій та особливостей майбутнього програмного засобу. У цьому розділі буде проводитись огляд та порівняльний порівняння трьох найпопулярніших та одних з найкращих існуючих інструментів: Wireshark, Tcpdump, Tshark.

Кожен з цих інструментів має свої як переваги, так і недоліки, або обмеження, які варто розглянути задля якісної реалізації, щоб наперед уникнути створення продукту з великою кількістю недоліків.

1.2.1 Wireshark

Wireshark – це передовий аналізатор мережевих протоколів, розроблений для фіксації та інтерактивного перегляду трафіку в комп'ютерних мережах. Завдяки своєму різноманітному та потужному набору функцій, Wireshark є найпопулярнішим у світі інструментом у цьому сегменті (рисунок 1.1).

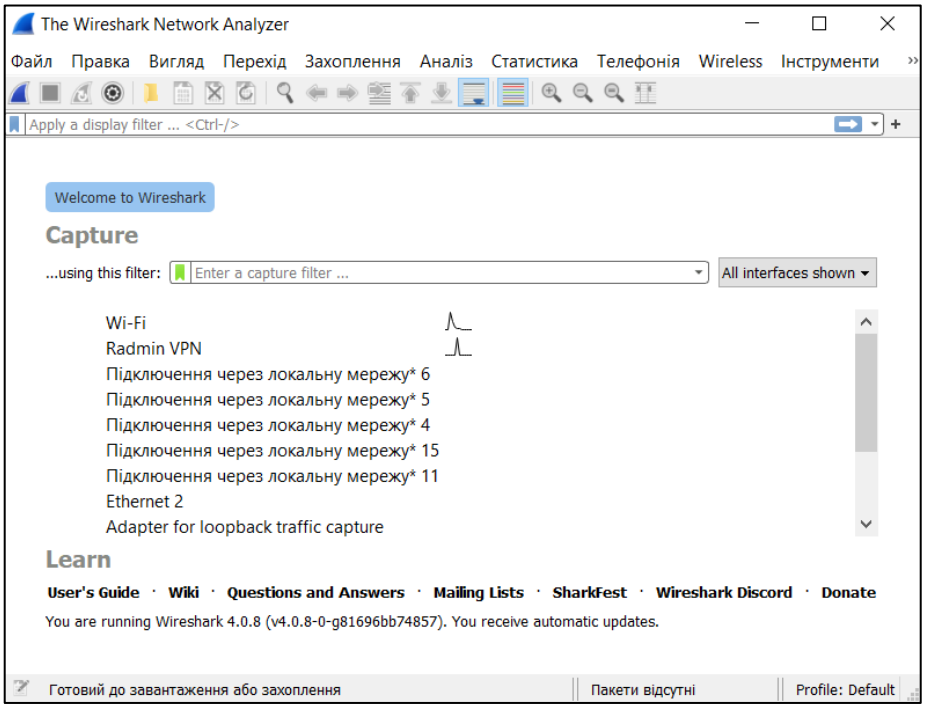


Рисунок 1.1 – Інтерфейс програмного застосунку Wireshark

Його універсальність дозволяє використовувати його на різних платформах, таких як Windows, macOS, Linux і UNIX. Основною перевагою Wireshark є його здатність зафіксувати та аналізувати мережевий трафік у реальному часі, що робить його незамінним інструментом для мережевих інженерів, адміністраторів і безпекових фахівців.

Програма забезпечує детальний перегляд кожного пакета даних, що проходить через мережу, надаючи користувачеві можливість аналізувати протоколи та виявляти потенційні проблеми в мережевому з'єднанні. Wireshark володіє інтуїтивно зрозумілим графічним інтерфейсом, який дозволяє легко навігувати та фільтрувати дані. Його підтримка на різних операційних системах робить його доступним для широкого кола користувачів та дозволяє використовувати його у різних середовищах [3].

1.2.2 Tcpdump

Tcpdump – це невід'ємний інструмент для аналізу мережевого трафіку в середовищі UNIX. Завдяки його можливостям користувач може отримати прямий доступ до даних, що передаються по мережі і вивчити їх деталі. Використання tcpdump здійснюється через командний рядок, що робить його ефективним інструментом для аналізу в реальному часі та відладки мережевих проблем. Його основна сфера застосування – це відлагодження мережевих проблем, моніторинг та аналіз трафіку. Tcpdump здатний прослуховувати та аналізувати різні мережеві протоколи, що робить його корисним інструментом для інженерів мереж і системних адміністраторів. Ця утиліта дозволяє користувачеві визначити, які дані передаються через мережу, а також використовувати фільтри для вибору конкретного трафіку. Це допомагає зосередитися лише на необхідних деталях та спрощує аналіз загального обсягу трафіку. Важливою особливістю Tcpdump є його невимагаючий графічний інтерфейс, що робить його зручним для використання в ситуаціях, де не

доступний графічний інтерфейс, або в разі, коли важлива швидкість та ефективність аналізу (рисунок 1.2).

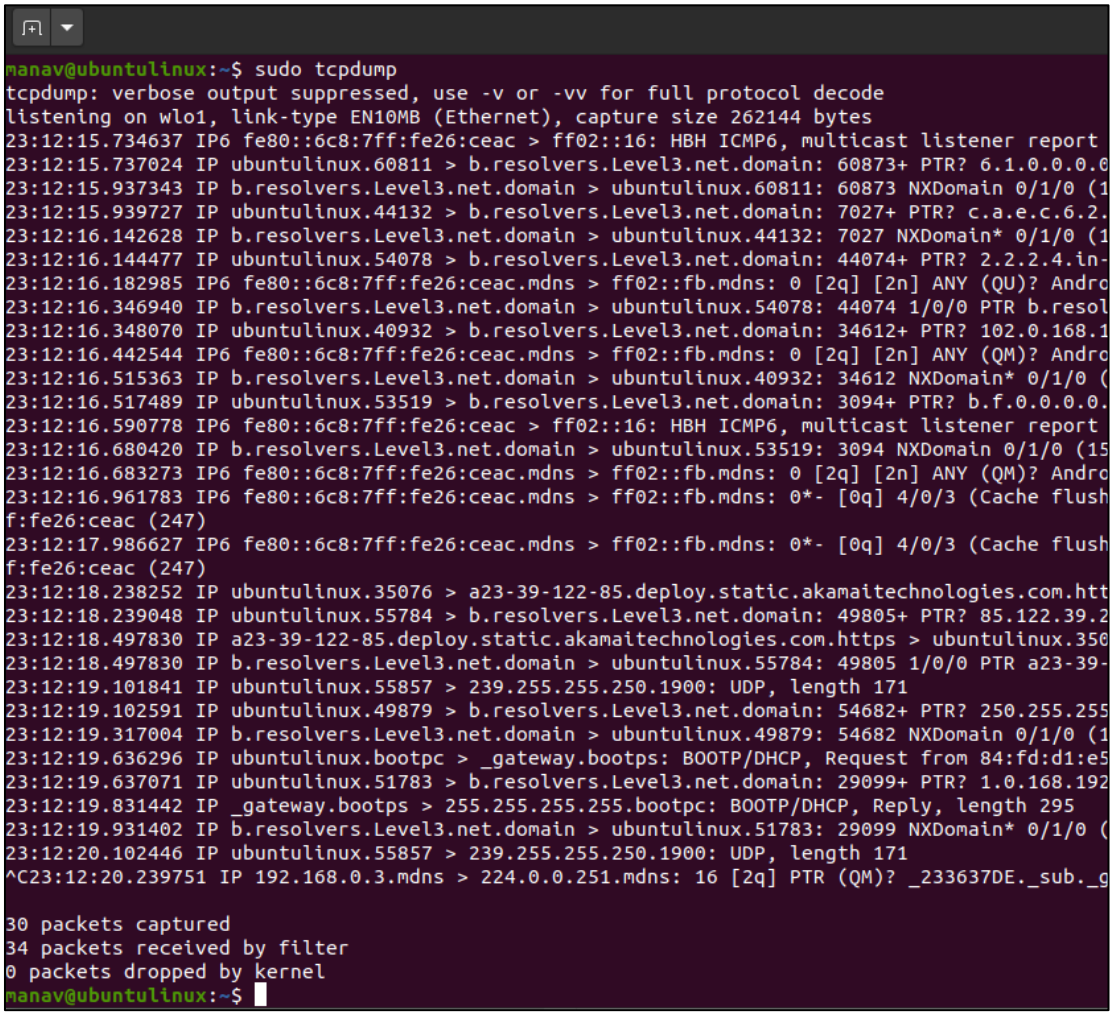


Рисунок 1.2 – Запуск сніфера tcpdump в терміналі операційної системи Ubuntu

Таким чином, tcpdump становить важливий інструмент для тих, хто потребує детального та ефективного моніторингу мережі в середовищі UNIX

1.2.3 tshark

tshark – це високоефективний аналізатор мережевого трафіку, який служить для захоплення та аналізу пакетів у термінальному режимі. Цей інструмент є командним рядком для тих випадків, коли використання графічного інтерфейсу є зайвим або неможливим. Програма базується на тій же

функціональності, що і відомий аналізатор Wireshark, але при цьому не вимагає графічного інтерфейсу користувача. Однією з ключових переваг tshark є його висока ефективність та можливість використання у ситуаціях, де ресурси графічного інтерфейсу обмежені або відсутні. Програма надає широкі можливості для захоплення та аналізу різноманітних мережевих пакетів, дозволяючи користувачеві отримати докладну інформацію про обмін даними у мережі. Використання tshark дозволяє ефективно взаємодіяти з програмними інтерфейсами та автоматизувати процес аналізу мережі через командний рядок. Його функціоналість включає можливість фільтрації пакетів, підтримку різноманітних мережевих протоколів та зручний вивід результатів для подальшого аналізу (рисунок 1.3).

```
root@kali:~# tshark -r packets.pcap -T pdml
Running as user "root" and group "root". This could be dangerous.
<?xml version="1.0" encoding="utf-8"?>
<?xml-stylesheet type="text/xsl" href="pdml2html.xsl"?>
<!-- You can find pdml2html.xsl in /usr/share/wireshark or at https://code.wireshark.org/
w/gitweb?p=wireshark.git;a=blob_plain;f=pdml2html.xsl. -->
<pdml version="0" creator="wireshark/3.0.5" time="Tue Jan 28 11:46:48 2020" capture_file=
ets.pcap">
<packet>
  <proto name="geninfo" pos="0" showname="General information" size="98">
    <field name="num" pos="0" show="1" showname="Number" value="1" size="98" />
    <field name="len" pos="0" show="98" showname="Frame Length" value="62" size="98" />
    <field name="caplen" pos="0" show="98" showname="Captured Length" value="62" size="98
    <field name="timestamp" pos="0" show="Jan 28, 2020 11:46:40.459901028 EST" showname="
red Time" value="1580230000.459901028" size="98" />
  </proto>
  <proto name="frame" showname="Frame 1: 98 bytes on wire (784 bits), 98 bytes captured (
its) on interface 0" size="98" pos="0">
    <field name="frame.interface_id" showname="Interface id: 0 (eth0)" size="0" pos="0" s
0">
      <field name="frame.interface_name" showname="Interface name: eth0" size="0" pos="0"
="eth0" />
    </field>
    <field name="frame.encap_type" showname="Encapsulation type: Ethernet (1)" size="0" p
" show="1" />
    <field name="frame.time" showname="Arrival Time: Jan 28, 2020 11:46:40.459901028 EST"
="0" pos="0" show="Jan 28, 2020 11:46:40.459901028 EST" />
    <field name="frame.offset_shift" showname="Time shift for this packet: 0.000000000 se
" size="0" pos="0" show="0.000000000" />
    <field name="frame.time_epoch" showname="Epoch Time: 1580230000.459901028 seconds" si
" pos="0" show="1580230000.459901028" />
```

Рисунок 1.3 – Запуск аналізатора мережевого трафіку в терміналі операційної системи Kali Linux

Таким чином, tshark є потужним інструментом для спеціалістів з мережевої безпеки та адміністраторів систем, які потребують ефективного інструмента для аналізу мережевого трафіку в середовищі командного рядка.

1.2.4 Порівняльний аналіз схожих продуктів

Тепер, коли кожна з програм була представлена попередньо окремо, варто їх порівняти за наступними критеріями: інтерфейс, здатність до автоматизації, фільтрація трафіку, підтримка протоколів, зручність використання для користувача, споживання ресурсів комп'ютера, наявність спеціалізованих функцій, кросплатформеність (таблиця 1.1).

Таблиця 1.1 - Порівняльна характеристика утиліт Wireshark, tcpdump та tshark

Критерії	Wireshark	tcpdump	tshark
Інтерфейс	Графічний	Командна строка	Командна строка
Здатність до автоматизації	Обмежена	Обмежена	Цілковита здатність
Фільтрація трафіку	Висока гнучкість в фільтрації	Присутня	Висока гнучкість в фільтрації
Підтримка протоколів	Велика кількість	Обмежена кількість	Обмежена кількість
Зручність використання	Висока складність	Середня складність	Помірно висока складність
Споживання ресурсів	Великі обсяги	Малі обсяги	Малі обсяги
Спеціалізовані функції	Багато функцій	Мало функцій	Багато функцій
Кросплатфоменість	Windows, Linux, MacOS	Windows, Linux, MacOS	Windows, Linux, MacOS

Ця порівняльна таблиця дає можливість отримати об'єктивний огляд їхніх переваг і обмежень.

Щодо інтерфейсу, Wireshark відрізняється графічним інтерфейсом, тоді як tcpdump та tshark опираються на командний рядок. Здатність до автоматизації виявилася обмеженою у Wireshark та tcpdump, в той час як tshark вразив цілковитою здатністю до автоматизації.

Щодо фільтрації трафіку, Wireshark та tshark вирізняються високою гнучкістю, тоді як у tcpdump теж присутня фільтрація, але менш гнучка.

У підтримці протоколів Wireshark видається лідером, маючи велику кількість підтримуваних протоколів, в той час як tcpdump та tshark обмежені кількістю підтримуваних протоколів.

Зручність використання користувачем у Wireshark вища, але при цьому його інтерфейс може викликати певні труднощі, особливо для новачків. У tcpdump середня складність використання, тоді як у tshark помірно висока складність.

Споживання ресурсів також гарантує Wireshark великими обсягами, у той час як у tcpdump та tshark воно є малим.

З урахуванням цих характеристик, можна сформулювати чіткі вимоги до майбутньої програмної реалізації. Важливо враховувати баланс між функціональністю, зручністю використання та оптимальним споживанням ресурсів для створення високоякісного та ефективного інструменту для аналізу мережі.

1.3 Аналіз вимог до програмного засобу та постановка завдання

Аналіз вимог до програмного засобу є критично важливим етапом, що передує реалізації проєкту, і дозволяє визначити функціональність та вимоги до майбутнього інструменту для аналізу мережевого трафіку. У даному пункті буде розглянуто ключові вимоги, які випливають з порівняльного аналізу наявних рішень з попереднього пункту. На цьому етапі будуть визначені перелік проблем,

які не слід втілювати у життя, склад функцій, що входитимуть до процесів та буде сформований список поставлених задач, які необхідно виконати в процесі роботи. При детальному аналізі вимог до програмного засобу для аналізу мережевого трафіку враховуються як функціональні, так і нефункціональні аспекти для забезпечення повноти та ефективності роботи інструменту.

Функціональні вимоги:

– Розроблення модулю для захоплення мережевого трафіку.

– Забезпечити можливість зберігання захоплених даних.

– Введення системи фільтрації мережевого трафіку.

– Підтримка широкого спектру мережевих протоколів.

– Мінімалізація використання системних ресурсів для збереження оптимальної продуктивності.

– Програма повинна мати можливість відновити зібрані дані, які розбиватимуться по мережевим пакетам.

Нефункціональні вимоги:

– Забезпечення ефективності роботи програми, вона має працювати швидко, навіть при обробці великих обсягів даних.

– Реалізація зручного та простого графічного інтерфейсу для користувачів.

– Забезпечення цілісності зібраних даних з мережі.

– Сумістність програмного засобу при роботі на різних операційних системах.

– Програма має бути гнучкою до впровадження оновлень.

Було визначено основні завдання для подальшої розробки програмного засобу. Ці завдання включають реалізацію функціональності захоплення та аналізу трафіку, створення зручного графічного інтерфейсу та модулів візуалізації, тестування та валідацію. Підготовлена чітка основу для подальшої розробки програмного засобу та визначані цілі, які слід враховувати для створення високоефективного та зручного у використанні інструменту для

аналізу мережі. Отже, аналізуючи результати порівняння та складання вимог, можна підбити підсумки. Важливим етапом є розробка ефективного модулю захоплення, який враховує специфіку різних видів пакетів та протоколів. Це є ключовим аспектом для точного та повного аналізу мережевого трафіку в реальному часі. Забезпечення системи фільтрації є критичним, оскільки користувачам необхідно мати можливість вибору та налаштування критеріїв відбору для отримання точних результатів. Створення зручного графічного інтерфейсу та модулів візуалізації є не менш важливим завданням. Це допоможе користувачам легко орієнтуватися в обсязі зібраних даних, робити аналіз та приймати відповідальні рішення. Гнучкість до впровадження оновлень визначає необхідність створення системи, яка легко адаптується до змін у середовищі та нових вимог користувачів. Тестування та валідація грають ключову роль у впевненій роботі програмного засобу. Важливо провести комплексні тести для перевірки функціональності, ефективності та безпеки. Валідація допоможе переконатися в тому, що програмний засіб відповідає встановленим стандартам та вимогам.

РОЗДІЛ 2 ПРОЄКТУВАННЯ СИСТЕМИ

2.1 Опис технічних процесів та формалізація вимог

На етапі проєктування системи формується архітектура та функціонал аналізатора мережевого трафіку з використанням Scapy на мові Python. Це включає в себе визначення основних компонентів, способів їх взаємодії та розробку інтерфейсу користувача. Розглядаються можливі технічні рішення та вибір оптимальних інструментів для реалізації функціональності, забезпечуючи ефективну та надійну роботу програмного засобу.

У цьому підрозділі будуть досліджені функціональні вимоги та технічні процеси, які будуть відбуватися у системі аналізатора мережевого трафіку з використанням Scapy на мові Python. Цей підрозділ – це вихідна точка роботи, де ідеї та вимоги, розглянуті в першому розділі, набирають форму та конкретизуються, описуються детальніше, враховуючи усі ньюанси, та технічні особливості процесів, які потрібні для оптимального функціонування системи.

2.1.1 Захоплення мережевого трафіку

Процес, суть якого є захоплення мережевого трафіку також називається Sniffing, або сніфінг. Сніфінг – це перехоплення та перевірка мережевого трафіку для захоплення даних під час їх переміщення через комп'ютерну мережу. Процес захоплення трафіку видно на рисунку 2.1.

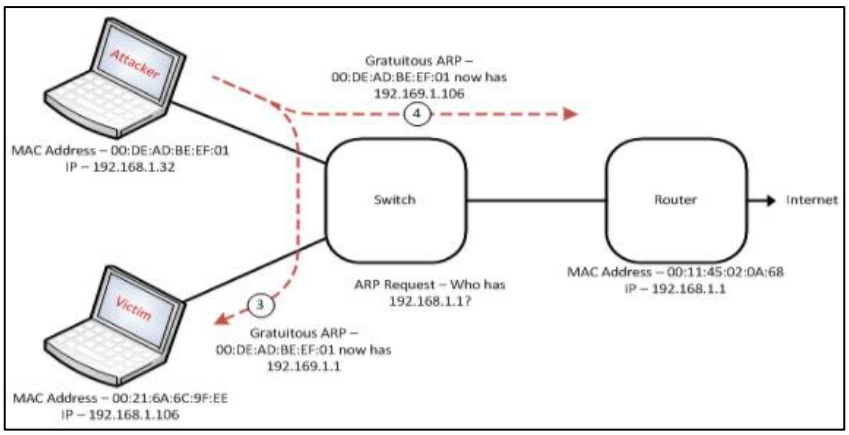


Рисунок 2.1 – Схема захоплення мережевого трафіку в мережі

Моніторинг трафіку є важливою сферою для управління, контролю, оптимізації та захисту систем. Сучасні корпоративні мережі стають все складнішими через велику кількість особистих пристроїв, розмаїття хмарних додатків, точок доступу та віртуальних серверів. Адміністраторам необхідно уважно керувати пропускною здатністю, забезпечувати якість обслуговування, використовувати балансувальники навантаження для підвищення доступності, створювати резервні копії та забезпечувати синхронізовану роботу компонентів у гібридному середовищі. Аналіз можна проводити на рівні окремих пакетів, в контексті роботи додатків або при діагностуванні складних систем.

Глибину аналізу мережевого трафіку можна поділити на рівні відповідно до моделі OSI (рисунок 2.2).

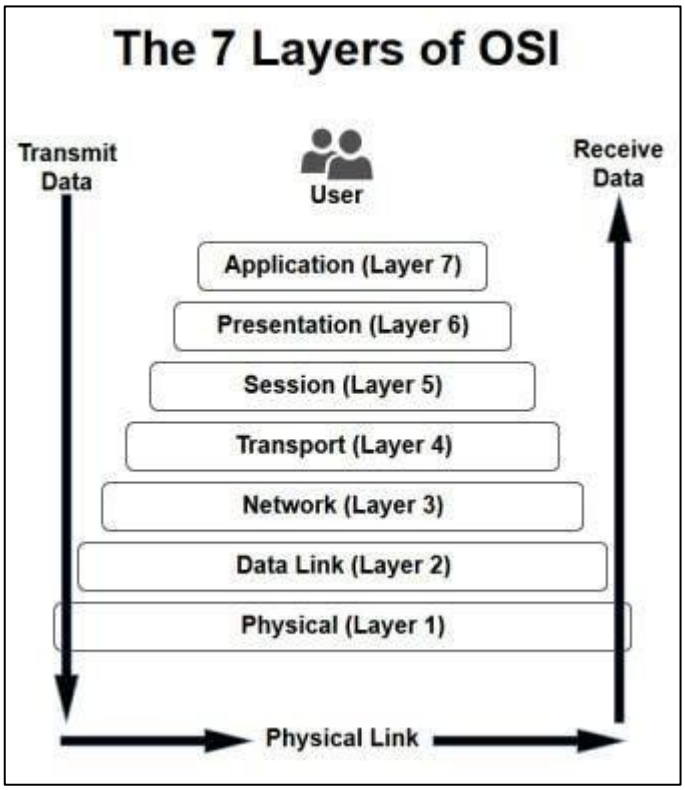


Рисунок 3.2 – Схема моделі OSI

Кожен мережевий пакет складається з двох основних компонентів: керуючої інформації та корисного навантаження. У цьому контексті термін "пакет" використовується для узагальнення таких понять, як фрейм, дейтаграма

або сегмент відповідних мережевих протоколів. Під час аналізу пакета виділяються заголовки протоколів, і проводиться аналіз значень полів у цих заголовках. Структура заголовка визначається специфікацією, тоді як корисне навантаження може містити дані будь-якої організації, хоча зазвичай це є пакет протоколу наступного, більш високого рівня, що вимагає визначення протоколу для подальшого розбору (рисунок 2.3).

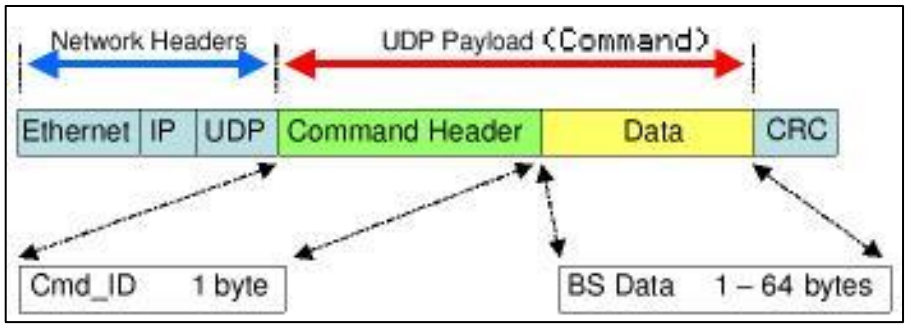


Рисунок 4.3 – Структура пакету

Згідно з моделлю OSI, заголовки мережевих протоколів утворюють стек і зазвичай розташовуються один за одним в природному порядку, знизу вгору. Проте при організації тунельних з'єднань цей порядок може бути порушений, наприклад, при передачі IPv4-пакетів (мережевий рівень) в рамках пакетів протоколу UDP (транспортний рівень). Тунельні протоколи дуже поширені і використовуються для організації віртуальних приватних мереж. У загальному випадку можлива будь-яка конфігурація тунелю, включаючи вкладення одного тунелю в інший.

Протоколи можуть бути класифіковані як зберіганням та без зберіганням стану. Для протоколів зі збереженням стану обов'язковою частиною є відповідний автомат станів. Під час реального часу аналізатор може зіткнутися з необмеженим зростанням кількості з'єднань, для яких потрібне збереження характеристик поточного стану. Тому аналізатор повинен ефективно управляти розподілом доступних ресурсів.

2.1.2 Фільтрація трафіку

Процес фільтрації трафіку в програмі для аналізу мережевого трафіку є критично важливим етапом, який дозволяє здійснювати відбір певних мережевих пакетів з усього потоку даних відповідно до заданих критеріїв. Цей процес виконується за допомогою встановлення різних умов та параметрів, які визначають, які пакети повинні бути включені до аналізу, а які - відфільтровані.

Один із ключових аспектів фільтрації трафіку - це визначення критеріїв фільтрації. Ці критерії можуть включати різноманітні параметри, такі як IP-адреси відправників та отримувачів, порти, протоколи, типи пакетів, розмір пакетів тощо. Наприклад, користувач може бажати проаналізувати лише HTTP-запити до певного веб-сервера, тому він встановить фільтр для відбору пакетів з IP-адресою веб-сервера та портом 80.

Після встановлення критеріїв фільтрації програма проводить аналіз кожного мережевого пакета, що проходить через мережевий адаптер. Пакети, що відповідають заданим критеріям, включаються до вихідного набору даних, тоді як ті, які не відповідають умовам, відкидаються або ігноруються (рисунок 2.4).

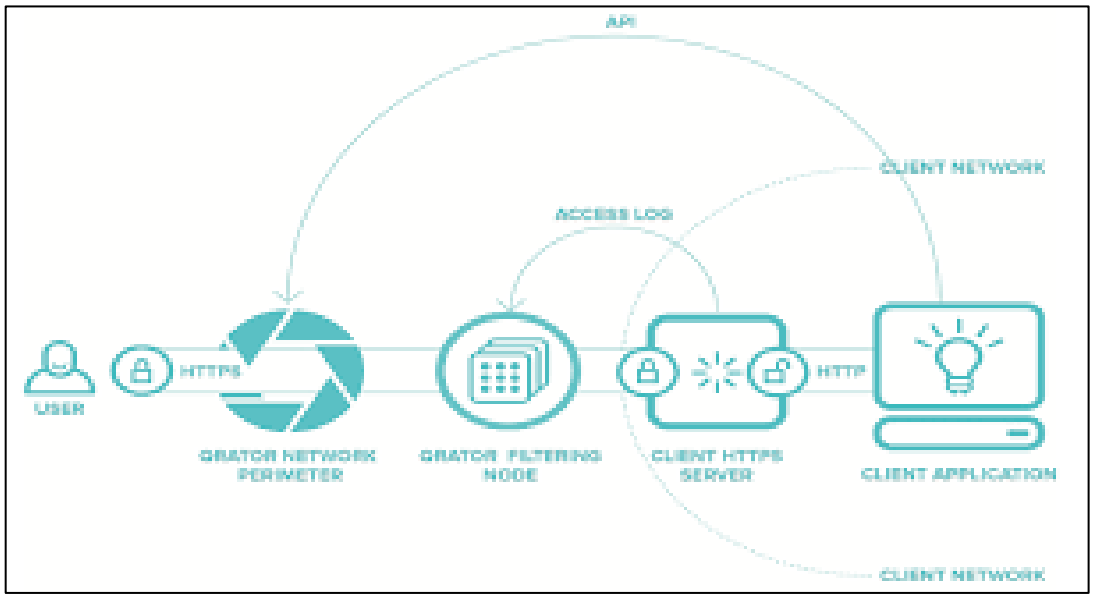


Рисунок 5.4 – Схема фільтрації трафіку

Після фільтрації отримані дані можуть бути використані для подальшого аналізу. Наприклад, можуть бути побудовані графіки, статистика або звіти щодо

певних аспектів мережевого трафіку, що дозволяє виявити аномалії, проблеми з мережевим з'єднанням або потенційні загрози безпеці. Такий підхід допомагає адміністраторам мережі та інженерам у виявленні, аналізі та вирішенні проблем, пов'язаних з мережевим трафіком, та забезпечує більшу ефективність і надійність мережі.

2.1.3 Аналіз мережевих протоколів

Аналіз мережевих протоколів - це ключова складова аналізатора мережевого трафіку, яка визначає типи протоколів, які використовуються в мережі, і дозволяє аналізувати їхню взаємодію та динаміку передачі даних. Мережеві протоколи можуть бути різними за своєю природою і призначенням, і включають в себе такі, як TCP, UDP, HTTP, DNS, ICMP, ARP тощо (рисунок 2.5).

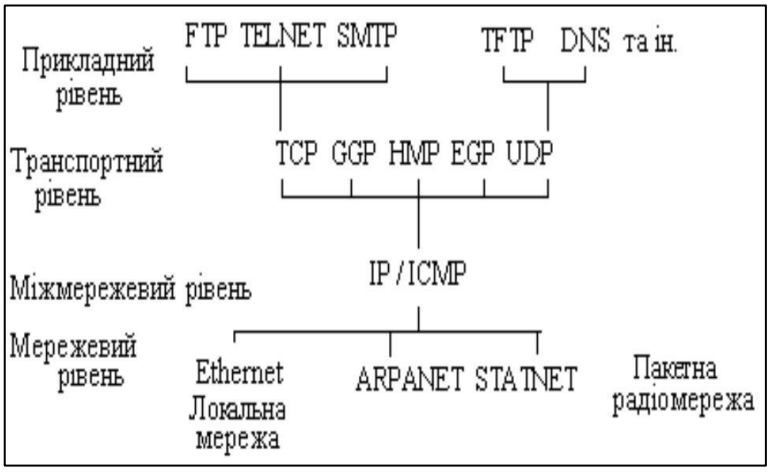


Рисунок 6.5 – Схема, яка містить протоколи та рівні, до яких вони відносяться

Протокол TCP (Transmission Control Protocol) є надійним з'єднанням, орієнтованим на поток, і забезпечує доставку даних у порядку, контроль передачі та відновлення пакетів в разі втрати. У порівнянні з цим, UDP (User Datagram Protocol) працює без встановлення з'єднання і не забезпечує гарантовану доставку, що робить його швидшим, але менш надійним протоколом.

HTTP (Hypertext Transfer Protocol) використовується для передачі веб-сторінок та інших ресурсів через Інтернет. Він базується на моделі клієнт-сервер і використовує запити та відповіді для обміну інформацією.

DNS (Domain Name System) використовується для перетворення доменних імен в IP-адреси та забезпечення розподіленої бази даних для вирішення доменних імен. ICMP (Internet Control Message Protocol) використовується для передачі повідомлень про помилки та інших діагностичних інформаційних повідомлень між мережевими пристроями.

ARP (Address Resolution Protocol) використовується для вирішення фізичних адрес пристроїв в мережі, асоціюючи їх з їхніми IP-адресами.

Аналіз мережевих протоколів дозволяє розуміти, як працює мережа, виявляти проблеми, виконувати діагностику та впроваджувати відповідні заходи для оптимізації та забезпечення надійності мережевого трафіку.

2.1.4 Збереження результатів аналізу мережевого трафіку

Після того, як модуль аналізу завершив обробку мережевих пакетів і виконав аналіз мережевих протоколів, отримані результати можуть бути збережені для подальшого використання або аналізу. Процес збереження результатів аналізу включає наступні етапи:

– Вибір формату збереження.

Після аналізу користувач може вибрати формат, в якому будуть збережені результати. Це може бути текстовий файл для зручного перегляду, файл бази даних для подальшого структурованого аналізу або спеціалізований формат, який підтримується аналізатором мережевого трафіку.

– Задання параметрів збереження.

Користувач може визначити додаткові параметри збереження, такі як шлях до файлу, інтервал збереження, обсяг даних, який слід зберегти тощо. Це дозволяє налаштувати процес збереження згідно з потребами користувача і обмеженнями системи.

– Збереження даних.

Після визначення параметрів збереження результати аналізу зберігаються в обраному форматі та місці. Цей процес може включати створення або оновлення файлу даних або запис до бази даних.

– Перевірка цілісності даних.

Після завершення збереження може бути виконана перевірка цілісності збережених даних. Це важливий крок, щоб переконатися, що дані збереглися коректно та повністю і можуть бути використані в подальшому.

2.1.5 Діаграма послідовностей

Діаграма послідовностей - це тип діаграми в уніфікованій моделі мови (UML), який використовується для моделювання послідовності взаємодії між об'єктами або учасниками системи в певному сценарії використання. Вона дозволяє показати, які дії виконуються в системі і які повідомлення передаються між учасниками у конкретних ситуаціях або процесах (рис. 2.6).

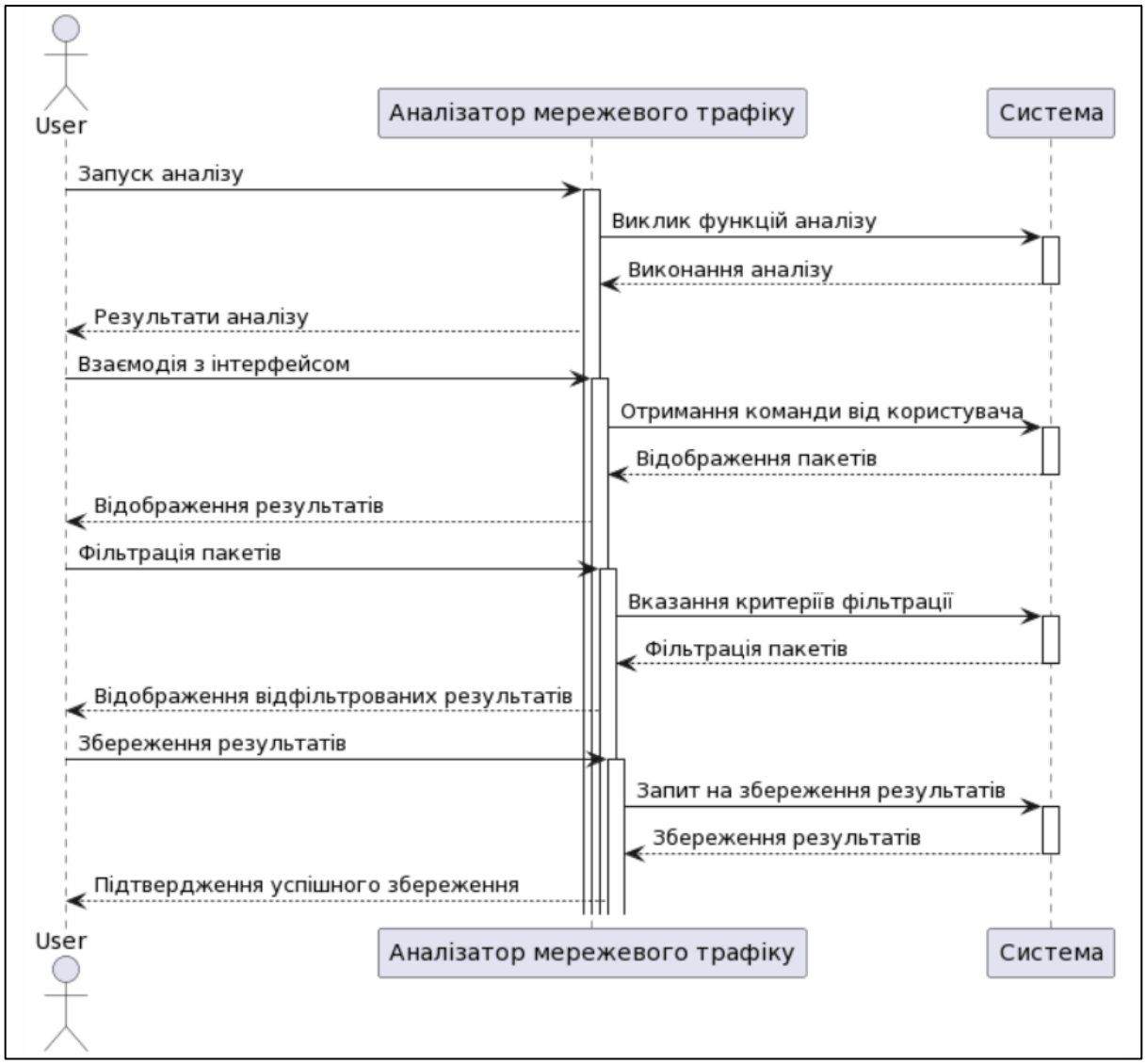


Рисунок 7.6 – Діаграма послідовностей

2.1.6 Діаграма станів

Діаграма станів (State Diagram) - це вид діаграми UML, який використовується для візуалізації різних станів, через які може проходити об'єкт або система під час свого життєвого циклу (рисунок 2.7).

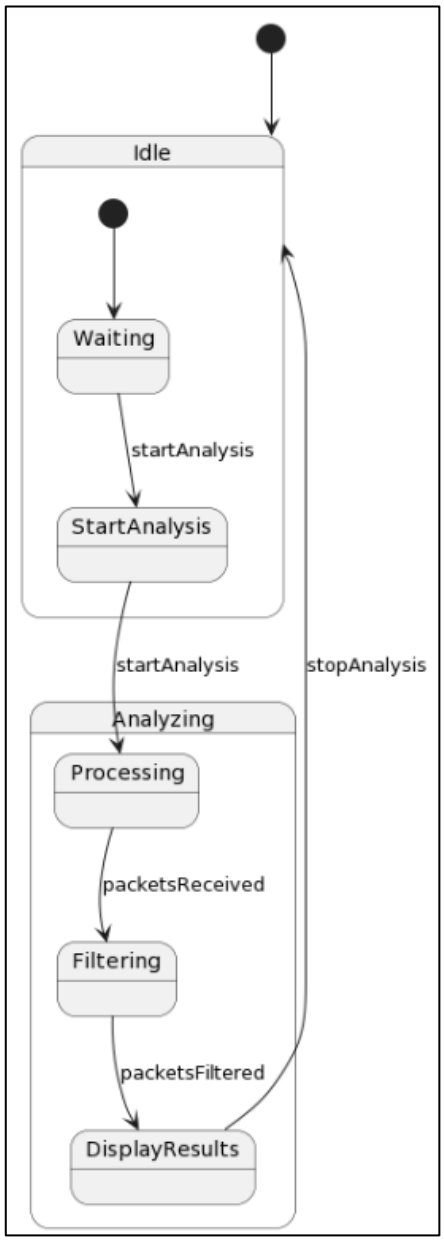


Рисунок 8.7 – Діаграма станів

Вона складається зі станів, подій та переходів між цими станами.

2.2 Аналіз технологій реалізації

Перед втіленням проєкту аналізатора мережевого трафіку на базі бібліотеки Scapy у мові програмування Python необхідно детально проаналізувати доступні технології, які можна використати для реалізації цієї системи.

Цей аналіз включає оцінку можливостей мови програмування Python, зокрема її зручності, продуктивності та розширюваності для створення програмного забезпечення такого типу.

Також важливо детально розглянути можливості бібліотеки Scapy, визначити її переваги та обмеження, а також способи її ефективного використання для аналізу мережевого трафіку. Додатково, слід розглянути можливість використання бібліотеки threading для підвищення швидкодії обробки трафіку шляхом використання паралельних потоків виконання.

2.2.1 Мова програмнування Python

Python – це високорівнева, інтерпретована мова програмування загального призначення з простим синтаксисом та сильним фокусом на читабельності коду. Вона надає широкі можливості у розробці різноманітних програм, включаючи веб-додатки, наукові обчислення, автоматизацію завдань, обробку даних, штучний інтелект та інше.

Мова має простий і зрозумілий синтаксис, що дозволяє легко вивчати і швидко створювати програми. Вона підтримує різні парадигми програмування, включаючи процедурне, об'єктно-орієнтоване, функціональне та структурне програмування. Python є вільним програмним забезпеченням, що означає, що його можна використовувати, змінювати та поширювати безкоштовно.

Він має велику та активну спільноту розробників, яка підтримує стандарти якості, розробляє різноманітні бібліотеки та фреймворки, що полегшують роботу з мовою та забезпечують її актуальність у світі програмування.

2.2.2 Бібліотека Scapy

Scapy – це бібліотека на Python, яка дозволяє користувачеві надсилати, захоплювати, аналізувати та підробляти мережеві пакети. Ця можливість дозволяє створювати інструменти, які можуть досліджувати, сканувати або атакувати мережі [1]. Іншими словами, Scapy – це потужна інтерактивна програма для обробки пакетів. Він здатний підробляти або декодувати пакети з великою кількістю протоколів, надсилати їх по дроту, перехоплювати, зіставляти запити та відповіді та багато іншого. Scapy (рисунок 2.8) може легко впоратися з більшістю класичних завдань, таких як сканування, трасування, зондування, модульні тести, атаки або виявлення мережі. Він може замінити hping, arpspoof, arp-sk, arping, p0f і навіть деякі частини Nmap, tcpdump і tshark.

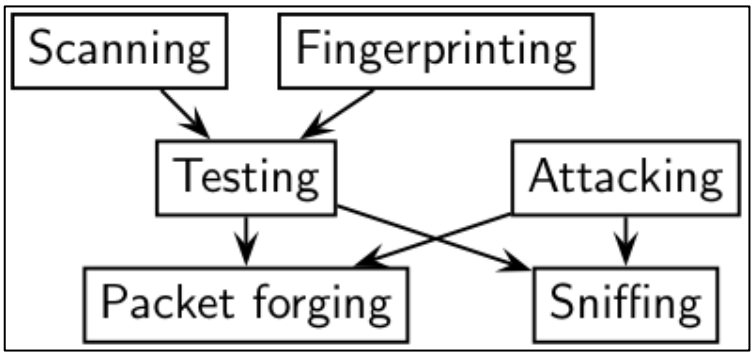


Рисунок 9.8 – Ієрархічна схема захоплювання трафіку в Scapy

Scapy також дуже добре виконує багато інших специфічних завдань, з якими більшість інших інструментів не може впоратися, як-от надсилання недійсних кадрів, введення власних кадрів 802.11, комбінування методів (перестрибування VLAN та отруєння кешу ARP, декодування VOIP на каналі, зашифрованому WEP) тощо.

Ідея проста. Scapy в основному робить дві речі: надсилає пакети та отримує відповіді. Ви визначаєте набір пакетів, він надсилає їх, отримує відповіді, зіставляє запити з відповідями та повертає список пар пакетів (запит, відповідь) і список невідповідних пакетів. Це має велику перевагу перед такими

інструментами, як Nmap або hping, оскільки відповідь не зводиться до відкритих, закритих або відфільтрованих, а є цілим пакетом. Крім цього, можна створювати функції більш високого рівня.

Наприклад, такий, який виконує traceroutes і надає в результаті лише початковий TTL запиту та вихідну IP-адресу відповіді. Такий, який перевіряє всю мережу та видає список машин, що відповідають. Той, який виконує сканування портів і повертає звіт LaTeX.

Інші інструменти дотримуються парадигми програми, яку ви запускаєте з оболонки.

У результаті виходить жахливий синтаксис для опису пакета. Для цих інструментів прийняте рішення використовує вищий, але менш потужний опис у формі сценаріїв, які придумав автор інструменту.

Наприклад, сканеру портів потрібно надати лише IP-адресу, щоб запустити сценарій сканування портів. Навіть якщо сценарій трохи змінено, ви все одно застрягли на скануванні портів.

Парадигма Scapy полягає в тому, щоб запропонувати доменно-специфічну мову (DSL), яка забезпечує потужний і швидкий опис будь-якого типу пакету.

Використання синтаксису Python та інтерпретатора Python як синтаксису та інтерпретатора DSL має багато переваг: немає потреби писати окремий інтерпретатор, користувачам не потрібно вивчати ще одну мову, і вони виграють від повного, стислого та дуже потужна мова. Scapy дає змогу користувачеві описати пакет або набір пакетів як шари, накладені один на одного.

Поля кожного шару мають корисні значення за замовчуванням, які можна перевантажувати. Scapy не зобов’язує користувача використовувати заздалегідь визначені методи чи шаблони (рисунок 2.9).

Це полегшує необхідність написання нового інструменту кожного разу, коли потрібен інший сценарій. У мові C для опису пакета може знадобитися в середньому 60 рядків.

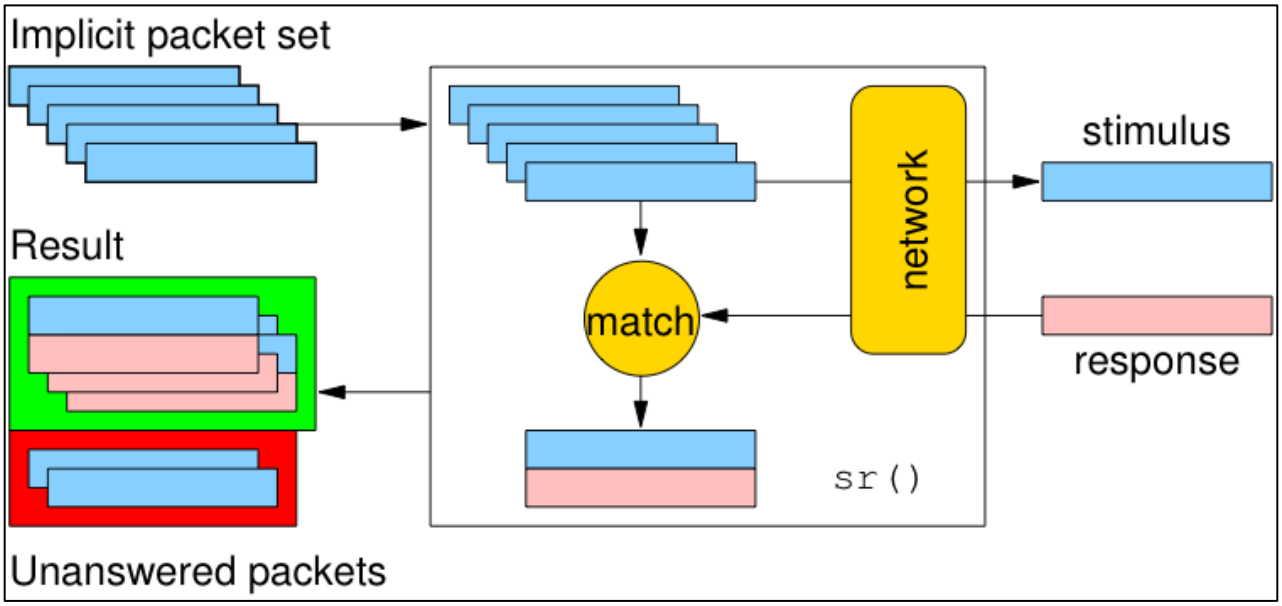


Рисунок 10.9 – Концепт поділу трафіку на інтернет-пакети в Scapy

За допомогою Scapy пакети, які потрібно надіслати, можна описати лише в одному рядку з іншим рядком для друку результату. 90% інструментів зондування мережі можна переписати в два рядки Scapy.

2.2.3 Бібліотека threading

Модуль threading в мові програмування Python є потужним інструментом для роботи з паралельністю та багатопотоковим програмуванням [2].

Він дозволяє створювати, керувати та взаємодіяти з потоками в програмі, що дозволяє виконувати кілька завдань одночасно та ефективно використовувати ресурси комп'ютера (рисунок 2.10).

Основною перевагою модуля threading є можливість створювати і керувати потоками в межах одного процесу. Це означає, що програми можуть виконувати кілька завдань одночасно, що підвищує ефективність програм та зменшує час виконання.

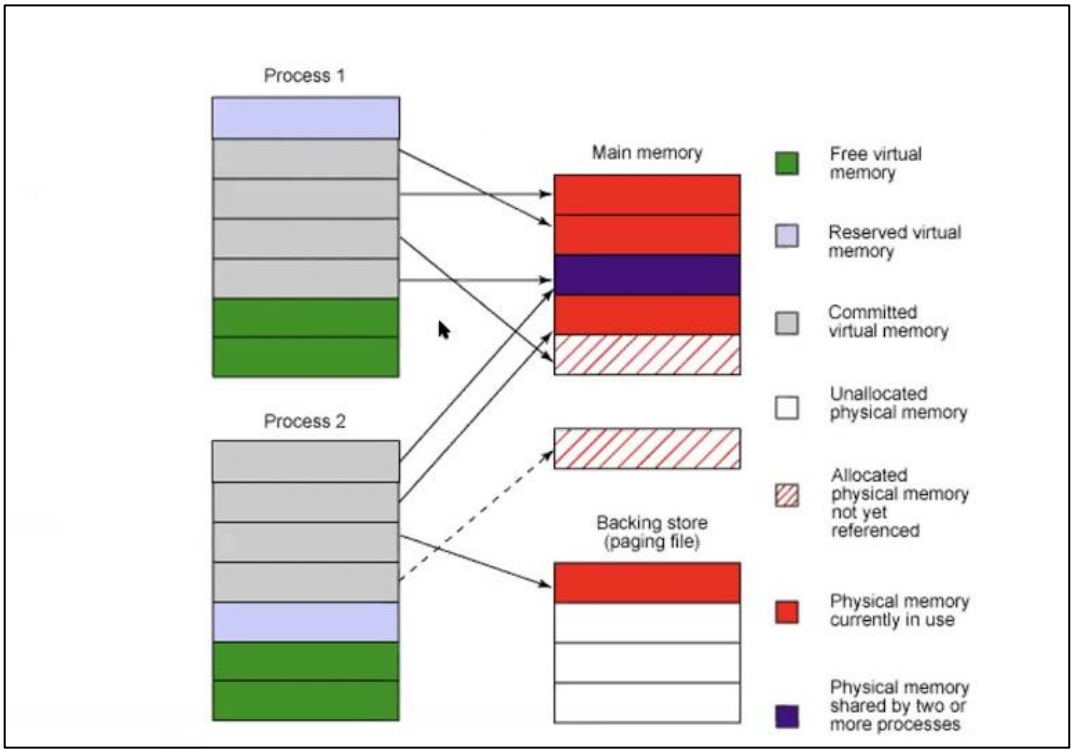


Рисунок 11.10 – Схема використання багатопроцесності та багатопотоковості в програмах

Модуль threading надає інструменти для створення нових потоків, керування їх виконанням, синхронізації взаємодії між ними та управління життєвим циклом потоків. Він включає в себе класи, такі як Thread, Lock, Event, Semaphore та інші, які дозволяють розробникам створювати потокобезпечні програми та уникати проблем конкурентної взаємодії [4].

Одним з основних принципів роботи з потоками є використання механізмів синхронізації, таких як блокування (Locks) та бар'єри (Barriers), щоб уникнути конфліктів при доступі до спільних ресурсів. Також важливо правильно керувати життєвим циклом потоків, включаючи їх створення, запуск, призупинення та завершення.

Загалом, модуль threading є потужним інструментом для роботи з паралельністю в мові програмування Python, що дозволяє розробникам створювати ефективні та швидкодійні програми, які виконуються в багатопотоковому середовищі.

2.3 Проєктування інтерфейсу

У цьому підрозділі буде розглянуто проєктування інтерфейсу для програмного засобу, який буде представлений графічно та простим у використанні. Головна мета – це створити інтуїтивно зрозумілий інтерфейс, який дозволить користувачам легко взаємодіяти з програмою, виконувати аналіз трафіку та отримувати необхідну інформацію без зайвих зусиль. У цьому контексті буде розглянуто основні принципи дизайну інтерфейсу, враховуючи зручність використання та ефективність сприйняття користувачем.

2.3.1 Діаграма варіантів використання

Діаграма варіантів використання (Use Case Diagram) - це візуальний інструмент моделювання, який показує, як користувачі взаємодіють з системою шляхом визначення різних варіантів використання або функціональних можливостей системи. Вона допомагає визначити основні функціональність системи та її взаємодію з користувачами або іншими системами.

У контексті програми "Аналізатор мережевого трафіку" діаграма варіантів використання допомагає уявити основні можливості, які надає цей інструмент користувачеві. Кожен варіант використання представляє собою конкретну функціональність програми, яка може бути викликана користувачем. Наприклад, це можуть бути дії, такі як запуск аналізу мережі, зупинка аналізу, відображення пакетів, фільтрація пакетів та збереження результатів (рисунок 2.11).

Рисунок 12.11 – Діаграма варіантів використання

Дії, які відбуваються у кожному з варіантів використання:

– Запуск аналізу. Користувач запускає аналіз мережі, взаємодіючи з системою.

Відповідь програми: Система отримує запит на виклик функцій аналізу.

зупинка аналізу.

– Користувач зупиняє аналіз мережі.

Відповідь програми: Система отримує запит на виклик функції зупинки аналізу.

– Відображення пакетів. Користувач взаємодіє з системою, яка відображає пакети, які проходять через мережу.

Відповідь програми: Система отримує запит на відображення пакетів.

– Фільтрація пакетів. Користувач взаємодіє з системою, вказуючи критерії фільтрації пакетів.

Відповідь програми: Система отримує запит на фільтрацію пакетів.

– Збереження результатів. Користувач зберігає результати аналізу мережі для подальшого використання або архівування.

Відповідь програми: Система отримує запит на збереження результатів.

2.3.2 Макети вікон

У цьому підпункті розглядаються макети вікон для графічного інтерфейсу програми, які відображаються користувачеві під час роботи з аналізатором мережевого трафіку (рисунок 2.12).

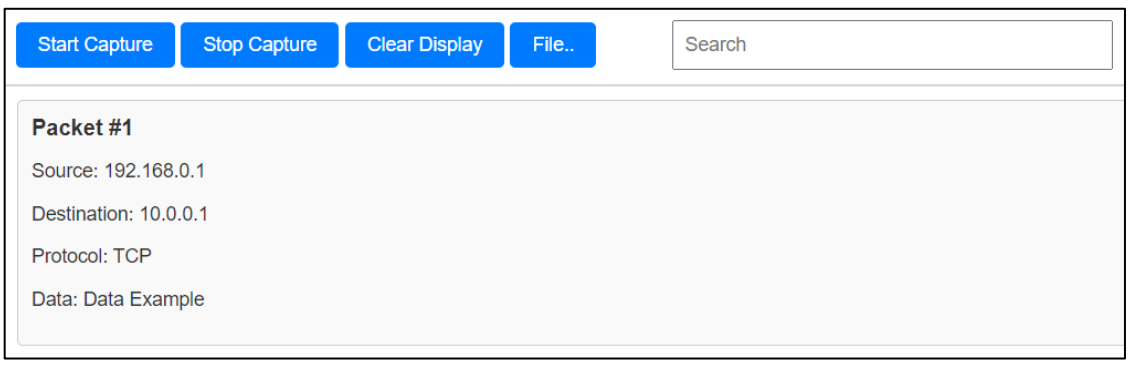


Рисунок 13.12 – Макет головного вікна

В головному вікні застосунку має бути відображені кнопки для використання основного функціоналу програми. На цьому макеті є:

– Кнопка «Start Capture», або ж «Почати захоплення», за допомогою якої можна почати захоплення трафіку та збір даних з мережі.

– Кнопка «Stop Capture», або ж «Зупинити захоплення», яка негайно зупиняє збір трафіку.

– Кнопка «Clear Display», або ж «Очищення екрану», що стане у нагоді, коли дані не потрібно буде зберігати і їх можна буде очистити.

– Кнопка «File…», або ж «Файл…», після натиснення якої з’явиться контекстне меню, в якому будуть функції відкриття попередньо збережених файлів з інтернет-пакетами та збереження даних і утворення файлів.

– Поле для вводу тексту з підписом «Search», або ж «Пошук», де користувач зможе застосовувати свої фільтри для пошуку пакетів.

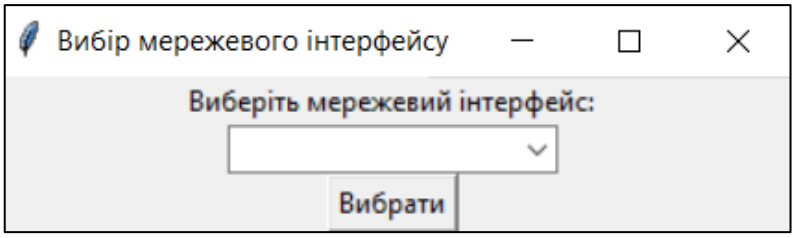


Рисунок 14.13 – Макет вікна з вибором мережевого інтерфейсу

В вікні з вибором інтерфейсу (рис. 2.13) є кнопка підтвердження вибору, а також контекстне меню, де можна вибрати мережевий інтерфейс (рис. 2.14).

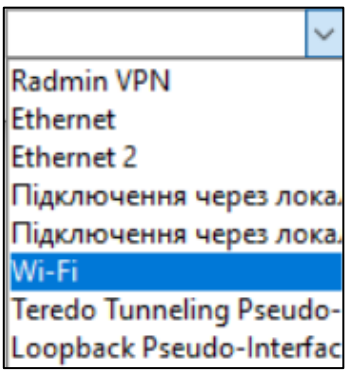


Рисунок 15.14 – Макет контекстного меню з вибором мережевого інтерфейсу

Цей вибір забезпечує точне та зручне використання аналізатора, тому що користувачу буде наперед відомо яку саме мережу він прослуховує.

Було проведено детальне визначення та уточнення вимог до аналізатора мережевого трафіку на основі Scapy з використанням мови програмування Python. Цей етап дозволив детальніше проаналізувати потреби користувачів і виокремити ключові функціональні та технічні вимоги до системи.

Результатом цього аналізу стали конкретизовані вимоги, які дозволяють чітко визначити завдання та напрямки подальшої роботи. Були внесені детальні розшифрування та уточнення до вимог, що дозволило визначити більш конкретні критерії успішності проєкту.

Таким чином, формалізація вимог стала важливим етапом у процесі розробки системи, який додав більше конкретики до поставлених вимог.

РОЗДІЛ 3 РЕАЛІЗАЦІЯ ТА НАЛАГОДЖЕННЯ ПРОГРАМНОГО ЗАСОБУ

3.1 Розробка інтерфейсу користувача

Проєктування інтерфейсу для програмного засобу аналізу мережевого трафіку є важливою частиною проєкту, оскільки зручність використання та ефективність інтерфейсу визначають продуктивність і задоволення користувачів. Інтерфейс повинен бути інтуїтивно зрозумілим, забезпечувати легкий доступ до основних функцій програми та надавати чітку і зрозумілу інформацію про аналізований трафік. Основна мета інтерфейсу програмного засобу для аналізу мережевого трафіку - забезпечити користувачам зручний інструмент для моніторингу та аналізу мережевої активності в реальному часі.

Інтерфейс повинен надавати такі можливості:

- Вибір мережевого інтерфейсу для захоплення трафіку.
- Запуск, зупинка та відновлення процесу захоплення трафіку.
- Відображення детальної інформації про кожен захоплений пакет.
- Збереження та імпорт даних аналізу.
- Фільтрація та сортування даних для зручного перегляду.

3.1.1 Структура інтерфейсу

Інтерфейс програми складається з кількох основних компонентів, кожен з яких буде містити власні функції для взаємодії з програмою. Одне з таких це головне вікно програми, яке є основним робочим простором користувача. В ньому міститься меню, панель інструментів та безпосередньо поле з відображеними даними пакетів та інформаційні панелі над ними для точного відображення їх складових частих.

Меню містить два розділи: «Файл» та «Захоплення». Розділ «Файл» створений заради взаємодії з даними на рівні файлової системи та має два пункти в контекстному меню, перший називається «Зберегти дані», натиснувши на який,

можна буде зберегти попередньо захоплений трафік, а другий, «Імпортувати дані», передбачає функцію зчитування даних з вже створеного файлу. Розділ «Захоплення» відповідатиме за керування захопленням трафіку користувачем, та матиме в контекстному меню три пункти, які відповідатимуть за початок захоплення трафіку, його зупинку, та відновлення захоплення.

Поле для відображення даних пакетів є одним з найголовніших частин інтерфейсу. Пакети в цьому полі подані в якості таблиці, щоб структуризувати всю інформацію і зробити її більш зручнішою та легшою для сприйняття людським оком. Таблиця поділена на такі колонки:

– Протокол.
– Джерело MAC.
– Джерело IP.
– Призначення IP.

Розподілення по цим колонкам забезпечує швидкий доступ до інформації про кожен пакет.

3.1.2 Програмна реалізація компонентів інтерфейсу

Для створення інтерфейсу було обрано вбудовану бібліотеку Tkinter в мові програмування Python, яка забезпечує швидкий та простий спосіб взаємодії в віконних додатках.

Для початку роботи з додатком, користувач повинен вибрати на якому мережевому інтерфейсі буде здійснюватись захоплення трафіку. Для цього буде використовуватись вікно діалогу, яке містить список доступних мережевих інтерфейсів, які отримані за допомогою бібліотеки psutil [5].

Програмний код для вікна вибору мережевого інтерфейсу:

```
    def open_interface_selection(self):
        if self.interface_window and
self.interface_window.winfo_exists():
            return
        self.interface_window = tk.Toplevel()
```

```
        self.interface_window.title("Вибір мережевого інтерфейсу")
        available_interfaces = [interface for interface, addrs in
psutil.net_if_addrs().items()]
        interface_label = tk.Label(self.interface_window,
text="Виберіть мережевий інтерфейс:")
        interface_label.pack()
        interface_combobox = ttk.Combobox(self.interface_window,
values=available_interfaces, state="readonly")
        interface_combobox.pack()
        def select_interface():
            self.capture_interface = interface_combobox.get()
            self.interface_window.destroy()
        select_button = tk.Button(self.interface_window,
text="Вибрати", command=select_interface)
        select_button.pack()
```

Для управління процесом захоплення трафіку створені такі функції, як «start_analysis», «stop_analysis» та «resume_analisys», які запускають, зупиняють та відновлюють захоплення трафіку відповідно.

Програмний код для управління захопленням трафіку:

```
    def start_analysis(self):
        if not self.capture_interface:
            self.add_to_text("Не вибрано мережевий інтерфейс.\n")
            self.open_interface_selection()
            return
        def analyze_packet(packet):
            self.sniffing_paused.wait()
            self.analyze_packet_details(packet)
        self.sniffing_thread =
threading.Thread(target=self.sniff_packets, args=(analyze_packet,))
        self.sniffing_thread.start()
    def stop_analysis(self):
        if self.sniffing_thread:
            self.sniffing_paused.clear()
    def resume_analysis(self):
        if self.sniffing_thread:
            self.sniffing_paused.set()
```

Відображення даних про захоплені пакети використовують таблицю з бібліотеки tkinter під назвою «Treeview», яка дозволяє зручно розташовувати дані. Програмний код для реалізації відображення даних:

```
columns = ["Протокол", "Джерело MAC", "Призначення MAC", "Джерело IP", "Призначення IP"]
analyzer.treeview = ttk.Treeview(tree_frame, columns=columns, show="headings")

for col in columns:
    analyzer.treeview.heading(col, text=col)
    analyzer.treeview.column(col, anchor='center', stretch=True)

vsb = ttk.Scrollbar(tree_frame, orient="vertical", command=analyzer.treeview.yview)
vsb.pack(side='right', fill='y')
hsb = ttk.Scrollbar(tree_frame, orient="horizontal", command=analyzer.treeview.xview)
hsb.pack(side='bottom', fill='x')

analyzer.treeview.configure(yscrollcommand=vsb.set, xscrollcommand=hsb.set)
analyzer.treeview.pack(fill=tk.BOTH, expand=True)
```

Для створення самих вікон застосовуються також і інші методи бібліотеки tkinter, такі як «window», «menu», та інші. Решта коду, який відповідає за графічний інтерфейс:

```
window = tk.Tk()
window.title("Аналізатор мережевого трафіку")
# Налаштування сітки для основного вікна
window.rowconfigure(1, weight=1)
window.columnconfigure(0, weight=1)
# Створення меню
menu_bar = tk.Menu(window)
file_menu = tk.Menu(menu_bar, tearoff=0)
file_menu.add_command(label="Зберегти дані", command=analyzer.save_data)
```

```
        file_menu.add_command(label="Імпортувати дані",
command=analyzer.import_data)
        menu_bar.add_cascade(label="Файл", menu=file_menu)
        capture_menu = tk.Menu(menu_bar, tearoff=0)
        capture_menu.add_command(label="Почати аналіз",
command=analyzer.start_analysis)
        capture_menu.add_command(label="Припинити аналіз",
command=analyzer.stop_analysis)
        capture_menu.add_command(label="Відновити аналіз",
command=analyzer.resume_analysis)
        menu_bar.add_cascade(label="Захоплення", menu=capture_menu)

        window.config(menu=menu_bar)
```

Для створення заголовків колонок і текстового поля для відображення результатів розроблено наступний код:

```
        tree_frame = tk.Frame(window)
        tree_frame.grid(row=1, column=0, sticky='nsew')

        columns = ["Протокол", "Джерело MAC", "Призначення MAC", "Джерело
IP", "Призначення IP"]
        analyzer.treeview = ttk.Treeview(tree_frame, columns=columns,
show="headings")

        for col in columns:
            analyzer.treeview.heading(col, text=col)
            analyzer.treeview.column(col, anchor='center', stretch=True)

        # Додавання горизонтального і вертикального скролбарів
        vsb = ttk.Scrollbar(tree_frame, orient="vertical",
command=analyzer.treeview.yview)
        vsb.pack(side='right', fill='y')
        hsb = ttk.Scrollbar(tree_frame, orient="horizontal",
command=analyzer.treeview.xview)
        hsb.pack(side='bottom', fill='x')

        analyzer.treeview.configure(yscrollcommand=vsb.set,
xscrollcommand=hsb.set)
        analyzer.treeview.pack(fill=tk.BOTH, expand=True)
```

```
    # Кнопки для вибору інтерфейсу і запуску аналізу
    button_frame = tk.Frame(window)
    button_frame.grid(row=0, column=0, pady=10, sticky='ew')

    select_interface_button = tk.Button(button_frame, text="Вибрати
мережевий інтерфейс", command=analyzer.open_interface_selection)
    select_interface_button.pack(side=tk.LEFT, padx=5)

    window.mainloop()
```

Проєктування інтерфейсу для програмного засобу аналізу мережевого трафіку є важливим етапом, який впливає на загальну ефективність та зручність використання програми. Вибраний підхід та технології забезпечують інтуїтивний, функціональний та гнучкий інтерфейс, що відповідає вимогам користувачів та завданням аналізу мережевого трафіку.

3.2 Реалізація основних функцій програмного засобу

Невід'ємною частиною роботи над проєктом є реалізація функцій, які є логічною та основною складовою програми. Основні функції включають захоплення трафіку, аналіз захоплених пакетів, а також збереження та імпорт даних.

Захоплення трафіку здійснюється за допомогою бібліотеки Scapy. Функція «sniff_packets» відповідає за захоплення пакетів на вибраному мережевому інтерфейсі та передачу їх для аналізу.

Програмний код розроблений для цього завдання :

```
    def sniff_packets(self, analyze_packet):
        sniff(prn=analyze_packet, store=0,
iface=self.capture_interface, filter="ip")
```

Аналіз захоплених пакетів включає визначення типу протоколу та витягнення основної інформації про пакет. Функція «analyze_packet_details» виконує цей аналіз та передає результати для відображення у таблиці.

Наведемо програмний код для відображення у таблиці:

```
def analyze_packet_details(self, packet):
    protocol = "Unknown"
    if IP in packet:
        if TCP in packet:
            protocol = "TCP"
        elif UDP in packet:
            protocol = "UDP"
        else:
            protocol = packet[IP].proto
    elif Ether in packet:
        protocol = "Ethernet"

    data = {
        "protocol": protocol,
        "src_mac": packet[Ether].src if Ether in packet else "",
        "dst_mac": packet[Ether].dst if Ether in packet else "",
        "src_ip": packet[IP].src if IP in packet else "",
        "dst_ip": packet[IP].dst if IP in packet else ""
    }
    self.add_to_table(data)
```

Функції збереження та імпорту даних дозволяють користувачу зберігати результати аналізу у форматі CSV та PCAP, а також завантажувати їх для подальшого аналізу.

Наведемо програмний код розроблений для збереження даних:

```
     def save_data(self):
         file_path =
filedialog.asksaveasfilename(defaultextension=".csv", filetypes=[("CSV
files", "*.csv"), ("All files", "*.*")])
         if file_path:
             with open(file_path, mode='w', newline='') as file:
                 writer = csv.writer(file)
                 writer.writerow(["Протокол", "Джерело MAC",
"Призначення MAC", "Джерело IP", "Призначення IP"])
                 for row in self.treeview.get_children():
                     writer.writerow(self.treeview.item(row)['values'])
             pcap_file_path = file_path.replace(".csv", ".pcap")
             wrpcap(pcap_file_path, self.captured_packets)
```

Для імпорту даних розроблено наступний програмний код:

```
    def import_data(self):
    file_path = filedialog.askopenfilename(defaultextension=".csv",
filetypes=[("CSV files", "*.csv"), ("All files", "*.*")])
    if file_path:
        with open(file_path, mode='r', newline='') as file:
            reader = csv.reader(file)
            next(reader)  # Skip the header row
            for row in reader:
                self.add_to_table({
                    "protocol": row[0],
                    "src_mac": row[1],
                    "dst_mac": row[2],
                    "src_ip": row[3],
                    "dst_ip": row[4]
                })
        pcap_file_path = file_path.replace(".csv", ".pcap")
        self.captured_packets = rdpcap(pcap_file_path)
        for packet in self.captured_packets:
            self.analyze_packet_details(packet)
```

При створенні функцій збереження та імпорту була зосереджена увага на тому, щоб файли, які б зберігала програма, могли взаємодіяти з іншими аналізаторами мережевого трафіку, таких як найпопулярніший Wireshark, тому що в користувача може виникнути потреба в іншому функціоналі.

Реалізація основних функцій програмного засобу для аналізу мережевого трафіку є критично важливим етапом у створенні ефективного та надійного інструменту. Основна увага була приділена забезпеченню коректного захоплення мережевих пакетів у реальному часі та їх подальшого аналізу. Використання бібліотеки Scapy дозволило створити потужний механізм для збору мережевого трафіку, що є основою для подальшого аналізу.

Аналіз захоплених пакетів включав визначення їх типу та витягнення ключових атрибутів, таких як MAC та IP адреси, протоколи та інші метадані. Це дозволило забезпечити глибоке розуміння структури та змісту мережевого трафіку, що проходить через обраний інтерфейс [6].

Також було реалізовано функції для збереження та імпорту даних у форматі, що відповідає стандартам Wireshark, що забезпечує сумісність з іншими інструментами аналізу мережевого трафіку та спрощує подальший аналіз і обробку даних.

Загалом, реалізація основних функцій програмного засобу створила міцну основу для забезпечення надійного та ефективного аналізу мережевого трафіку, що є необхідним для моніторингу та забезпечення безпеки мережі. Повний лістинг програмного коду програми знаходиться в додатку А.

3.3 Тестування програмного засобу

Тестування є важливим етапом розробки, який забезпечує виявлення та усунення помилок, а також перевірку відповідності програмного засобу вимогам користувачів. Основна мета тестування - забезпечити високу якість та надійність програмного засобу, перевірити його функціональність та стабільність роботи. Тестування повинно охоплювати всі основні функції програми, включаючи захоплення трафіку, аналіз пакетів, збереження та імпорт даних, а також відображення інформації про пакети у таблиці, спрямоване на доведення працездатності розробленого програмного засобу для аналізу мережевого трафіку та його відповідності поставленому завданню.

3.3.1 Розробка тестових сценаріїв та стратегії тестування

Для проведення тестування були розроблені тестові сценарії, які включають різні аспекти функціональності програмного засобу. Основні тестові сценарії включають:

Для захоплення мережевого трафіку:

- Перевірка можливості вибору мережевого інтерфейсу для захоплення трафіку.
- Перевірка коректного початку, паузи та відновлення процесу захоплення трафіку.

– Перевірка захоплення пакетів різних типів (TCP, UDP, ICMP) та коректного їх відображення.

Для аналізу мережевого трафіку:

– Перевірка коректного визначення протоколу пакету.

– Перевірка витягнення та відображення атрибутів пакету (MAC та IP адреси, протоколи).

– Перевірка коректного заповнення таблиці із захопленими даними.

Збереження даних:

– Перевірка можливості збереження захоплених даних у файл у форматі, сумісному з Wireshark.

– Перевірка коректності збережених даних, включаючи всі атрибути пакетів.

Імпорт даних:

– Перевірка можливості імпортування даних з файлу у форматі Wireshark.

– Перевірка коректного відображення імпортованих даних у таблиці.

Вибір стратегії тестування базується на забезпеченні всебічного тестування системи. Це включає використання методів «чорної» та «білої» скриньки, а також різних типів тестування: функціонального, нефункціонального, тестування продуктивності та зручності використання [7].

Метод "чорної скриньки" дозволяє тестувати функціональність програмного засобу без знання його внутрішньої структури. Тестові випадки створюються на основі специфікацій вимог та функціональних можливостей. Це дозволяє переконатися, що всі заявлені функції працюють коректно. Метод "білої скриньки" використовується для тестування внутрішньої структури та логіки програми. Це включає перевірку коректності алгоритмів, контролю потоків даних та логічних операцій.

Функціональне тестування перевіряє основні функції програми, такі як захоплення мережевого трафіку, аналіз пакетів, збереження та імпорт даних. Це

забезпечує коректність роботи кожної функції згідно зі специфікацією вимог. Нефункціональне тестування включає перевірку зручності використання інтерфейсу користувача, стабільності та надійності програми під час тривалого використання.

Функціональне тестування потребує перевірки всіх функціональних можливостей окремо та у комплексі. Для кожної функції повинні бути визначені вхідні дані, очікувані результати та критерії успішності. Нефункціональне тестування включає перевірку стабільності програми при тривалому використанні та тестування з великим обсягом мережевого трафіку для перевірки надійності та стабільності програми.

Обсяг тестування включає перевірку всіх функціональних та нефункціональних вимог, визначених у специфікаціях. Тестування проводиться у кілька етапів, включаючи модульне тестування, інтеграційне тестування, системне тестування та приймальне тестування. Модульне тестування перевіряє окремі модулі програми на коректність виконання їх функцій. Інтеграційне тестування перевіряє взаємодію між модулями програми та коректність їх роботи у комплексі. Системне тестування є комплексним тестуванням всієї системи для перевірки її відповідності специфікаціям та вимогам. Приймальне тестування є остаточним тестуванням системи перед її випуском, включаючи тестування зручності використання та продуктивності.

3.3.2 Проведення тестування

Тестування починається з підготовки середовища тестування, яке має відповідати умовам реальної експлуатації програмного засобу. Для цього необхідно перевірити конфігурацію пристрою для роботи з мережевими інтерфейсами та забезпечити доступ до мережі.

Функціональне тестування включає перевірку основних функцій програми. Спочатку тестується можливість вибору мережевого інтерфейсу, для

чого запускається програмний засіб (рисунок 3.1) та перевіряється, чи можна вибрати різні інтерфейси зі списку (рис. 3.2-3.3).

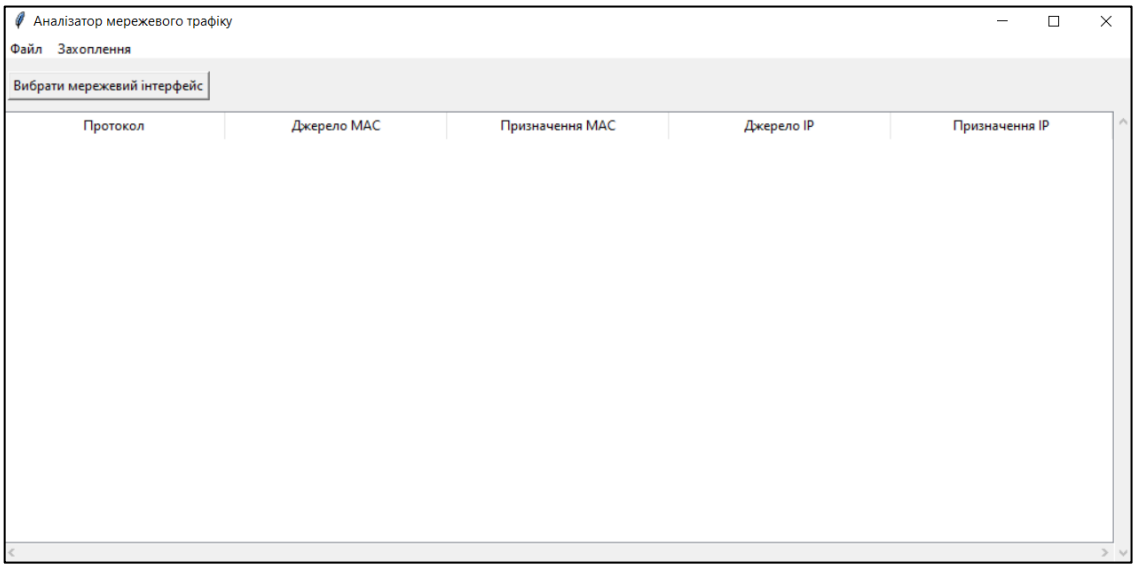


Рисунок 3.1 – Запущений додаток

Програма успішно запустилась, без виникнення критичних помилок, які б можна було помітити на цьому етапі. Коректно відображається графічний інтерфейс (рисунок 3.2).

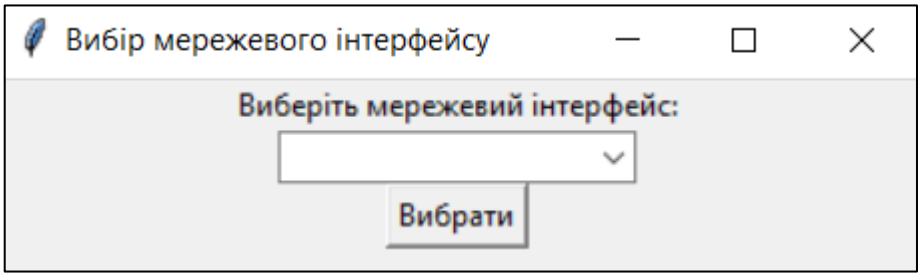


Рисунок 3.2 – Відкрите вікно вибору мережевого інтерфейсу

Вікно вибору мережевого інтерфейсу відкривається після натискання кнопки «Вибрати мережевий інтерфейс» та правильно показує свій вміст (рисунок 3.3).

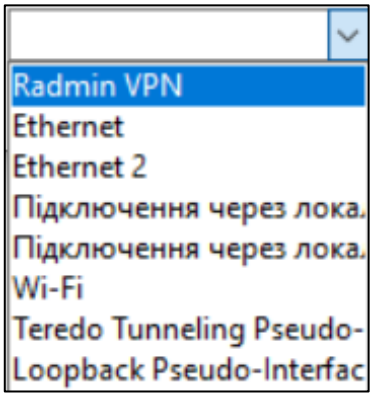


Рисунок 3.3 – Мережеві інтерфейси в комбінованому списку

При натисканні на елемент управління, згаданий раніше, відображається перелік наявних мережевих інтерфейсів, які є на пристрої. Для того, щоб перевірити чи дійсно ці інтерфейси є на пристрої і чи відображає всі наявні, можна зкористатись вбудованими консольними застосунками і командами в них. Для операційної системи Windows таким застосунком є cmd.exe, в якому потрібно ввести команду ipconfig, внаслідок чого, будуть виведені всі мережеві інтерфейси (рисунок 3.4). Для Unix-подібних операційних систем, такі як MacOS або різні дистрибутиви Linux, використовується застосунках terminal, в якому застосовується команда ifconfig.

Згідно з комбінованим списком, що видала програма, мають бути такі мережеві інтерфейси:

– Radmin VPN – штучно створений однойменним застосунком мережевий інтерфейс.
– Ethernet.
– Ethernet 2.
– Два незайнятих порти для дротового з’єднання з мережею.
– Wi-Fi – безпровідна мережа.
– Тунельний адаптер Teredo Tunneling Pseudo-Interface.

На рисунку 3.4 приведено типові мережеві інтерфейси для операційної системи Windows.

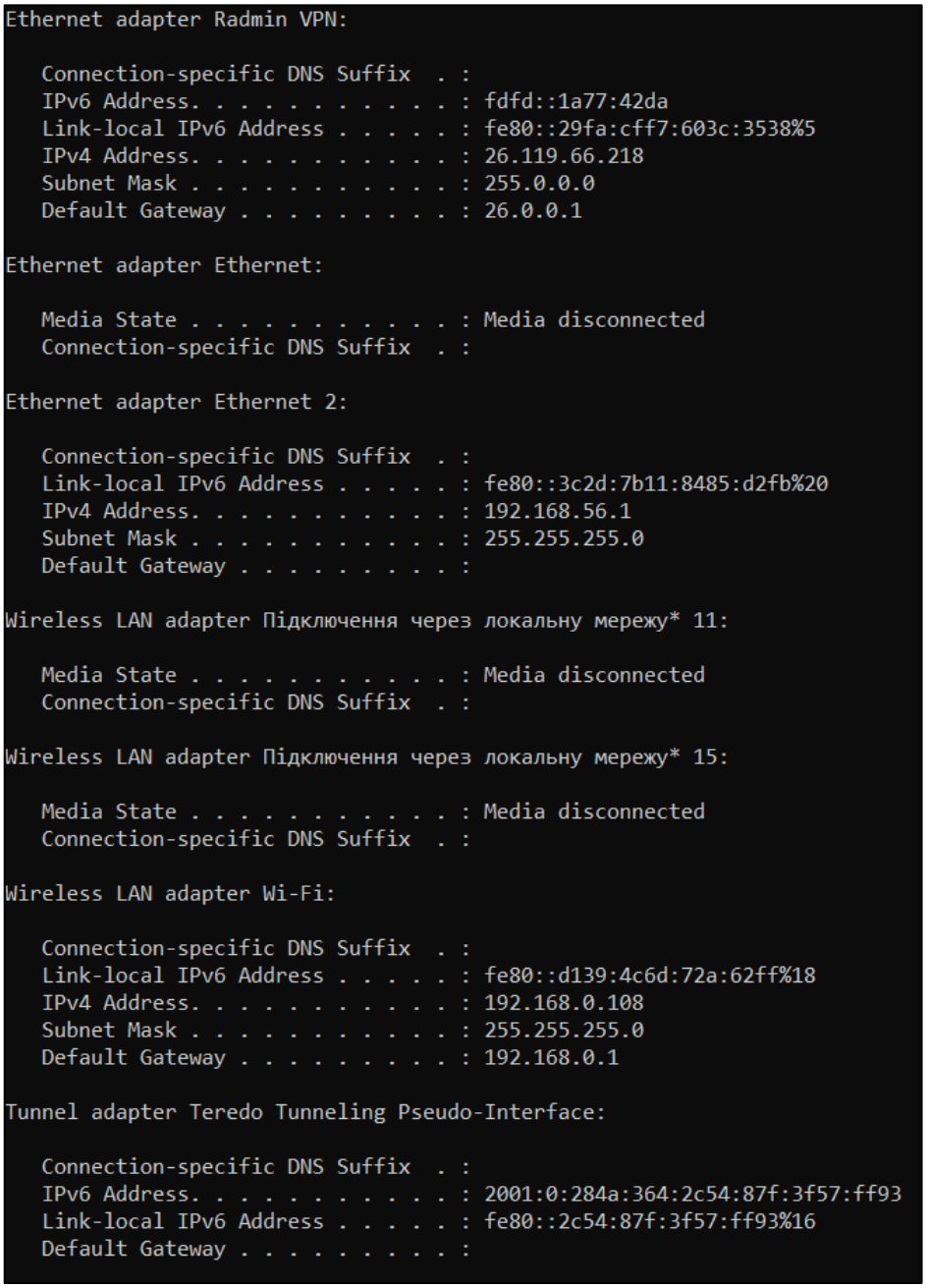


Рисунок 3.4 – Виведені мережеві інтерфейси в cmd.exe для операційної системи Windows

Дані з програми і з консольного застосунку збігаються, отже, в програмі правильно реалізований вибір мережевого інтерфейсу.

Далі тестується функція захоплення мережевого трафіку. Для цього обирається інтерфейс та починається захоплення, після чого аналізуються захоплені пакети на наявність коректної інформації. Під час тестування, інтернет

з’єднання було проведене через безпровідну мережу, отже треба вибрати Wi-Fi в якості мережевого інтерфейсу для захоплення трафіку (рисунок 3.5).

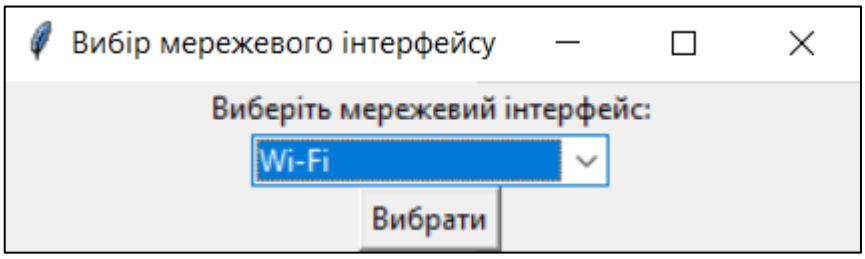


Рисунок 3.5 – Обраний мережевий інтерфейс для захоплення трафіку з нього

Після вибору, потрібно натиснути на кнопку «Вибрати» в цьому вікні, після чого в головному меню зверху вибрати вкладку «Захоплення» та натиснути на «Почати аналіз».

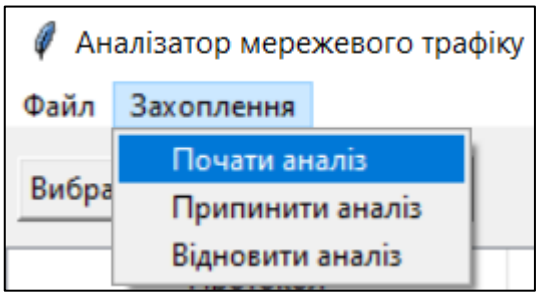


Рисунок 3.6 – Навігація для початку захоплення

Після виконання таких кроків програма почне захоплювати трафік, аналізувати його, а потім виводити в таблицю дані про кожен пакет. Для перевірки коректності захоплення і аналізу трафіку, слід застосувати ще один програмний засіб для аналізу мережевого трафіку, наприклад, Wireshark. Програма, що тестується і стороння професійна програма будуть запущенні синхронно і таким чином, можна буде звірити дані з обох програм і пересвідчитись в коректності захоплення даних застосунку, який тестується.

Спершу варто запустити та налаштувати Wireshark для роботи, для цього після запуску потрібно обрати такий ж мережевий інтерфейс, а саме Wi-Fi (рисунок 3.7).

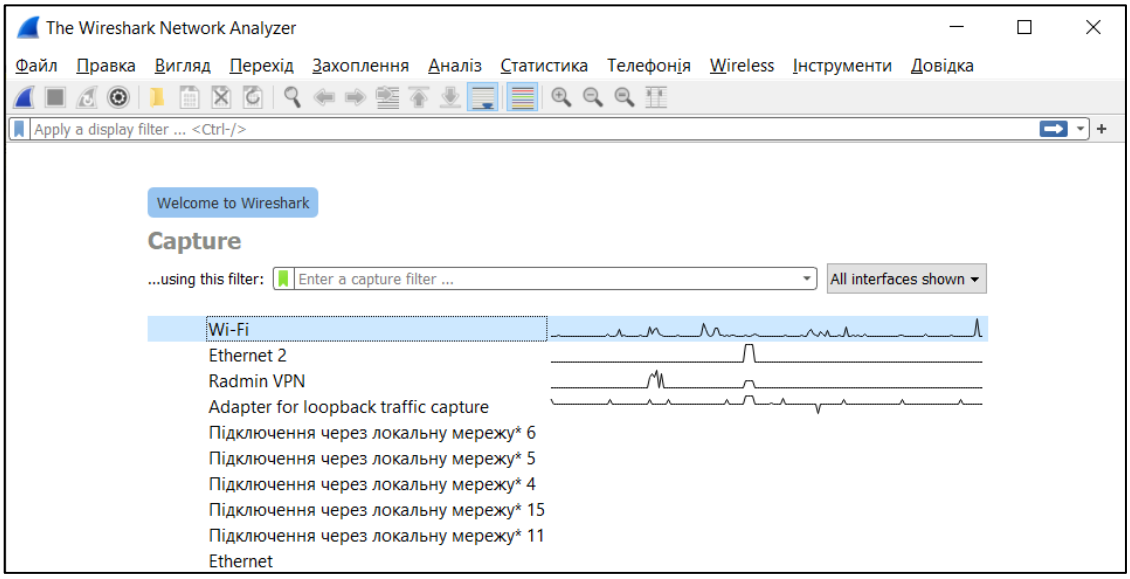


Рисунок 3.7 – Головне меню застосунку Wireshark

Потрібно двічі натиснути на необхідний мережевий інтерфейс, що одразу запустить захоплення трафіку. Коли обидва застосунки пропрацюють деякий час, достатньо не більше 10-ти секунд, можна вимикати захоплення в обох програмах і звіряти результати між собою (рисунки 3.8-3.9).

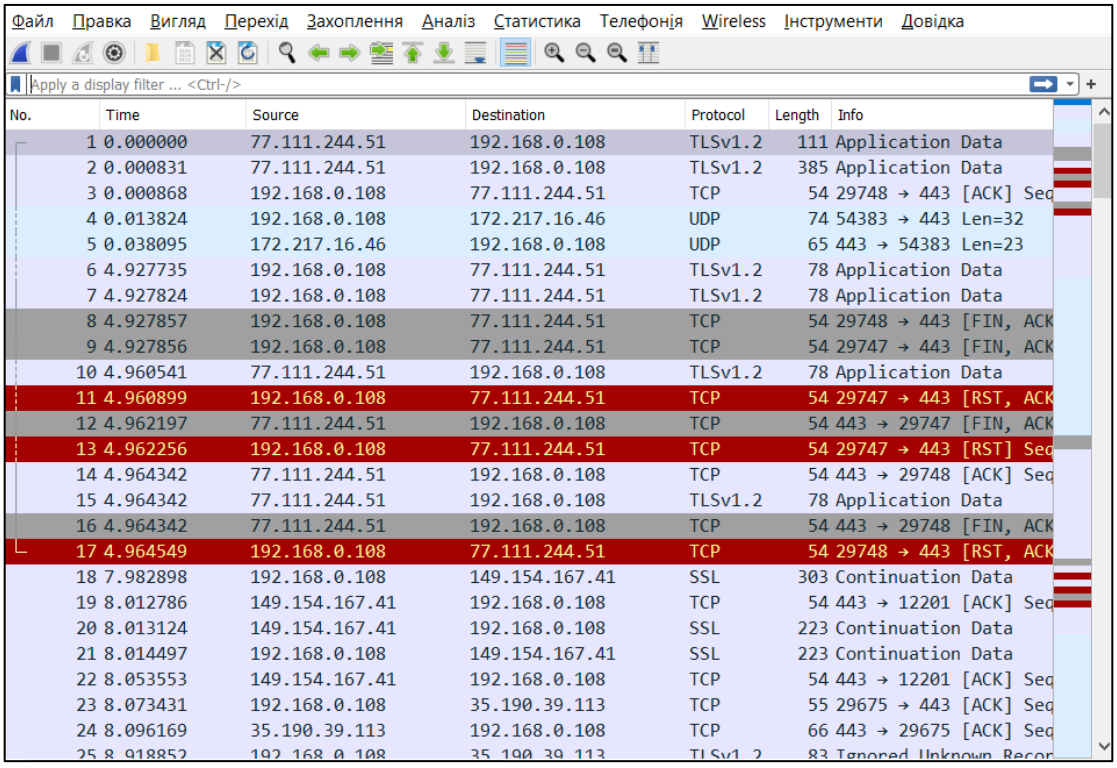


Рисунок 3.8 – Зібрані дані з Wireshark

Для перевірки можна зробити порівняння IP-адрес призначення та IP-адрес джерел перших десяти мережевих пакетів. Згідно з даних з Wireshark виділяємо наступні дані (перша адреса – джерело, друга - призначення):

1) 77.111.244.51 / 192.168.0.108
2) 77.111.244.51 / 192.168.0.108
3) 192.168.0.108 / 77.111.244.51
4) 192.168.0.108 / 172.217.16.46
5) 172.217.16.46 / 192.168.0.108
6) 192.168.0.108 / 77.111.244.51
7) 192.168.0.108 / 77.111.244.51
8) 192.168.0.108 / 77.111.244.51
9) 192.168.0.108 / 77.111.244.51
10) 77.111.244.51 / 192.168.0.108

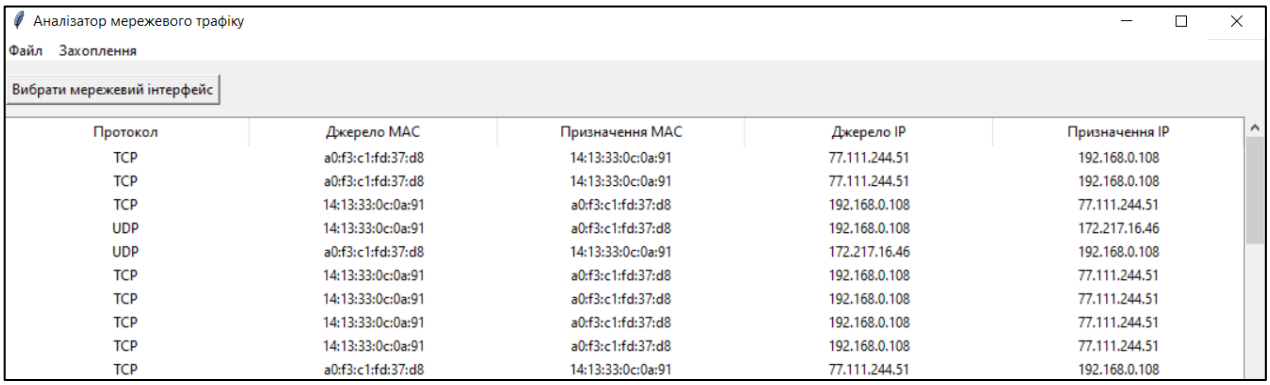


Рисунок 3.9 – Зібрані дані з тестованого застосунку

Як видно, дані пакетів програми збігаються з даними зібраних за допомогою застосунку Wireshark, а отже, захоплення трафіку працює належним чином.

Наступним етапом є тестування функцій зупинки та відновлення захоплення трафіку. Тестування зупинки включає перевірку коректності зупинки процесу захоплення. Тестування відновлення включає перевірку можливості продовження захоплення з місця зупинки. Для припинення захоплення трафіку

треба в головному меню натиснути на вкладку «Захоплення» і обрати «Припинити аналіз» (рисунок 3.10).

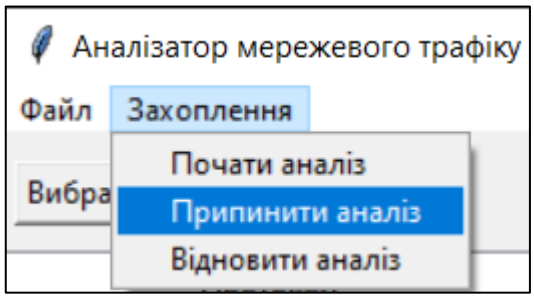


Рисунок 3.10 – Припинення захоплення трафіку

Після здійснення цих дій захоплення трафіку, і як наслідок, вивід в таблицю, зупиняється. Коли програма знаходиться на паузі, її роботу можна відновити, натиснувши на вкладку «Захоплення» і обрати «Відновити аналіз»(рисунок 3.11).

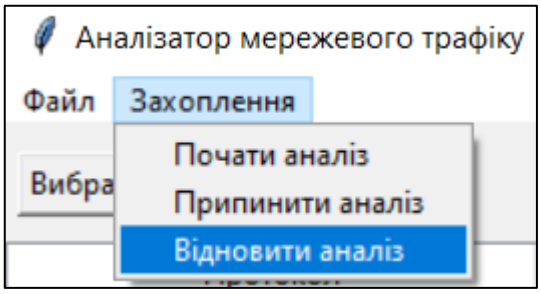


Рисунок 3.11 – Відновлення захоплення трафіку

Програма продовжить захоплення і аналіз трафіку з того ж моменту, на якому завершила і продовжить вивід в таблицю дані про пакети.

Далі, потрібно провести тестування Функції збереження та імпорту даних тестується шляхом збереження захоплених пакетів у файл та наступного імпорту цього файлу. Це включає перевірку формату файлу, правильності збереження даних та можливості їх подальшого імпорту. Щоб зберегти дані аналізу трафіку, потрібно спершу його захопити попередньо і зупинити захоплення трафіку, після

чого можна вже його зберігати за допомогою вкладки «Файл» та пункту «Зберегти дані» (рисунок 3.12).

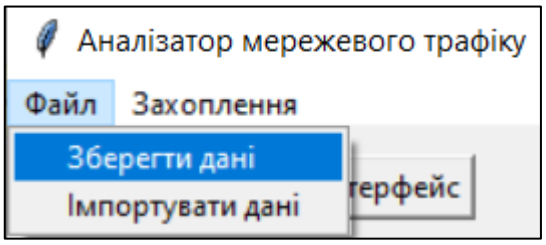


Рисунок 3.12 – Збереження даних захопленого трафіку

Після виконання цих кроків, відкриється вікно провідника, де потрібно буде вибрати місце де будуть збережені дані (рисунок 3.13). Дані зберігаються в двох файлах з форматом «.csv», який є типом файлів для табличних процесорів і використовується для імпорту в цю програму та «.pcap», який є уніфікованим типом даних для зберігання даних про мережевий трафік під назвою «capture file».

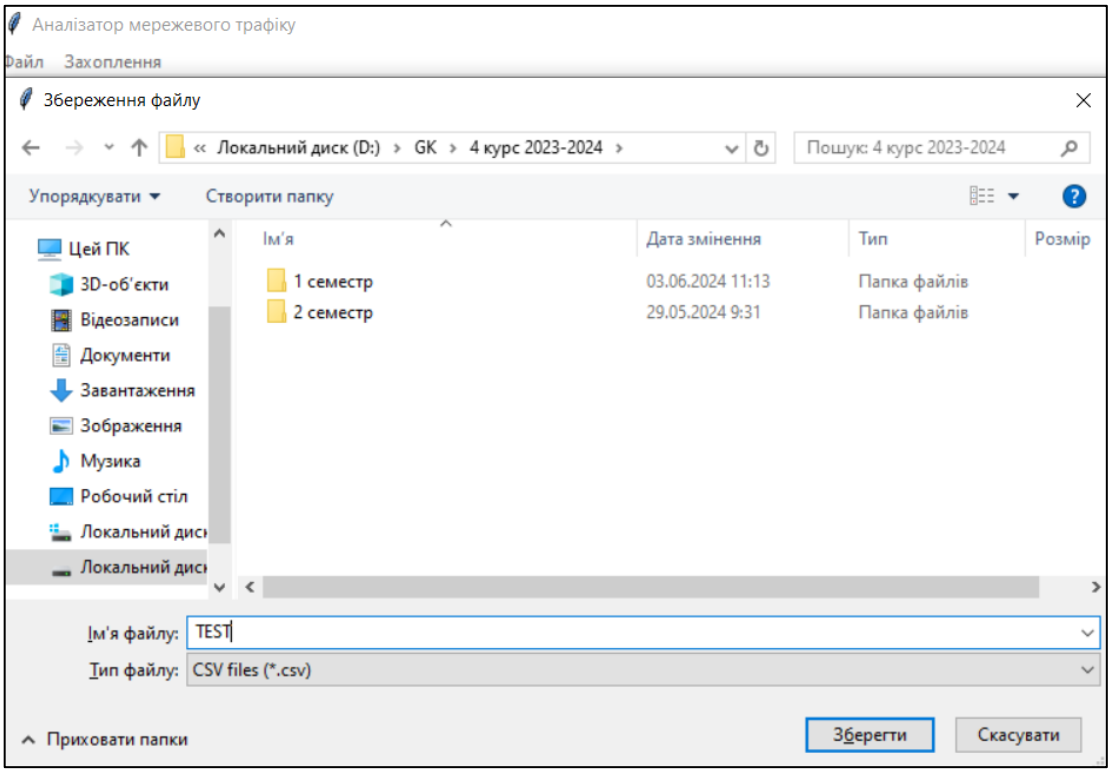


Рисунок 3.13 – Зберігання даних на пристрої

Зберігання у такий спосіб робить процес аналізу користувачем значно зручнішим, тому що подає їх в вигляді таблиці, яку можна відкрити за допомогою табличного процесора Microsoft Excel або Libre Office.

Тепер, ці файли можна використати для імпортування для більшості застосунків для аналізу мережевого трафіку, включаючи той, який тестується в цій роботі. Щоб імпортувати збережені файли назад в описаний застосунок, треба в вкладці «Файл» вибрати «Імпортувати дані» (рисунок 3.14).

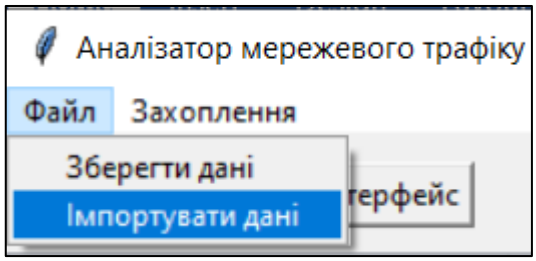


Рисунок 3.14 – Імпорт даних в програму

Після цих дій відкриється вікно, де потрібно буде обрати потрібний файл формату «.csv» і натиснути кнопку «Відкрити» в вікні провідника операційної системи (рисунок 3.15).

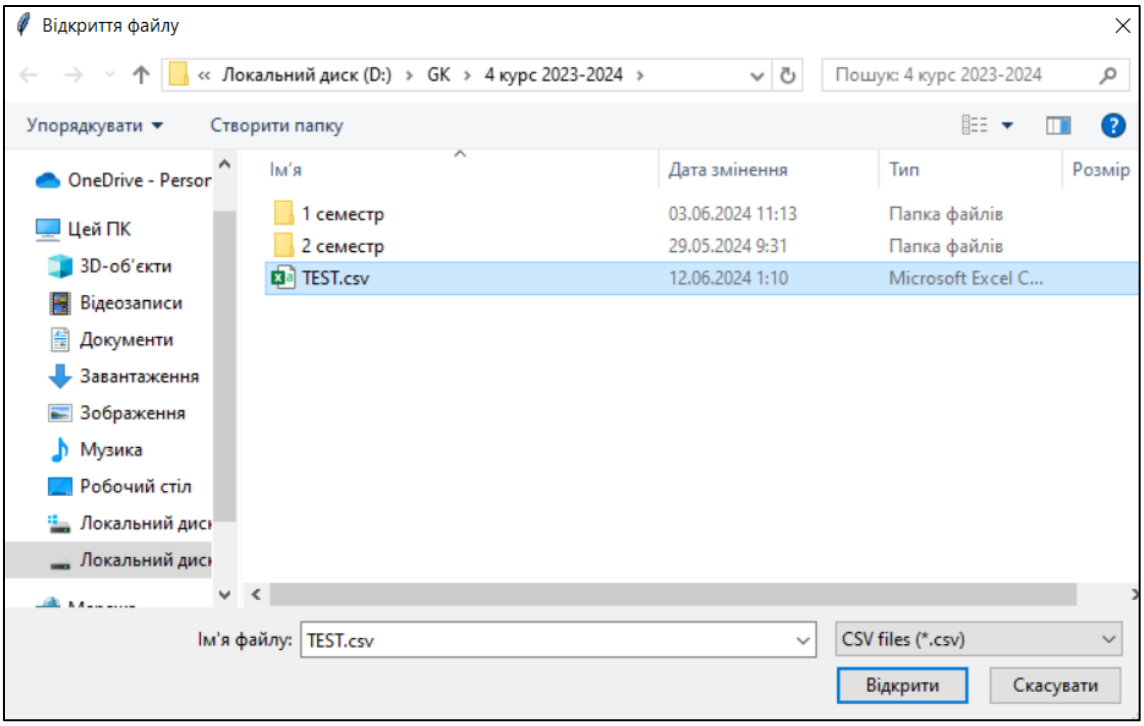
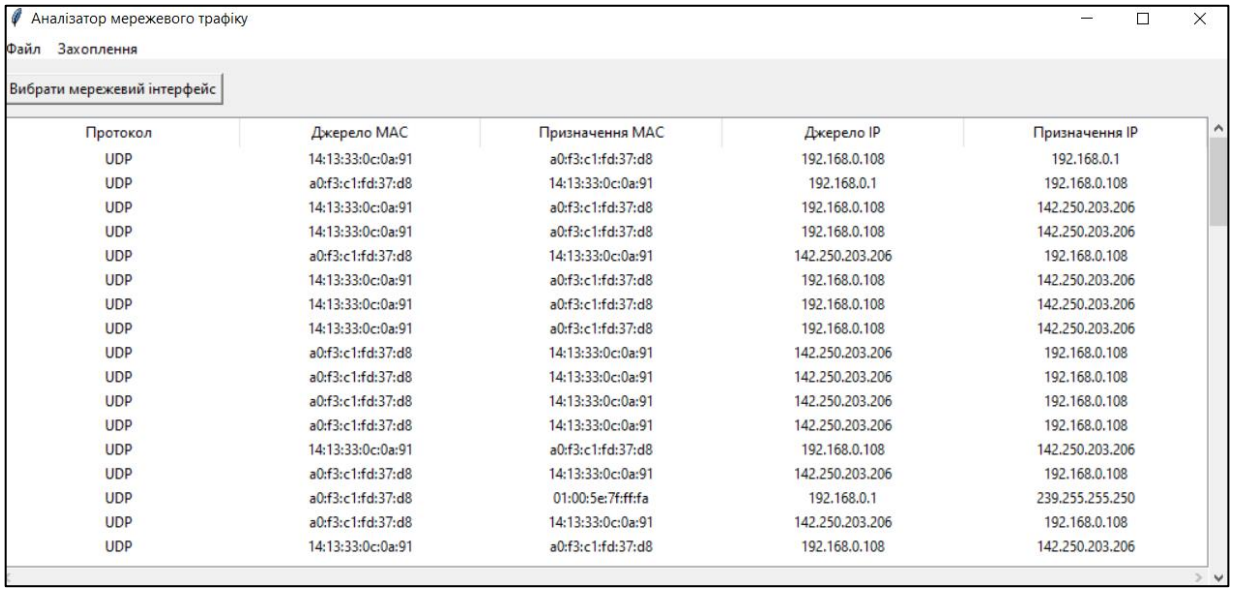


Рисунок 3.15 – Імпортування даних мережевих пакетів з пристрою

Після завершення цієї операції, в програмі, в таблиці з’являться всі дані, які були попередньо захоплені (рисунок 3.16), з якими в подальшому можна продовжити працювати та доповнювати таблицю повторним захопленням даних.



Протокол	Джерело MAC	Призначення MAC	Джерело IP	Призначення IP
UDP	14:13:33:0c:0a:91	a0:f3:c1:fd:37:d8	192.168.0.108	192.168.0.1
UDP	a0:f3:c1:fd:37:d8	14:13:33:0c:0a:91	192.168.0.1	192.168.0.108
UDP	14:13:33:0c:0a:91	a0:f3:c1:fd:37:d8	192.168.0.108	142.250.203.206
UDP	14:13:33:0c:0a:91	a0:f3:c1:fd:37:d8	192.168.0.108	142.250.203.206
UDP	a0:f3:c1:fd:37:d8	14:13:33:0c:0a:91	142.250.203.206	192.168.0.108
UDP	14:13:33:0c:0a:91	a0:f3:c1:fd:37:d8	192.168.0.108	142.250.203.206
UDP	14:13:33:0c:0a:91	a0:f3:c1:fd:37:d8	192.168.0.108	142.250.203.206
UDP	14:13:33:0c:0a:91	a0:f3:c1:fd:37:d8	192.168.0.108	142.250.203.206
UDP	a0:f3:c1:fd:37:d8	14:13:33:0c:0a:91	142.250.203.206	192.168.0.108
UDP	a0:f3:c1:fd:37:d8	14:13:33:0c:0a:91	142.250.203.206	192.168.0.108
UDP	a0:f3:c1:fd:37:d8	14:13:33:0c:0a:91	142.250.203.206	192.168.0.108
UDP	a0:f3:c1:fd:37:d8	14:13:33:0c:0a:91	142.250.203.206	192.168.0.108
UDP	14:13:33:0c:0a:91	a0:f3:c1:fd:37:d8	192.168.0.108	142.250.203.206
UDP	a0:f3:c1:fd:37:d8	14:13:33:0c:0a:91	142.250.203.206	192.168.0.108
UDP	a0:f3:c1:fd:37:d8	01:00:5e:7f:ff:fa	192.168.0.1	239.255.255.250
UDP	a0:f3:c1:fd:37:d8	14:13:33:0c:0a:91	142.250.203.206	192.168.0.108
UDP	14:13:33:0c:0a:91	a0:f3:c1:fd:37:d8	192.168.0.108	142.250.203.206

Рисунок 3.16 – Імпортовані дані в полі програмного засобу

Також, варто провести тестування на сумістність захоплених даних з застосунком Wireshark, виконавши імпорт даних, які зібрані тестованим застосунком в професійному. Для цього в Wireshark потрібно натиснути на вкладку «Файл» та вибрати «Відкрити», або ж просто використати комбінацію клавіш Ctrl+O (рисунок 3.17).

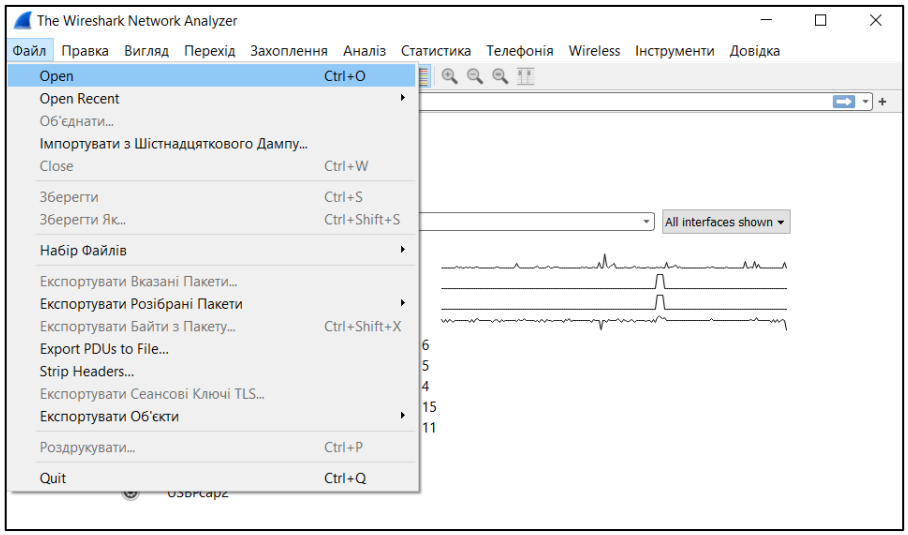


Рисунок 3.17 – Імпортування даних мережевого трафіку в Wireshark

Відкриється вікно провідника, де потрібно обрати файл в форматі «.pcap», який підтримується в цій програмі (рисунок 3.18).

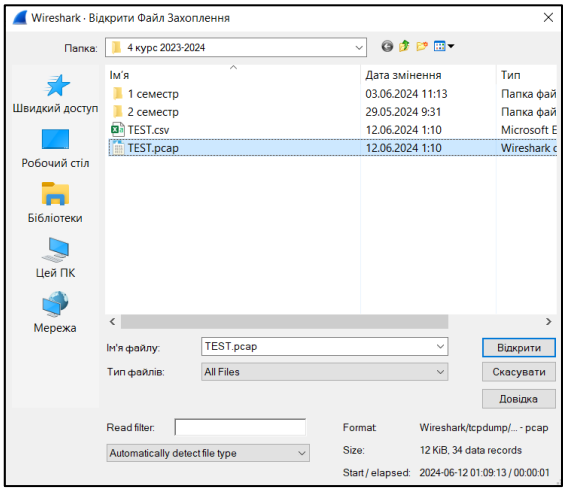


Рисунок 3.18 – Імпортування тестового файлу

Як наслідок, файл успішно відкриється в застосунку (рисунок 3.19) і буде повністю підтримуватись, про що свідчить додаткова інформація, яка не відображається в тестованій програмі.

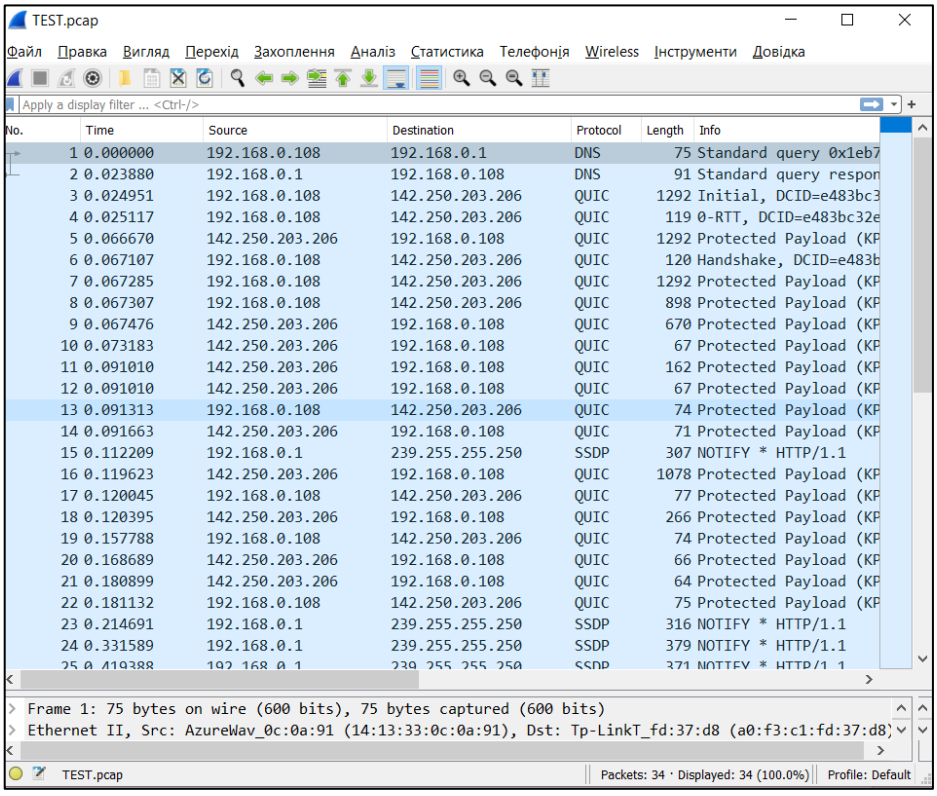


Рисунок 3.19 – Успішно імпортований файл в Wireshark

Таким чином, доведена сумістність програми з іншими поширеними програмами, які є аналогами до проєкту.

Результати повного тестування представлені в таблиці 3.1.

Таблиця 3.1 – Тестові сценарії для програмного забезпечення

Тестовий сценарій	Очікуваний результат	Фактичний результат	Статус
Вибір мережевого інтерфейсу	Інтерфейс успішно обрано	Інтерфейс успішно обрано	Пройдено
Початок захоплення трафіку	Захоплення трафіку почалося	Захоплення трафіку почалося	Пройдено
Пауза захоплення трафіку	Захоплення трафіку призупинено	Захоплення трафіку призупинено	Пройдено
Відновлення захоплення трафіку	Захоплення трафіку відновлено	Захоплення трафіку відновлено	Пройдено
Визначення протоколу пакету	Протокол пакету визначено коректно	Протокол пакету визначено коректно	Пройдено
Витягнення та відображення атрибутів	Атрибути пакету витягнуто та відображено коректно	Атрибути пакету витягнуто та відображено коректно	Пройдено
Заповнення таблиці із захопленими даними	Дані відображено в таблиці	Дані відображено в таблиці	Пройдено
Збереження даних у файл	Дані збережено у файл коректно	Дані збережено у файл коректно	Пройдено
Імпорт даних з файлу	Дані імпортовано та відображено коректно	Дані імпортовано та відображено коректно	Пройдено
Сумістність збережених файлів з іншими застосунками-аналогами	Дані успішно імпортуються при захопленні трафіку з застосунків в обидві сторони	Дані успішно імпортуються при захопленні трафіку з застосунків в обидві сторони	Пройдено

Тестування програмного засобу для аналізу мережевого трафіку продемонструвало високу коректність роботи всіх основних функцій та їх

відповідність заданим вимогам. Було виконано комплексний підхід до перевірки, який включав функціональне, нефункціональне та тестування продуктивності.

Функціональне тестування охоплювало всі ключові можливості програмного засобу. Спочатку була протестована функція вибору мережевого інтерфейсу. Програмний засіб коректно відображає всі доступні інтерфейси та дозволяє користувачу без проблем обрати потрібний. Після цього проводилося тестування захоплення мережевого трафіку. Програмний засіб успішно розпочинає захоплення трафіку через вибраний інтерфейс, фіксуючи всі пакети, що проходять через нього.

Аналіз захоплених пакетів також показав позитивні результати. Програмний засіб правильно розпізнає та відображає основну інформацію про пакети, включаючи тип протоколу, MAC-адреси джерела та призначення, а також IP-адреси джерела та призначення. Це підтверджує коректність реалізації логіки аналізу пакетів.

Функції зупинки та відновлення захоплення трафіку також були перевірені. Після натискання кнопки зупинки процес захоплення припиняється, і користувач може переглянути всі захоплені пакети. При відновленні захоплення процес продовжується без втрати даних, що вказує на стабільність роботи програмного засобу.

Збереження та імпорт даних були протестовані шляхом збереження захоплених пакетів у форматі CSV та подальшого імпорту цих даних. Програмний засіб коректно зберігає всі необхідні дані у файл та дозволяє успішно імпортувати їх, відображаючи інформацію у таблиці.

Нефункціональне тестування включало перевірку зручності використання інтерфейсу. Користувачі, які тестували програмний засіб, зазначили високу інтуїтивність інтерфейсу, зручність навігації та легкість виконання основних операцій. Відгуки користувачів підтвердили, що інтерфейс добре спроєктований і відповідає очікуванням.

Тестування продуктивності показало, що програмний засіб може обробляти великий обсяг мережевого трафіку без значних затримок або зниження продуктивності. Час відгуку залишався на прийнятному рівні навіть при високому навантаженні, що свідчить про ефективність реалізації програмного засобу.

В результаті тестування всі основні функції програмного засобу були перевірені та підтверджені. Програмний засіб успішно захоплює, аналізує, зберігає та імпортує мережевий трафік, забезпечуючи надійність і точність роботи. Це дозволяє зробити висновок про відповідність програмного засобу заданим вимогам та його готовність до практичного використання.

РОЗДІЛ 4 ТЕХНІКО-ЕКОНОМІЧНЕ ОБҐРУНТУВАННЯ

4.1 Аналіз ринку збуту розробленого застосунку

Розроблений попередньо аналізатор мережевого трафіку є десктопним застосунком, який чудово підійде новачкам у сфері комп'ютерних мереж або ж системного адміністрування. Популярні програми-аналоги, часто, дуже складні в освоєнні, що породжує безліч трудностей для нових користувачів в процесі навчання в цих сферах. Цей ж застосунок простий і дозволить новачкам в сфері комп'ютерних мереж плідно засвоювати нову інформацію, не відволікаючись на велику кількість не потрібної інформації, на початковому рівні знань.

Потенційними користувачами цього аналізатора мережевого трафіку можуть бути студенти молодших груп за спеціальностями, що є у сфері інформаційних технологій, учні старших класів в школах на уроках інформатики, що лише ознайомлюються з комп'ютерними технологіями, а також фахівці в цій, або дотичних галузях, яким базовий функціонал цілком достатній, або їм надзвичайно важлива швидкодія захоплення і аналіз трафіку.

Застосунок для аналізу мережевого трафіку є спеціалізованою програмою, що забезпечує захоплення, аналіз, збереження та імпорт даних мережевого трафіку. Його основними техніко-економічни характеристики є висока продуктивність при обробці великих обсягів трафіку, яка забезпечена простотою програми та стійкістю до фатальних помилок, які б припиняли роботу програми, є можливість збереження та імпорту даних у зручному форматі, який є сумістний з більш професійним застосунком для аналізу трафіку Wireshark, а також захоплює мережеві пакети без втрат, що доведено тестуванням.

Програмний засіб є новим продуктом на ринку, який поєднує в собі необхідні функції для аналізу мережевого трафіку з простим і зручним інтерфейсом. Представники потенційних замовників ринку збуту включають IT-компанії, провайдери інтернет-послуг, а також різні навчальні заклади для

освітніх цілей. Очікується високий попит на програмний засіб, оскільки щороку дедалі більше потреба в забезпеченні кібербезпеки, виявлення вразивостей в мережі і усування кіберзагроз.

Основним методом реалізації продукту будуть продажі через інтернет-магазини, методом згенерованого унікального ключа та на спеціалізованих платформах для програмного забезпечення. Прогнозується, що обсяги продажу будуть зростати поступово в перші роки після запуску на ринок, з подальшим стабільним зростанням, але помірно нижчим ніж на початку темпом, у супровід з оновленням продукції, її модернізації, модифікаії та вдосконалення. Продукція буде поширена в обмежених та тематичних спільнотах, здебільшого серед IT-фахівців, або студентів. Можливий життєвий цикл нового програмного забезпечення прогнозується як тривалий, враховуючи постійний розвиток мережевих технологій та зростаючу потребу в забезпеченні безпеки мереж. Регулярні оновлення та вдосконалення функціоналу будуть сприяти підтримці актуальності програмного продукту та його конкурентоспроможності на ринку збуту.

Головними конкурентами є іменити продукти, такі як Wireshark, SolarWinds Network Performance Monitor та PRTG Network Monitor. Конкурентна продукція зазвичай має високий рівень техніко-економічних та споживчих показників, пропонуючи широкий набір функцій для аналізу мережевого трафіку. Цінова політика конкурентів варіюється в залежності від набору функцій та типу ліцензії. Проте, розроблений програмний засіб буде конкурентоспроможним завдяки поєднанню високої якості, надійності, інтуїтивного інтерфейсу та доступної цінової політики. Пропонуватимуться різні ліцензійні плани, зокрема корпоративні знижки та бонуси для постійних клієнтів.

Отже, всі ці фактори разом підтверджують доцільність просування нового продукту, який здатний зайняти своє місце на ринку програмного забезпечення для аналізу мережевого трафіку.

4.2 Обрахування витрат на проєктування застосунку

Задля визначення суми всіх витрат коштів, залучених до розробки програмного засобу, варто використати «трудовий» метод обрахунку, суть якого в тому, щоб оцінювати всі витрати, виходячи з заробітної плати проєктувальників та інших робочих, що брали участь в реалізації проєкту. Витрати роздподіляються по 11-и статтям витрат і оформлюються в спеціальний документ під назвою кошторис витрат [8]. Таблиця кошторису, що ілюструє всі витрати, зібрані в одну таблицю, знаходиться в додатку Б.

Обрахунок заробітньої плати для кожної категорії працівників виходить з кількості працівників-виконавців, посадових окладів та терміну участі у розробці в місяцях.

Рекомендована кількість виконавців – до 3 осіб, тривалість розробки – 5 місяців. Розрахунок заробітної плати відображений в таблиці 4.1.

Таблиця 4.1 – Розраховані заробітні плати для учасників проєкту

N п/п	Посада	Оклад, грн/міс	Відрахування, грн/міс	Кількість, чол.	Кількість, міс	З/п, грн
1	Старший науковий співробітник	8234.1	1811.5	1	5	41170
2	Науковий співробітник	7729.9	1700.6	1	5	38649.5
3	Інженер першої категорії	6773.4	1 489.7	1	5	33867
	Усього зарплати:					113686.5

Оклади визначаються за допомогою множення ставки першого розряду на тарифний коефіцієнт що дорівнює тарифному коефіцієнту, що врегульовується державою, помноженому на 3195 грн, тобто на ставку першого розряду.

Сума відрахувань на соціальні потреби дорівнює 22% від суми заробітних плат проєктувальників, а це 25011 грн.

Вартість контрагентських робіт дорівнює 20% від суми заробітніх плат проєктувальників, що дорівнює 22737 грн.

Витрати на відрядження обраховуються за допогомогою реального передбачення можливої кількості відряджень. Під час роботи над цим проєктом, заплановано близько 20 відряджень, з оціночною вартістю поїздки 750 грн. Отже, ці витрати дорівнюють добутку кількості відряджень та їх вартості, а саме 15000 грн.

Інші прямі витрати запроваджені на врахування витрат на різного роду матеріалів, обаднання і, загалом, всього, що може знадобитись для продовження робочої діяльності і роботи над розробкою проєкту. Такі витрати дорівнюють 50% від заробітної плати, що виходить 56843 грн.

Підсумкова сума прямих витрат обраховується з суми всіх попередніх витрат, включаючи заробітню плату (113686 грн), відрахування на соціальні потреби (25011 грн), контрагентські послуги (22737 грн), на відрядження (15000 грн), інші прямі витрати (56843 грн) що при обрахунку виходить 233277 грн.

До накладних витрат відносяться загальногосподарські витрати, які забезпечують процес роботи: опалення, електроенергія, зарплата адміністративного персоналу та інші. Накладні витрати складають 30% від суми прямих витрат, і це становить 69983 грн.

Планові накопичення – прибуток, що використовується з метою розвитку матеріально-технічної бази підприємства або надання премій співробітникам. Ці накопичення становлять 20% від суми прямих витрат та накладних витрат, що дорівнює 60652 грн.

Повна кошторисна вартість проєкту включає прямі витрати, накладні витрати та планові накопичення. Тобто, кошторисна вартість проєкту обчислюється як 233277 грн (прямі витрати) + 69983 грн (накладні витрати) + 60652 грн (планові накопичення) = 364912 грн.

До кошторисної вартості проєкту додається податок на додану вартість (ПДВ), який визначається за діючим нормативом (20%). ПДВ складає 20% від кошторисної вартості проєкту, що дорівнює: 364912 грн * 20% = 72982.4 грн.

Загальна договірна ціна розробки проєкту розраховується як сума кошторисної вартості робіт та податку на додану вартість, і визначає витрати на проєктування, які має покрити підприємство-замовник. Отже, загальна договірна ціна розробки дорівнює: 364912 грн (кошторисна вартість) + 72982.4 грн (ПДВ) = 437894.4 грн.

4.3 Обґрунтування необхідності розробки

Сучасне управління та моніторинг мережевого трафіку вимагають ефективних інструментів для забезпечення безпеки та оптимальної роботи мережі. Розробка програмного засобу для аналізу мережевого трафіку є необхідною для задоволення потреб різних зацікавлених сторін, таких як мережеві адміністратори, IT-відділи, компанії, що займаються безпекою, а також для підвищення ефективності та безпеки роботи мереж.

Впровадження такого програмного засобу дозволить задовольнити наступні потреби замовників:

1) Мережеві адміністратори:

– Доступ до детальної інформації про мережевий трафік в режимі реального часу.

– Можливість швидкого виявлення аномалій та потенційних загроз.

– Зручні інструменти для аналізу та відстеження трафіку.

– Можливість оптимізації роботи мережі на основі отриманих даних.

2) IT-відділи:

– Централізоване управління мережею та її ресурсами.

– Можливість швидкого реагування на інциденти безпеки.

– Зручний спосіб моніторингу продуктивності мережі.

– Полегшення процесу аудиту та звітності.

3) Компанії, що займаються безпекою:

– Інструменти для виявлення та запобігання кіберзагрозам.

– Можливість інтеграції з існуючими системами безпеки і застосунками.

– Зручний та простий інтерфейс для аналізу загроз.

– Підвищення загального рівня безпеки мережевої інфраструктури.

Програмний засіб для аналізу мережевого трафіку позитивно вплине на низку економічних показників підприємств. Він дозволить підвищити ефективність роботи мережевих адміністраторів та ІТ-персоналу. Автоматизація процесів моніторингу та аналізу трафіку зменшить потребу у ручній праці, скоротить час на виявлення та реагування на інциденти, що підвищить загальну продуктивність праці.

Крім того, впровадження системи дозволить значно зменшити витрати на усунення наслідків мережевих інцидентів. Своєчасне виявлення загроз та аномалій дозволить мінімізувати збитки від потенційних атак та простоїв систем.

А також, програмний засіб сприятиме оптимізації використання ресурсів мережі. Завдяки детальному аналізу трафіку можна буде виявляти та усувати недоліки у мережі, що дозволить ефективніше використовувати наявні ресурси та знижувати потребу у додаткових інвестиціях у обладнання.

Таким чином, впровадження програмного засобу для аналізу мережевого трафіку дозволить підприємствам оптимізувати витрати на персонал, зменшити витрати на усунення мережевих інцидентів, підвищити ефективність роботи мережі та забезпечити високий рівень її безпеки.

Запровадження проєкту з розробки програмного засобу для аналізу мережевого трафіку спрямоване на отримання ефекту за кількома ключовими напрямками. По-перше, це підвищення якості моніторингу та аналізу мережевого трафіку. Впровадження сучасного програмного засобу покращить організацію процесу моніторингу, полегшить доступ до аналітичних даних та

покращить комунікацію між учасниками мережевої інфраструктури, що сприятиме підвищенню загальної безпеки мережі.

Другим важливим напрямком є покращення комунікації та обміну інформацією між учасниками мережевої інфраструктури - адміністраторами, IT-персоналом та керівництвом компанії. Програмний засіб забезпечить зручний спільний доступ до необхідних даних та ефективний інформаційний обмін.

Крім того, система дозволить оптимізувати управління мережею та її ресурсами. Адміністрація матиме зручний інструмент для моніторингу продуктивності мережі, планування оновлень та раціонального розподілу ресурсів.

Для мережевих адміністраторів та IT-персоналу система забезпечить зручний доступ до аналітичних інструментів та даних в режимі реального часу, що значно спростить їхню роботу.

Нарешті, запровадження програмного засобу дозволить скоротити витрати на усунення наслідків мережевих інцидентів та підвищити загальний рівень безпеки мережевої інфраструктури, що матиме позитивний вплив на загальні економічні показники діяльності компанії.

Таким чином, розробка програмного засобу для аналізу мережевого трафіку є необхідною для задоволення потреб замовників, підвищення ефективності роботи мережі та отримання економічних вигод для підприємств.

ВИСНОВКИ

Створення програмного застосунку для аналізу мережевого трафіку на мові програмування Python з використанням Scapy відіграє роль в забезпеченні безпеки та ефективної роботи мережевої інфраструктури. Метою реалізації цього проєкту було створення інструменту, який буде здатен виявляти мережеві аномалії, допомагати запобігати кіберзагрозам та надати можливість адміністраторам мережі забезпечувати стабільну роботу мереж .

Проведено аналіз існуючих рішень виконана постановка завдання, що дає можливість виділити основні функціональні вимоги до програмного засобу що проєктується та розробляється. Виконано проєктування програмного засобу та модулів, що дозволило провести аналіз процесів, формалізацію вимог до системи та виконати розробку модулів. Розроблений програмний засіб має функцію моніторингу і аналізу мережевого трафіку, а саме забезпечує можливість спостереження за мережевим трафіком в режимі реального часу, аналізувати його та ідентифіковувати аномальні активності в мережі, такі як розподілені атаки на відмову серверної частини сервісів в обслуговуванні, витоки даних, або інтернет-активність підозрілих небажаних програм. Програмний засіб є зручним інструментом для оптимізації мережевої взаємодії з відкритим кодом. Використання сучасних інструментів для аналізу та моніторингу мережевого трафіку забезпечує підвищення рівня захисту конфіденційної інформації, ефективне виявлення та реагування на потенційні загрози, а також зменшення ризиків для користувачів і підвищення загального рівня кібербезпеки [9].

У процесі розробки програмного засобу було використано мову програмування Python, яка відома своєю ефективністю та гнучкістю, що дозволяє розширювати засіб та його можливості в залежності від потреби. Для реалізації аналізу даних та мережевих пакетів були застосовані та досліджені бібліотеки, що дозволяють ефективно обробляти великі обсяги інформації і швидко реагувати на зміни у мережі.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Довідник по Scapy. *Документація Scapy:* вебсайт. URL: https://scapy.readthedocs.io/en/latest/ (дата звернення: 27.05.2024).
2. Python Network Programming. *Документація Python:* вебсайт. URL: https://docs.python.org/3/howto/sockets.html (дата звернення: 25.05.2024).
3. Документація Wireshark. *Офіційний сайт Wireshark:* вебсайт. URL: https://www.wireshark.org/docs/ (дата звернення: 20.04.2024).
4. Alharbi, Moustafa. Network Analysis using Scapy: монографія. Birmingham: Packt Publishing, 2019. – 245 с.
5. Bluw, J. Mastering Python Networking: монографія. Birmingham: Packt Publishing, 2020. – 550 с.
6. Python підручник. *W3Schools українською:* вебсайт. URL: https://w3schoolsua.github.io/python/index.html#gsc.tab=0 (дата звернення: 22.03.2024).
7. Wireshark Filters. *Cisco Community:* вебсайт. URL: https://community.cisco.com/t5/network-management/understanding-wireshark-filters/td-p/2244984 (дата звернення: 21.04.2024).
8. How to Calculate Project Cost. *ProjectManager:* вебсайт. URL: https://www.projectmanager.com/blog/how-to-calculate-project-cost (дата звернення: 30.04.2024).
9. Аналіз мережевого трафіку для виявлення аномалій та його роль в сфері кібербезпеки. Лобас С.А. ЗБІРНИК тез за матеріалами студентських науково-практичних конференцій в рамках Днів Науки 2024 в Галицькому фаховому коледжі імені В’ячеслава Чорновола (секція «Комп’ютерних технологій» та секція «Фізико-математичних та природничих дисциплін») – Тернопіль: Галицький фаховий коледж імені В’ячеслава Чорновола, 2024. __с.

ДОДАТКИ

Додаток А

Лістинг програми

```
import tkinter as tk
from tkinter import ttk, filedialog
from scapy.all import sniff, IP, TCP, UDP, Ether, wrpcap, rdpcap
import psutil
import threading
import csv

class NetworkTrafficAnalyzer:
    def __init__(self):
        self.capture_interface = ""
        self.interface_window = None
        self.sniffing_thread = None
        self.sniffing_paused = threading.Event()
        self.sniffing_paused.set()  # Initially not paused
        self.captured_packets = []  # To store captured packets

    def open_interface_selection(self):
        if self.interface_window and
self.interface_window.winfo_exists():
            return

        self.interface_window = tk.Toplevel()
        self.interface_window.title("Вибір мережевого інтерфейсу")

        available_interfaces = [interface for interface, addrs in
psutil.net_if_addrs().items()]

        interface_label = tk.Label(self.interface_window, text="Виберіть
мережевий інтерфейс:")
        interface_label.pack()

        interface_combobox = ttk.Combobox(self.interface_window,
values=available_interfaces, state="readonly")
        interface_combobox.pack()
```

```
        def select_interface():
            self.capture_interface = interface_combobox.get()
            self.interface_window.destroy()

        select_button = tk.Button(self.interface_window, text="Вибрати",
command=select_interface)
        select_button.pack()

    def start_analysis(self):
        if not self.capture_interface:
            self.add_to_text("Не вибрано мережевий інтерфейс.\n")
            self.open_interface_selection()
            return

        def analyze_packet(packet):
            self.sniffing_paused.wait()  # Pauses sniffing if event is
clear
            self.captured_packets.append(packet)  # Store captured packet
            self.analyze_packet_details(packet)

        self.sniffing_thread =
threading.Thread(target=self.sniff_packets, args=(analyze_packet,))
        self.sniffing_thread.start()

    def stop_analysis(self):
        if self.sniffing_thread:
            self.sniffing_paused.clear()  # Pause sniffing

    def resume_analysis(self):
        if self.sniffing_thread:
            self.sniffing_paused.set()  # Resume sniffing

    def sniff_packets(self, analyze_packet):
        sniff(prn=analyze_packet, store=0, iface=self.capture_interface,
filter="ip")

    def analyze_packet_details(self, packet):
```

```
        protocol = "Unknown"
        if IP in packet:
            if TCP in packet:
                protocol = "TCP"
            elif UDP in packet:
                protocol = "UDP"
            else:
                protocol = packet[IP].proto
        elif Ether in packet:
            protocol = "Ethernet"

        data = {
            "protocol": protocol,
            "src_mac": packet[Ether].src if Ether in packet else "",
            "dst_mac": packet[Ether].dst if Ether in packet else "",
            "src_ip": packet[IP].src if IP in packet else "",
            "dst_ip": packet[IP].dst if IP in packet else ""
        }
        self.add_to_table(data)

    def add_to_text(self, data):
        if isinstance(data, str):
            result_text.insert(tk.END, data)

    def add_to_table(self, data):
        self.treeview.insert('', 'end', values=(data['protocol'],
data['src_mac'], data['dst_mac'], data['src_ip'], data['dst_ip']))

    def save_data(self):
        file_path = filedialog.asksaveasfilename(defaultextension=".csv",
filetypes=[("CSV files", "*.csv"), ("All files", "*.*")])
        if file_path:
            with open(file_path, mode='w', newline='') as file:
                writer = csv.writer(file)
                writer.writerow(["Протокол", "Джерело MAC", "Призначення
MAC", "Джерело IP", "Призначення IP"])
                for row in self.treeview.get_children():
                    writer.writerow(self.treeview.item(row)['values'])
```

```
            # Save packets in PCAP format as well
            wrpcap(file_path.replace(".csv", ".pcap"),
self.captured_packets)

    def import_data(self):
        file_path = filedialog.askopenfilename(defaultextension=".csv",
filetypes=[("CSV files", "*.csv"), ("All files", "*.*")])
        if file_path:
            with open(file_path, mode='r', newline='') as file:
                reader = csv.reader(file)
                next(reader)  # Skip the header row
                for row in reader:
                    self.add_to_table({
                        "protocol": row[0],
                        "src_mac": row[1],
                        "dst_mac": row[2],
                        "src_ip": row[3],
                        "dst_ip": row[4]
                    })
            # Load packets from PCAP file
            pcap_file_path = file_path.replace(".csv", ".pcap")
            self.captured_packets = rdpcap(pcap_file_path)
            for packet in self.captured_packets:
                self.analyze_packet_details(packet)

analyzer = NetworkTrafficAnalyzer()

window = tk.Tk()
window.title("Аналізатор мережевого трафіку")

# Налаштування сітки для основного вікна
window.rowconfigure(1, weight=1)
window.columnconfigure(0, weight=1)

# Створення меню
menu_bar = tk.Menu(window)

file_menu = tk.Menu(menu_bar, tearoff=0)
```

```
file_menu.add_command(label="Зберегти дані", command=analyzer.save_data)
file_menu.add_command(label="Імпортувати дані",
command=analyzer.import_data)
menu_bar.add_cascade(label="Файл", menu=file_menu)

capture_menu = tk.Menu(menu_bar, tearoff=0)
capture_menu.add_command(label="Почати аналіз",
command=analyzer.start_analysis)
capture_menu.add_command(label="Припинити аналіз",
command=analyzer.stop_analysis)
capture_menu.add_command(label="Відновити аналіз",
command=analyzer.resume_analysis)
menu_bar.add_cascade(label="Захоплення", menu=capture_menu)

window.config(menu=menu_bar)

# Створення заголовків колонок і текстового поля для відображення
результатів
tree_frame = tk.Frame(window)
tree_frame.grid(row=1, column=0, sticky='nsew')

columns = ["Протокол", "Джерело MAC", "Призначення MAC", "Джерело IP",
"Призначення IP"]
analyzer.treeview = ttk.Treeview(tree_frame, columns=columns,
show="headings")

for col in columns:
    analyzer.treeview.heading(col, text=col)
    analyzer.treeview.column(col, anchor='center', stretch=True)

# Додавання горизонтального і вертикального скролбарів
vsb = ttk.Scrollbar(tree_frame, orient="vertical",
command=analyzer.treeview.yview)
vsb.pack(side='right', fill='y')
hsb = ttk.Scrollbar(tree_frame, orient="horizontal",
command=analyzer.treeview.xview)
hsb.pack(side='bottom', fill='x')
```

```
analyzer.treeview.configure(yscrollcommand=vsb.set,
xscrollcommand=hsb.set)
analyzer.treeview.pack(fill=tk.BOTH, expand=True)

# Кнопки для вибору інтерфейсу і запуску аналізу
button_frame = tk.Frame(window)
button_frame.grid(row=0, column=0, pady=10, sticky='ew')

select_interface_button = tk.Button(button_frame, text="Вибрати мережевий
інтерфейс", command=analyzer.open_interface_selection)
select_interface_button.pack(side=tk.LEFT, padx=5)

window.mainloop()
```

Додаток Б

Кошторис витрат на проєктування

Найменування статей витрат	Сума, грн	Обґрунтування
1. Зарплата проєктувальників	113686	Виходячи з кількості виконавців, окладів та місяців участі
2. Відрахування на соціальні потреби	25011	Відрахування становлять 22% від суми заробітної плати
3. Контрагентські роботи і послуги	22737	Вартість послуг субпідрядників, 20% від суми зарплати
4. Витрати на відрядження	15000	Прямий підрахунок передбачуваних витрат на відрядження
5. Інші прямі витрати	56843	Вартість обладнання, матеріалів, 50% від суми заробітної плати
6. Усього прямих витрат	233277	Сума пп. 1-5

7. Накладні витрати	69983	30% від суми прямих витрат
8. Планові накопичення	60652	20% від суми прямих та накладних витрат
9. Усього кошторисна вартість проєкту	364912	Сума прямих, накладних витрат та планових накопичень
10. Податок на додану вартість	72982	20% від кошторисної вартості проєкту
11. Загалом договірна ціна розробки	437894	Сума кошторисної вартості та ПДВ