

**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СУМСЬКИЙ НАЦІОНАЛЬНИЙ АГРАРНИЙ УНІВЕРСИТЕТ
ФАКУЛЬТЕТ ЕКОНОМІКИ І МЕНЕДЖМЕНТУ**

Кафедра кібернетики та інформатики

КВАЛІФІКАЦІЙНА РОБОТА

другий (магістерський) рівень вищої освіти

**на тему: “РОЗРОБКА ВЕБІНФОРМАЦІЙНОЇ
ТЕХНОЛОГІЇ МОНІТОРИНГУ ІНТЕРНЕТ-ТРАФІКУ”**

Виконав:

студент спеціальності

126 «Інформаційні системи та технології»

ОП «Інформаційні системи та технології»

Корецький Василь Андрійович

Керівник:

Братушка Сергій Миколайович

к.ф.-м.н., доцент кафедри кібернетики та інформатики

Рецензент:

Москаленко Альона Сергіївна

Кандидат технічних наук, доцент кафедри комп'ютерних наук
Сумського державного університету

Суми – 2025

СУМСЬКИЙ НАЦІОНАЛЬНИЙ АГРАРНИЙ УНІВЕРСИТЕТ

Факультет

Економіки і менеджменту

Кафедра

Кібернетики та інформатики

Освітній ступінь

другий (магістерський) рівень вищої освіти

Спеціальність

126 «Інформаційні системи та технології»

ЗАТВЕРДЖУЮ:

Зав. кафедри

Ігор ШЕЛЕХОВ

« _ » «__»

2025 р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ ЗДОБУВАЧА ВИЩОЇ ОСВІТИ

Корецького Василя Андрійовича

1. Тема кваліфікаційної роботи: «Розробка вебінформаційної технології моніторингу інтернет-трафіку»

2. База практичного дослідження: ФОП «СПЕКТР-АС», м. Суми

3. Науковий керівник кваліфікаційної роботи: Братушка С.М., к. ф.-м. н., доцент, затверджений наказом по університету «__» _____ 20__ р. № _____

4. Строк подання здобувачем закінченої кваліфікаційної роботи

« 22 » «грудня» 2025р.

5. Вихідні дані до кваліфікаційної роботи:

монографії, фахові наукові статті та тези доповіді міжнародних науково-практичних конференцій з питань моніторингу інтернет-трафіку

6. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити):

- виконати постановку задачі;
- вибрати середовище для реалізації;
- розробити технічне завдання;
- розробити програмний продукт відповідно до постановлених задач;
- розробити інструкцію для користувача

Завдання прийняв до виконання _____

(підпис)

Дата отримання завдання

21 жовтня 2024р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту	Примітка
1	Визначення та затвердження теми дослідження, складання плану та графіку виконання кваліфікаційної роботи	<i>Жовтень 2024</i>	виконано
2	Підбір та аналіз літературних джерел, законодавчої та нормативної бази	<i>Листопад 2024</i>	виконано
3	Узагальнення теоретичного матеріалу з теми дослідження та представлення чорнового варіанту першого розділу кваліфікаційної роботи	<i>Лютий-квітень 2025</i>	виконано
4	Збір та обробка аналітичного матеріалу, узагальнення аналізу застосування досліджуваного питання на підприємстві (переддипломна практика)	<i>02-31 травня 2025</i>	виконано
5	Оформлення теоретичної частини кваліфікаційної роботи, узагальнення аналітичної частини, захист звіту з переддипломної практики	<i>Червень 2025</i>	виконано
6	Розробка напрямів вирішення (удосконалення) досліджуваної проблеми	<i>Липень-Серпень 2025</i>	виконано
7	Завершення рекомендаційної частини (програмної реалізації) кваліфікаційної роботи та подання попереднього варіанту третього розділу керівнику	<i>Вересень 2025</i>	виконано
8	Доопрацювання другого і третього розділів кваліфікаційної роботи	<i>Жовтень 2025</i>	виконано
9	Оформлення кваліфікаційної роботи, та її подання на попередній захист та рецензування	<i>Листопад 2025</i>	виконано
10	Перевірка кваліфікаційної роботи на автентичність	<i>Листопад 2025</i>	виконано
11	Подання закінченої кваліфікаційної роботи та документів до неї до деканату	<i>12 грудня 2025</i>	виконано
12	Захист кваліфікаційної роботи	<i>25-26 грудня 2025</i>	виконано

Здобувач

_____ Василь Корецький

Керівник роботи

_____ Сергій БРАТУШКА

Кваліфікаційна робота допущена до захисту

_____ Світлана ЛУКАШ

АНОТАЦІЯ

Корецький В.А. «Розробка вебінформаційної технології моніторингу інтернет-трафіку»

Кваліфікаційна робота на здобуття другого (магістерського рівня вищої освіти) зі спеціальності 126 «Інформаційні системи та технології» ОП Інформаційні системи та технології, СНАУ, Суми-2025р.

Кваліфікаційна робота присвячена розробці вебінформаційної технології для моніторингу інтернет-трафіку. Метою роботи є створення ефективного інструменту, що допоможе забезпечити моніторинг трафіку в реальному часі. У роботі проведений аналіз аналогів для моніторингу трафіку. Запропоновано концептуальну модель вебінформаційної технології для моніторингу статистики використання трафіку процесами. Для розробки вебзастосунку було використано сучасну мову програмування та бібліотеки Python, такі як flet, Psutil, Time, sqlite3, collections (конкретно deque), Threading та Datetime. Середовищем програмування було обрано PyCharm, забезпечуює швидке та надійне функціонування. Проведено тестування вебінформаційної технології з використанням реальних даних. Результати тестування підтвердили правильність функціонування вебінформаційної технології та відповідність вимогам та очікуванням користувачів. Отримані результати дозволяють зробити висновок про доцільність впровадження розробленого вебінформаційної технології для моніторингу трафіка. Цей застосунок сприятиме покращенню ефективності використання трафіку, а також забезпечить зручний доступ до необхідної інформації.

Ключові слова: Python, Npcap, NDIS, API, NPF, моніторинг.

ANNOTATION

Koretskyi V.A. “Development of a Web Information Technology for Internet Traffic Monitoring”

Qualification thesis for obtaining the second (master's) level of higher education in specialty 126 “Information Systems and Technologies”, Educational Program “Information Systems and Technologies”, SNAU, Sumy – 2025.

The qualification thesis is devoted to the development of a web information technology for monitoring Internet traffic. The aim of the work is to create an effective tool that ensures real-time traffic monitoring. The thesis includes an analysis of existing analogues for traffic monitoring. A conceptual model of a web information technology for monitoring statistics on traffic usage by processes is proposed. To develop the web application, a modern programming language and Python libraries such as flet, psutil, time, sqlite3, collections (specifically deque), threading, and datetime were used. PyCharm was chosen as the development environment, ensuring fast and reliable operation. Testing of the web information technology was conducted using real data. The test results confirmed the correct functioning of the system and its compliance with user requirements and expectations. The obtained results make it possible to conclude that the developed web information technology for traffic monitoring is suitable for practical implementation. This application will contribute to improving the efficiency of traffic usage and will provide convenient access to necessary information.

Keywords: Python, Npcap, NDIS, API, NPF, monitoring.

ЗМІСТ

ВСТУП

РОЗДІЛ 1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Огляд існуючих рішень.....	11
1.2 Аналоги готових продуктів.....	13
1.2.1 Wireshark	13
1.2.2 Zabbix.....	15
1.2.3 Nagios.....	18
1.2.4 Cisco Stealthwatch.....	20
1.2.5 SolarWinds Network Performance Monitor.....	22
1.2.6 Cacti.....	25
1.3 Постановка задачі.....	27

РОЗДІЛ 2 ВИБІР ПРОГРАМНИХ ЗАСОБІВ ТА ПРОЕКТУВАННЯ ВЕБІНФОРМАЦІЙНОЇ ТЕХНОЛОГІЇ МОНІТОРИНГУ ІНТЕРНЕТ- ТРАФІКУ

2.1 Технологія захоплення трафіку прсар.....	29
2.2 Принцип роботи прсар.....	32
2.3 flet.....	36
2.4 time.....	38
2.5 Psutil.....	39
2.6 sqlite3.....	41
2.7 collections.....	42
2.8 threading.....	43
2.9 Засоби розробки програмного забезпечення	44

РОЗДІЛ 3 ПРОЕКТУВАННЯ ТА РЕАЛІЗАЦІЯ ВЕБЗАСТОСУНКУ ДЛЯ МОНІТРОНИГУ ТРАФІКУ ІНТЕРНЕТ-ПРОВАЙДЕРА

3.1 Проектування вебзастосунку.....	51
3.2 Опис функціоналу вебзастосунку.....	54
3.3 Опис програмного забезпечення.....	55

ВИСНОВКИ.....	58
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ	59
ДОДАТКИ	
Додаток А ТЕХНІЧНЕ ЗАВДАННЯ	62
Додаток Б ПЛАНУВАННЯ РОБІТ	67
Додаток В КОПІЇ ПУБЛІКАЦІЙ	74
Додаток Г ЛІСТИНГ ПРОГРАМ.....	83

ВСТУП

Моніторинг і аналіз інтернет-трафіку посідають ключове місце в сучасному управлінні мережами та системами інформаційної безпеки, оскільки саме вони гарантують стабільність роботи, захищеність від загроз і сприяють стратегічному розвитку організацій.

Відстеження трафіку дозволяє своєчасно помічати підозрілу активність та кіберзагрози, зокрема DDoS-атаки, спроби злому, поширення шкідливого ПЗ чи витоки конфіденційних даних. Окрім цього, воно дає змогу визначити, які сервіси або додатки споживають найбільше ресурсів, що є основою для їх оптимізації.

Комплексне дослідження мережевих потоків дозволяє своєчасно реагувати на потенційні загрози, оперативно їх усувати та запобігати несанкціонованому доступу до внутрішніх систем організації. Такий аналіз також дає змогу виявляти проблеми з пропускнуою здатністю та інші технічні несправності, що можуть негативно позначатися на загальній продуктивності мережі.

Таким чином, вивчення сучасних підходів до перехоплення пакетів, обробки мережевих даних та розробки вебінформаційної системи для моніторингу трафіку є безумовно актуальним і важливим напрямом дослідження. Метою роботи є розробка вебінформаційної технології для моніторингу інтернет трафіку.

Об'єкт дослідження – процес розробки вебінформаційної технології для моніторингу інтернет трафіку.

Предметом дослідження є інформаційні технології розробки вебінформаційної технології для моніторингу трафіку.

Задачі дослідження:

- проведення огляду наявних рішень та визначення ключових принципів роботи і проєктування програм, призначених для моніторингу трафіку;
- розроблення архітектурної моделі вебінформаційної технології для моніторингу мережевого трафіку;

- реалізацію спроектованої архітектури вебінформаційної технології моніторингу із застосуванням сучасних технологій;
- тестування програмного продукту та оцінювання його ефективності.

Для досягнення мети використані такі методи дослідження:

- 1) логіко-аналітичні методи – використовувалися для формулювання цілей, вибору інструментів їх досягнення, обґрунтування отриманих результатів та підготовки висновків;
- 2) теоретичні методи – застосовані для вивчення й узагальнення наукових джерел, присвячених темі дослідження;
- 3) принципи системного аналізу – використані під час опрацювання предметної області.

Наукова новизна роботи полягає у створенні інтегрованої системи моніторингу мережевої активності, яка об'єднує потокове збирання даних, багатовимірний аналіз процесів, багаторівневе архівування трафіку та реальновідтворювану графічну візуалізацію. Запропонована архітектура дозволяє проводити комплексний поведінковий аналіз мережевих процесів і формувати часові моделі навантаження системи, що відрізняється від існуючих рішень: універсальністю і простотою розгортання; є засобом довготривалого моніторингу інтернет-трафіку без використання хмарних сервісів

Практична значимість розробленої програми полягає у забезпеченні користувача ефективним засобом контролю, аналізу та діагностики мережевої активності в режимі реального часу та на основі довготривалих статистичних даних. Система дозволяє оперативно виявляти аномальні процеси, аналізувати поведінку мережі залежно від часу доби, здійснювати аудит безпеки та отримувати достовірну історичну інформацію без використання зовнішніх сервісів. Програму може бути використано у практичній діяльності системних адміністраторів, навчальному процесі, дослідницькій роботі, в домашніх умовах у навчальних аудиторіях, на робочих комп'ютерах та невеликих офісах.

Дослідження структуроване у три розділи кваліфікаційної роботи. Перший розділ присвячений огляду існуючих рішень і аналогів продуктових систем для

моніторингу мережевого трафіку. У другому розділі здійснено вибір програмних інструментів для реалізації вебінформаційної технології моніторингу. Третій розділ містить опис розробки створеної вебінформаційної технології для контролю мережевого трафіку

Апробація матеріалів дослідження здійснювалася на міжнародній науково-практичній конференції “Інформаційні технології та автоматизація 2025”, 30-31 жовтня 2025 р., Одеса: Одеський національний технологічний університет Інститут комп’ютерної інженерії, автоматизації, робототехніки та програмування ім. П. Н. Платонова.

РОЗДІЛ 1

АНАЛІТИЧНИЙ ОГЛЯД

1.1 Огляд існуючих рішень

Моніторинг і аналіз інтернет-трафіку передбачає перехоплення мережових пакетів, їх опрацювання та подальшу інтерпритацію. Основна мета — зрозуміти поведінку мережі, своєчасно виявляти несправності, підвищувати ефективність роботи інфраструктури та забезпечувати її захищеність.

Етап збору даних охоплює фіксацію трафіку з подальшим аналізом. Для цього застосовують пасивний і активний моніторинг. Пасивний контроль полягає у перехопленні копій мережових пакетів без втручання в їхній рух. Це може здійснюватися через network taps або port mirroring, коли комутатор дублює трафік на виділений порт для спостереження. Активний же моніторинг передбачає створення додаткового трафіку для перевірки стану мережі й пошуку несправностей, наприклад, за допомогою ping-запитів, трасування маршруту чи тестових завантажень.

Подальший аналіз даних спрямований на виявлення закономірностей, відхилень та конкретних проблемних зон. Він може включати фільтрацію та декодування пакетів для виділення суттєвих параметрів - IP-адрес, номерів портів, використовуваних протоколів тощо. У межах пакетної інспекції окремі системи проводять детальну перевірку структури та вмісту кожного пакета, тоді як інші агрегують дані для формування загальної статистичної картини, наприклад, визначення загального обсягу трафіку, найактивніших вузлів або найбільш ресурсомістких додатків.

Інтерпретація та подання результатів включає такі елементи:

- 1) візуалізація — представлення отриманих даних у вигляді графіків, діаграм, таблиць чи інших наочних форм, що суттєво спрощує їх розуміння;

- 2) система сповіщень — можливість налаштовувати автоматичні повідомлення про важливі події або аномальні ситуації в мережі в режимі реального часу;
- 3) звітування — формування регулярних звітів, які узагальнюють стан мережевої інфраструктури, фіксують виявлені проблеми та пропонують рекомендації щодо їх усунення.

Етап реагування та оптимізації охоплює:

- 1) прийняття рішень і дія — на основі отриманих результатів адміністратори можуть оперативно втручатися для поліпшення роботи мережі або усунення виявлених недоліків;
- 2) налаштування обладнання — зміну конфігурацій мережевих пристроїв з метою підвищення продуктивності, стабільності чи рівня захисту;
- 3) удосконалення політик безпеки — актуалізацію правил доступу, блокування підозрілих IP-адрес та обмеження доступу до потенційно небезпечних ресурсів;
- 4) планування та розподіл ресурсів — раціональну організацію пропускну здатності та інших мережевих ресурсів, зокрема пріоритизацію критично важливих сервісів.

Для виконання зазначених завдань системи моніторингу та аналізу інтернет-трафіку застосовують широкий спектр технологій і методів, серед яких:

- 1) Sniffers — програмні інструменти, що перехоплюють та зберігають мережеві пакети з метою подальшого опрацювання;
- 2) NetFlow/SFlow/IPFIX — спеціалізовані протоколи, призначені для збору статистики про трафік безпосередньо на мережевих пристроях;
- 3) DPI (Deep Packet Inspection) — метод глибокого аналізу пакетів, що дає змогу досліджувати не лише заголовки, а й їхній вміст.

Отже, системи моніторингу та аналізу трафіку є ключовими засобами для гарантування безпеки, підвищення ефективності та стабільності мережевої інфраструктури.

1.2 Аналоги готових продуктів

1.2.1 Wireshark

Wireshark — це один із найпоширеніших інструментів для дослідження мережевого трафіку, який дозволяє здійснювати захоплення та детальний розбір пакетів даних. Незважаючи на наявність інших спеціалізованих засобів для моніторингу та аналізу мережі, Wireshark зберігає лідируючі позиції завдяки своїй широкій функціональності та безкоштовній доступності.

Цей інструмент дає змогу фіксувати трафік у реальному часі з мережевих інтерфейсів локального комп'ютера або зберігати захоплені дані для подальшого розгляду. Програма підтримує великий спектр типів інтерфейсів, серед яких Ethernet, Wi-Fi, Bluetooth тощо. Wireshark розпізнає тисячі мережевих протоколів — від HTTP, FTP, TCP та UDP до DNS і SSL/TLS — автоматично і відображає їхню структуру та вміст у зручній для аналізу формі.

Процес роботи Wireshark можна розділити на кілька етапів:

Wireshark підключається до обраного мережевого інтерфейсу (Ethernet, Wi-Fi або інших) і перехоплює пакети, що проходять через нього. Для цього використовується спеціалізована бібліотека (libpcap на Linux та WinPcap/npcap на Windows), яка працює на низькому рівні мережевого стеку. Завдяки цьому інструмент отримує доступ до «сирого» трафіку, що дозволяє аналізувати його до обробки прикладними програмами.

Кожен пакет складається з кількох рівнів: Ethernet → IP → TCP/UDP → додаткові протоколи прикладного рівня (HTTP, DNS тощо). Wireshark розпізнає ці рівні та відображає їх у структурованому вигляді, що дає можливість дослідити окремі елементи пакета

Оскільки обсяг трафіку може бути великим, Wireshark дозволяє застосовувати точні фільтри для виділення потрібних пакетів. Це можуть бути, наприклад:

- 1) пакети від або до конкретної IP-адреси (`ip.addr == 192.168.1.10`);

- 2) пакети конкретного порту (`tcp.port == 443`);
- 3) лише HTTP-запити (`http.request`).

Власна мова фільтрів Wireshark дозволяє гнучко і швидко відокремлювати потрібні дані для подальшого аналізу.

Після відбору пакетів можливий детальний аналіз їхніх складових:

- 1) заголовки протоколів (IP, TCP, UDP, DNS та ін.);
- 2) прапорці сесій (SYN, ACK, FIN);
- 3) корисне навантаження (payload);
- 4) час передачі, затримки, відновлення TCP-сесій.

Wireshark також дозволяє відтворити повну TCP-сесію, що дає повне уявлення про обмін даними між вузлами.

Зібрані дані можна відображати у вигляді графіків, діаграм і таблиць для зручного аналізу:

- 1) розподіл трафіку за протоколами
- 2) швидкість передачі та затримки
- 3) найактивніші вузли та додатки

Це допомагає виявляти аномалії, такі як підозрілі IP-адреси, DoS-атаки або витоки даних, а також оцінювати продуктивність мережі та пріоритизацію ресурсів.

Wireshark також надає можливість роботи через зручний графічний інтерфейс (див. рисунок 1.1), що дозволяє користувачу застосовувати всі функції програми безпосередньо з робочого середовища. Інтерфейс є інтуїтивно зрозумілим і підтримує відображення захоплених даних у різних форматах — таблицях, графіках, діаграмах. Додатково реалізоване кольорове підсвічування, яке полегшує візуальне розрізнення різних типів мережевого трафіку та прискорює процес аналізу.

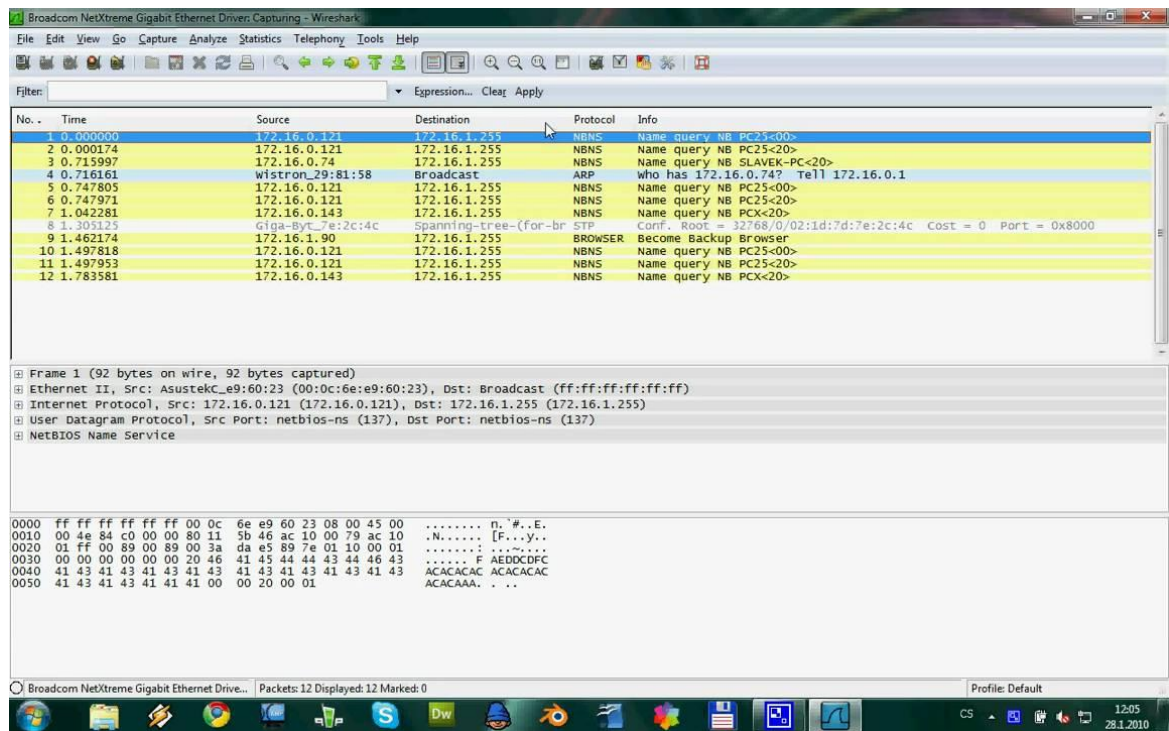


Рисунок 1.1 – Графічний інтерфейс «Wireshark»

«Wireshark»

Джерело [19]

1.2.2 Zabbix

Zabbix — це високофункціональна платформа з відкритим вихідним кодом, призначена для моніторингу мережевої інфраструктури та програмних сервісів. Вона дає можливість контролювати роботу серверів, мережевого обладнання, віртуальних середовищ, баз даних та інших компонентів у режимі реального часу, забезпечуючи адміністратора детальними відомостями про їхній стан і продуктивність.

Zabbix отримує інформацію про стан об'єктів за допомогою різних методів:

- 1) Агентський збір — спеціальні програми (Zabbix Agent) встановлюються на сервери або пристрої та передають метрики про використання CPU, пам'яті, дискового простору, процесів, стан служб тощо;

- 2) Безагентський збір — дані отримуються через протоколи SNMP, IPMI, JMX або шляхом опитування API, що дозволяє контролювати мережеві пристрої та сторонні сервіси без встановлення додаткового програмного забезпечення;
- 3) Моніторинг логів та подій — Zabbix може відслідковувати файли журналів і події системи для виявлення аномалій чи помилок.

Завдяки різноманітним методам збору даних система здатна охоплювати як серверні ресурси, так і мережеву інфраструктуру, забезпечуючи комплексний контроль.

Після отримання метрик Zabbix виконує їхню обробку, порівнює з встановленими пороговими значеннями та виявляє аномалії або несправності. Система дозволяє:

- 1) формувати тригери для визначення критичних станів;
- 2) агрегувати дані для оцінки тенденцій використання ресурсів;
- 3) відслідковувати залежності між компонентами інфраструктури, що допомагає прогнозувати можливі збої.

Це дозволяє не лише виявляти поточні проблеми, а й планувати оптимізацію ресурсів і запобігати аварійним ситуаціям.

Zabbix забезпечує оперативне інформування адміністратора про критичні події:

- 1) Сповіщення в реальному часі через електронну пошту, SMS, месенджери або інтеграції з іншими системами;
- 2) Автоматичні дії — запуск скриптів або команд для усунення проблем без участі людини (наприклад, перезапуск служби чи корекція конфігурації);
- 3) Ескалація подій — можливість налаштування кількох рівнів оповіщень для різних категорій проблем.

Це дозволяє значно скоротити час реагування на інциденти та підвищити стабільність роботи інфраструктури.

Zabbix перетворює отримані дані на наочну інформацію:

- 1) Графіки та діаграми використання ресурсів та навантаження (див. Рисунок 1.2);

- 2) Дашборди з оглядом стану всіх компонентів мережі та серверів;
- 3) Регулярні звіти про роботу системи, виявлені проблеми та рекомендації щодо їх усунення.

Ця візуалізація допомагає легко оцінити продуктивність, знайти аномалії та ефективно планувати подальшу роботу мережі.

Переваги Zabbix:

- Zabbix є безкоштовним, і його код доступний для зміни та покращення;
- можливість адаптації до різних сценаріїв моніторингу;
- підходить для моніторингу як малих, так і великих ІТ-інфраструктур;
- підтримка інтеграції з іншими системами та сервісами.

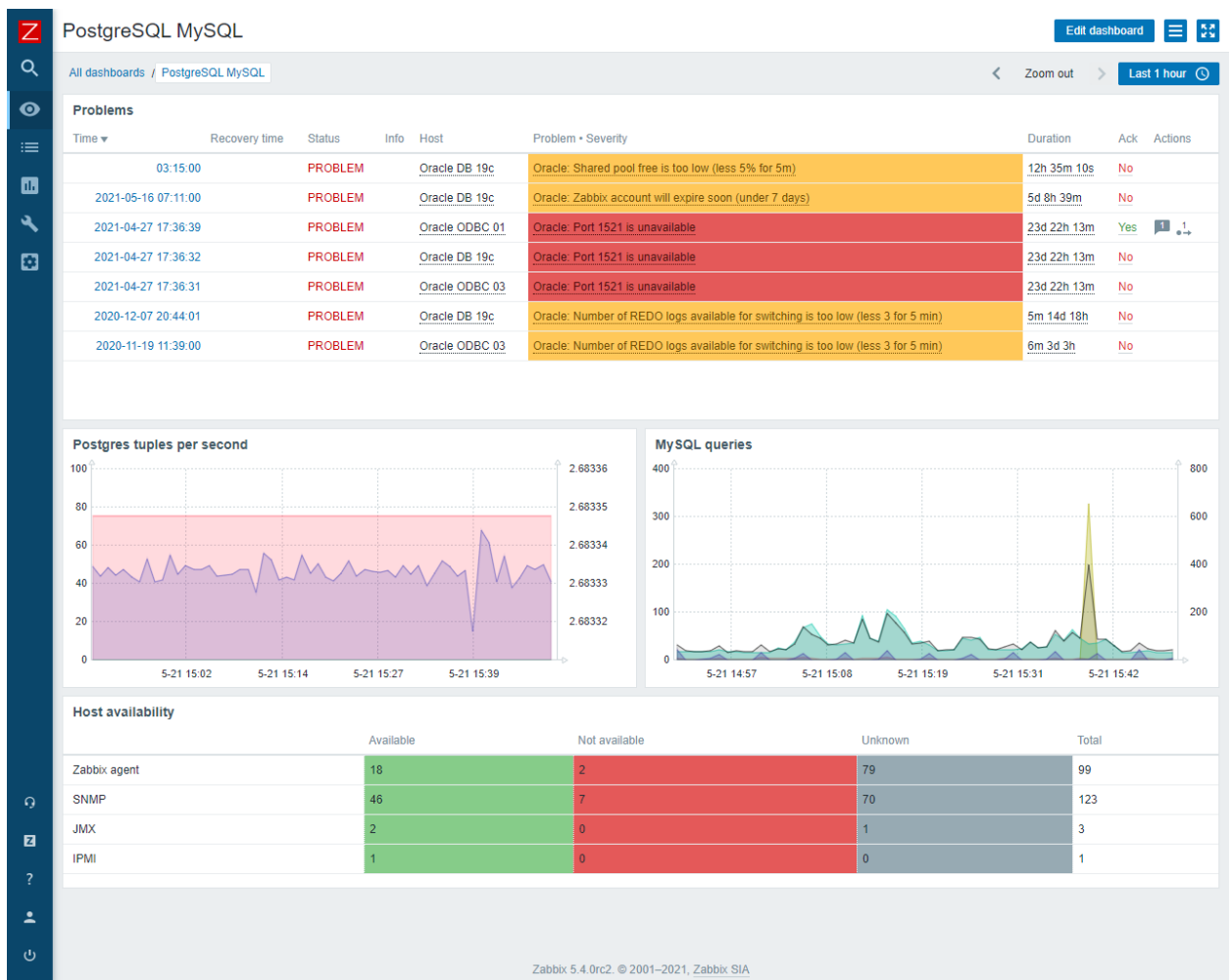


Рисунок 1.2 – Приклад графічного інтерфейсу «Zabbix»

«Zabbix»

Джерело [20]

1.2.3 Nagios

Nagios — одна з найбільш відомих платформ для контролю мережевих ресурсів і сервісів. Вона дозволяє здійснювати моніторинг працездатності та продуктивності елементів ІТ-інфраструктури, швидко виявляти несправності й оперативно сповіщати адміністратора про можливі збої.

Nagios збирає інформацію про стан інфраструктури кількома способами:

- 1) Агентський моніторинг — на сервери або пристрої встановлюються спеціальні плагіни, які передають метрики про завантаження CPU, використання пам'яті, стан дисків, роботу служб та процесів.
- 2) Безагентський моніторинг — Nagios використовує стандартні протоколи, такі як SNMP, HTTP, ICMP (ping) або SSH, щоб перевіряти стан мережевих пристроїв і сервісів без встановлення додаткових агентів.
- 3) Перевірка сервісів — система опитує різні сервіси, такі як веб-сервери, бази даних або поштові системи, щоб переконатися в їхній доступності та коректній роботі.

Ці методи дозволяють охоплювати як сервери, так і мережеві компоненти, забезпечуючи комплексний моніторинг.

Отримані дані обробляються з урахуванням встановлених порогів і правил:

- 1) Nagios порівнює поточні значення метрик з допустимими межами і визначає критичні стани;
- 2) система відстежує залежності між об'єктами, щоб правильно інтерпретувати несправності (наприклад, якщо сервер недоступний, Nagios перевіряє, чи це не проблема мережевого вузла);
- 3) дані агрегуються для формування статистики та історичних графіків використання ресурсів (див. Рисунок 1.3).

Таким чином, Nagios дозволяє не лише фіксувати проблеми, а й планувати профілактичні заходи для забезпечення стабільності інфраструктури.

Nagios оперативно повідомляє адміністратора про проблеми:

- 1) Сповіщення через електронну пошту, SMS або інтеграції з зовнішніми системами;
- 2) Автоматичні дії — запуск скриптів або команд для усунення несправностей без участі людини;
- 3) Ескалація проблем — налаштування різних рівнів оповіщень залежно від серйозності інциденту.

Це дозволяє скоротити час реагування та підвищити ефективність адміністрування мережі та серверів.

Nagios надає інструменти для наочного представлення стану інфраструктури:

- 1) Графіки та діаграми навантаження серверів і мережевих пристроїв;
- 2) Дашборди з оглядом стану критичних систем;
- 3) Регулярні звіти про доступність сервісів, виконання SLA та історію інцидентів.

Ця візуалізація дозволяє легко відстежувати продуктивність та своєчасно виявляти аномалії.

Переваги Nagios:

- вільний доступ до коду, що дозволяє налаштовувати його під потреби організації;
- можливість моніторингу практично будь-яких систем та сервісів;
- велика база користувачів і розробників, яка сприяє швидкому розвитку та підтримці продукту.

Недоліки Nagios:

- для початківців може бути складно налаштувати і підтримувати;
- за замовчуванням веб-інтерфейс може здаватися застарілим і незручним без додаткових розширень.

Nagios є ефективним інструментом для моніторингу IT-інфраструктури, завдяки своїй гнучкості та багатій екосистемі плагінів, але його налаштування може вимагати значних технічних зусиль.

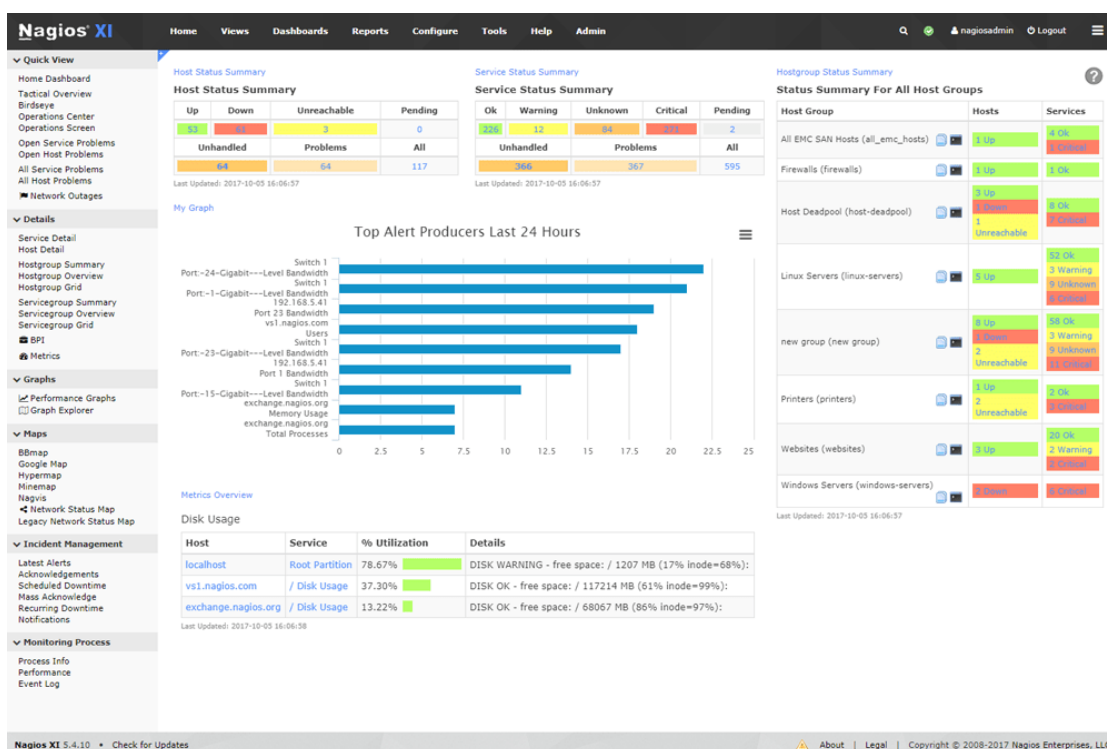


Рисунок 1.3 – Приклад графіку програми Nagios

«Nagios»

Джерело: [21]

1.2.4 Cisco Stealthwatch

Cisco Stealthwatch — це розгорнута платформа мережевої безпеки, що застосовує поведінкову аналітику для виявлення та протидії загрозам у мережевому середовищі. Вона забезпечує глибоку видимість трафіку, автоматично визначає аномальні дії та пропонує інструменти для детального розслідування інцидентів, що дає можливість оперативно виявляти й нейтралізувати кібератаки.

Stealthwatch отримує інформацію про мережеві потоки з різних джерел:

- 1) NetFlow, sFlow та IPFIX — протоколи, які дозволяють збирати детальну статистику про обмін даними між вузлами мережі без необхідності перехоплення всіх пакетів;
- 2) SPAN / TAP-портів — для збору «сирого» трафіку в реальному часі, що забезпечує глибокий аналіз і точність;

- 3) Інтеграція з мережевими пристроями — комутатори, маршрутизатори та файрволи надають інформацію про з'єднання та обсяги переданих даних.

Цей підхід дозволяє створювати повну картину потоків трафіку без надмірного навантаження на мережу.

Cisco Stealthwatch виконує детальний аналіз потоків даних для виявлення відхилень від нормальної поведінки мережі:

- 1) Поведенковий аналіз (behavioral analytics) — система формує базову модель нормальної роботи мережі та визначає аномалії, наприклад, нетипові обсяги трафіку або підозрілі комунікації;
- 2) Ідентифікація загроз — Stealthwatch здатен виявляти DDoS-атаки, рух шкідливого ПЗ, внутрішні загрози та несанкціоновані доступи;
- 3) Кореляція подій — дані з різних джерел поєднуються для точнішої оцінки загроз та прискорення реагування.

Аналіз потоків забезпечує швидке виявлення загроз навіть у великих корпоративних мережах, де перехоплення всіх пакетів було б неефективним.

Stealthwatch автоматично повідомляє адміністраторів про критичні події:

- 1) Реальні сповіщення про аномалії та загрози через електронну пошту, інтеграції з SIEM або інші системи управління;
- 2) Автоматизоване реагування — можливість запускати скрипти чи політики блокування підозрілих з'єднань або IP-адрес;
- 3) Пріоритизація загроз — система оцінює серйозність інцидентів, щоб фокусувати ресурси на найкритичніших подіях.

Це дозволяє швидко локалізувати та нейтралізувати загрози без втрати часу на ручну перевірку.

Stealthwatch надає розширені інструменти для візуалізації та аналітики:

- 1) Дашборди та графіки — відображення потоків трафіку, топ-хостів, активних протоколів і зон підвищеного ризику (див. Рисунок 1.4);
- 2) Звіти про безпеку та продуктивність — історія подій, аналіз тенденцій і рекомендації щодо оптимізації мережі;

3) Карта мережевих потоків — наочне представлення зв'язків між вузлами та сегментами мережі.

Такий підхід дозволяє швидко оцінювати стан мережі, виявляти аномалії та приймати стратегічні рішення щодо безпеки та оптимізації.

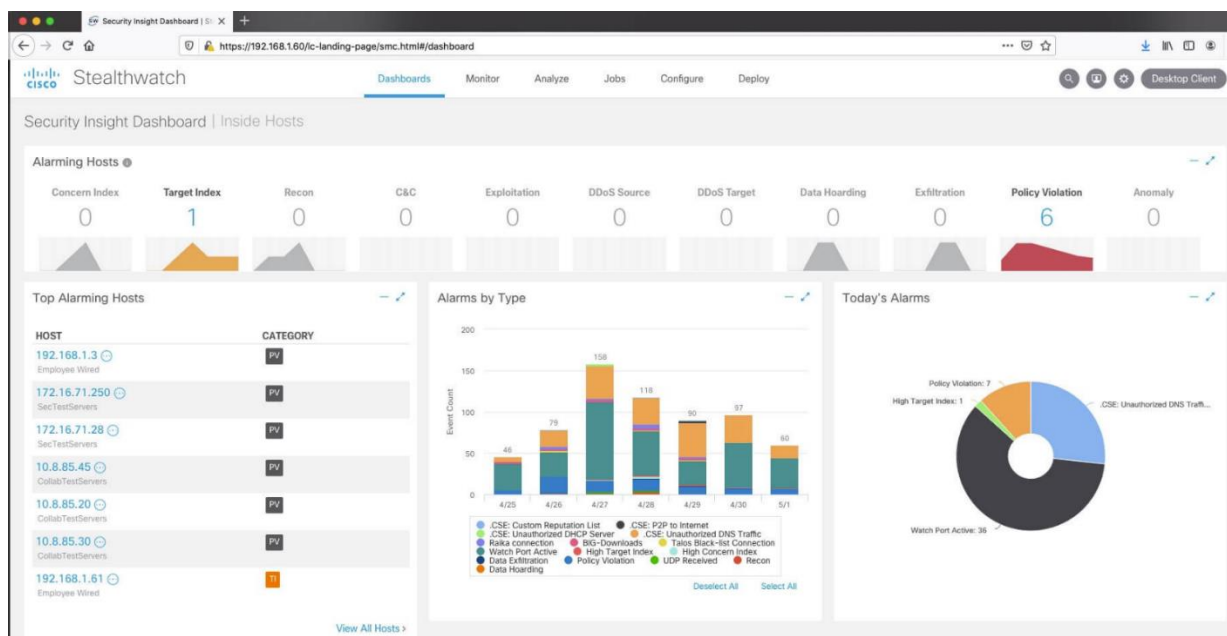


Рисунок 1.4 – Приклад дашбордів та графіків Cisco Stealthwatch

«Cisco Stealthwatch»

Джерело: [22]

1.2.5 SolarWinds Network Performance Monitor

SolarWinds Network Performance Monitor (NPM) — це інструмент моніторингу мережевої інфраструктури, який забезпечує повну прозорість, аналіз і контроль за продуктивністю мережі. Завдяки широкому набору інструментів та інтегрованих механізмів NPM допомагає швидко виявляти, діагностувати й усувати мережеві несправності, забезпечуючи стабільну та ефективну роботу всієї інфраструктури.

SolarWinds NPM отримує інформацію про мережеву активність та стан пристроїв різними методами:

- 1) SNMP (Simple Network Management Protocol) — використовується для збору метрик стану комутаторів, маршрутизаторів, серверів і мережевих пристроїв;
- 2) Flow-протоколи (NetFlow, sFlow, J-Flow) — дозволяють отримувати статистику трафіку між вузлами мережі без перехоплення всіх пакетів, визначати обсяг переданих даних та активність окремих пристроїв;
- 3) Ping і ICMP-моніторинг — для перевірки доступності пристроїв та визначення часу відповіді;
- 4) WMI/CLI — збір даних з серверів Windows або мережевих пристроїв через командний інтерфейс.

Цей комплексний підхід дозволяє охоплювати як фізичну, так і віртуальну мережеву інфраструктуру, забезпечуючи всебічний контроль.

Після збору даних SolarWinds NPM виконує їхній аналіз для оцінки стану мережі та виявлення потенційних проблем:

- 1) Моніторинг пропускної здатності — визначення пікових навантажень і вузьких місць;
- 2) Виявлення аномалій і проблем — система порівнює поточні значення метрик із нормальними або історичними даними, визначаючи відхилення;
- 3) Кореляція даних — аналіз зв'язків між пристроями та сегментами мережі для розуміння причин затримок або збоїв.

Цей етап дозволяє передбачати проблеми, оптимізувати маршрути передачі та забезпечувати стабільну роботу мережі.

SolarWinds NPM автоматично повідомляє адміністраторів про критичні події та дозволяє швидко реагувати:

- 1) Сповіщення в реальному часі через електронну пошту, SMS або інтеграції з іншими системами управління;
- 2) Автоматичні дії — запуск скриптів або зміна конфігурацій для усунення проблем без ручного втручання;
- 3) Пріоритизація інцидентів — система визначає найсерйозніші проблеми, щоб зосередити увагу на критично важливих подіях.

Це скорочує час реагування та підвищує ефективність адміністрування мережевої інфраструктури.

SolarWinds NPM надає зручні інструменти для відображення та аналізу даних:

- 1) Дашборди та графіки — візуалізація стану мережевих пристроїв, пропускної здатності, затримок і використання ресурсів (див. Рисунок 1.5);
- 2) Карти мережі — наочне представлення топології, зв'язків між пристроями та сегментами мережі;
- 3) Регулярні звіти — історія продуктивності, тенденції використання ресурсів та рекомендації щодо оптимізації.

Ці можливості допомагають швидко оцінювати стан мережі, виявляти вузькі місця та планувати поліпшення продуктивності.

SolarWinds Network Performance Monitor це потужний інструмент для забезпечення високої продуктивності і надійності мережі, який підходить як для середніх, так і великих підприємств завдяки своїй функціональності та зручному інтерфейсу.

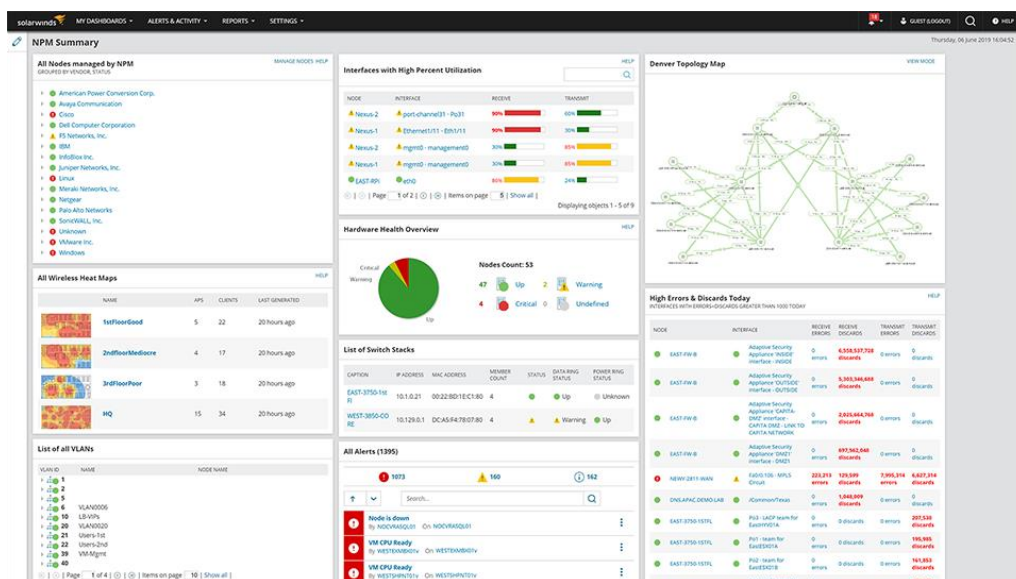


Рисунок 1.5 – Приклад інтерфейсу SolarWinds Network Performance Monitor
«SolarWinds Network Performance Monitor»

Джерело: [23]

1.2.6 Cacti

Cacti це інструмент з відкритим кодом для моніторингу мережевих ресурсів та трафіку, який спеціалізується на створенні графіків із використанням системи RRDTool. Cacti дозволяє адмініструвати та візуалізувати дані про продуктивність мережі та системи, використовуючи інтерфейс, орієнтований на простоту та ефективність.

Cacti використовує різні методи отримання інформації про стан інфраструктури:

- 1) SNMP (Simple Network Management Protocol) — основний метод, що дозволяє опитувати комутатори, маршрутизатори, сервери та інші мережеві пристрої;
- 2) Використання скриптів та плагінів — для збору додаткових метрик, які не підтримуються стандартними SNMP-об'єктами;
- 3) Опитування через командний інтерфейс (CLI) — збір інформації з серверів або спеціалізованих мережевих пристроїв, якщо SNMP недоступний.

Регулярне опитування дозволяє створювати історію використання ресурсів і контролювати стан мережевих пристроїв у часі.

Після збору даних Cacti обробляє їх і формує структуровані графіки:

- 1) Агрегація метрик — обчислення середніх значень, пікових навантажень та тенденцій використання ресурсів;
- 2) Візуалізація трендів — відображення змін навантаження на CPU, пам'ять, пропускну здатність каналів та інші ресурси у вигляді лінійних графіків;
- 3) Виявлення аномалій — графіки дозволяють візуально помітити відхилення від нормальної роботи пристроїв або пікові навантаження.

Це забезпечує наочний контроль та аналіз ефективності роботи мережевої та серверної інфраструктури.

Хоча основний фокус Cacti на графічній візуалізації, система підтримує інтеграцію з зовнішніми сповіщувальними механізмами:

- 1) Сповіщення про критичні стани — можливість інтегрувати Cacti з системами електронної пошти, SMS або зовнішніми скриптами для оповіщення адміністраторів;
- 2) Автоматичне оновлення даних — регулярне опитування забезпечує своєчасне виявлення проблем;
- 3) Розширюваність через плагіни — дозволяє додавати додаткові функції, включно зі сповіщеннями та обробкою подій.
- 4) Cacti надає потужні інструменти для графічного представлення даних:
- 5) Лінійні та гістограмні графіки — для відображення навантаження на сервери, пропускної здатності каналів та інших метрик;
- 6) Дашборди та панелі перегляду — огляд стану мережі та сервісів у реальному часі;
- 7) Історичні звіти — накопичення та аналіз даних за тривалий період, що допомагає прогнозувати потреби у ресурсах.
- 8) Візуалізація дозволяє адміністраторам швидко оцінювати продуктивність, виявляти аномалії та планувати модернізацію або оптимізацію мережевих ресурсів.

Основні можливості Cacti:

- збирання даних за допомогою SNMP, скриптів або команд для генерації графіків;
- автоматична генерація графіків на основі зібраних даних (див Рисунок 1.6);
- використання RRDTool для зберігання даних у форматі кругових баз даних, що дозволяє ефективно зберігати та візуалізувати великі обсяги даних;
- можливість створення та використання шаблонів для хостів, графіків, джерел даних;
- полегшує налаштування моніторингу схожих об'єктів у великих середовищах;
- підтримка простого мережевого управління протоколом SNMP для збору метрик з мережевих пристроїв;

- підтримка масштабування для моніторингу великої кількості пристроїв і джерел даних;
- веб-інтерфейс для налаштування та керування графіками, шаблонами та джерелами даних;
- можливість створення персоналізованих переглядів графіків.

Састі є корисним інструментом для організацій, яким потрібна ефективна система для збору та візуалізації даних про мережеву і серверну продуктивність, надаючи простий у використанні інтерфейс та широкі можливості для налаштування і розширення.

Device Description	Hostname	ID	Graphs	Data Sources	Status	In State	Uptime	Poll Time	Current (ms)	Average (ms)	Availability	Created
Cacti Server	localhost	1	4	5	Up	N/A	N/A	0.1	0	0	100 %	2020-09-06 21:43:06
Central NAS	192.168.11.105	56	12	22	Up	120	42	0.26	0.35	1.15	99.36 %	2020-09-06 21:43:06
IP Printer	192.168.11.174	55	22	137	Up	137	54	0.65	1.04	1.8	99.81 %	2020-09-06 21:43:06
vhost01	192.168.11.201	46	12	19	Up	120	4	0.38	1.45	1.61	99.99 %	2020-09-06 21:43:06
vhost02	192.168.11.202	45	12	19	Up	120	4	0.34	0.56	0.94	99.99 %	2020-09-06 21:43:06
vhost03	192.168.11.203	44	12	19	Up	120	4	0.24	0.9	2.09	99.98 %	2020-09-06 21:43:06
vhost04	192.168.11.204	43	12	19	Up	120	4	0.26	1.01	0.76	100 %	2020-09-06 21:43:06
vhost05	192.168.11.205	42	12	19	Up	120	4	0.33	0.83	1.25	99.99 %	2020-09-06 21:43:06
vhost06	192.168.11.206	41	12	19	Up	120	4	0.39	0.74	0.79	100 %	2020-09-06 21:43:06
vhost07	192.168.11.207	40	12	19	Up	267	4	0.4	0.52	1.06	98.93 %	2020-09-06 21:43:06
vhost08	192.168.11.208	39	12	19	Up	120	4	0.19	0.89	1.24	99.99 %	2020-09-06 21:43:06
vhost09	192.168.11.209	38	12	19	Up	267	4	0.15	0.7	1.07	98.93 %	2020-09-06 21:43:06
vhost10	192.168.11.210	37	12	19	Up	120	4	0.22	0.77	0.77	100 %	2020-09-06 21:43:06
vhost11	192.168.11.211	36	12	19	Up	120	4	0.09	2.61	1.01	99.98 %	2020-09-06 21:43:06
vhost12	192.168.11.212	35	12	19	Up	120	4	0.32	1.14	1.09	99.99 %	2020-09-06 21:43:06
vhost13	192.168.11.213	34	12	19	Up	120	4	0.25	2.63	1.05	99.98 %	2020-09-06 21:43:06
vhost14	192.168.11.214	33	12	19	Up	267	4	0.26	3.09	1.02	98.93 %	2020-09-06 21:43:06
vhost15	192.168.11.215	32	12	19	Up	120	4	0.31	1.11	0.93	99.99 %	2020-09-06 21:43:06

Рисунок 1.6 – Приклад графічного інтерфейсу Састі

«Састі»

Джерело [24]

1.3 Постановка задачі

Нижче представлено узагальнену постановку задачі та технічне обґрунтування, що слугує основою для розробки вебінформаційної системи, призначеної для моніторингу мережевого трафіку.

Функціональні вимоги:

- відображення поточних мережевих підключень із прив'язкою до відповідних ідентифікаторів процесів (PID);

- показ інформації про назву процесу, обсяг завантаженого та вивантаженого трафіку, а також швидкість завантаження та відвантаження в реальному часі;
- відображення загальної кількості пакетів, захоплених під час моніторингу;
- відображення обсягу захопленого трафіку за обраний період часу.

Технічні вимоги:

- реалізація меню для вибору мережевих інтерфейсів;
- таблиця для фіксації статистики використання трафіку;
- база даних для збереження історії використаних ресурсів;
- графічне представлення захопленого трафіку у вигляді таблиць для зручного аналізу.

РОЗДІЛ 2

ВИБІР ПРОГРАМНИХ ЗАСОБІВ ТА ПРОЕКТУВАННЯ ВЕБІНФОРМАЦІЙНОЇ ТЕХНОЛОГІЇ ДЛЯ МОНІТОРИНГУ ТРАФІКУ

2.1 Технологія захоплення трафіку нрсар

Nrpsar — це сучасна високопродуктивна бібліотека для захоплення та передачі мережових пакетів, яка є потужною заміною застарілій бібліотеці WinPcap. Вона розроблена спеціально для Windows і володіє рядом переваг та особливостей, що робить її популярним вибором серед мережових адміністраторів, розробників і дослідників.

Захоплення мережового трафіку на Windows без використання таких спеціалізованих рішень, як Nrpsar або WinPcap, супроводжується численними обмеженнями та проблемами, що суттєво ускладнює ефективне виконання цієї задачі.

Основні причини необхідності спеціалізованого програмного забезпечення для захоплення мережового трафіку на Windows:

- 1) Windows розділяє виконання коду на різні рівні безпеки (кільця): режим користувача (User Mode) та режим ядра (Kernel Mode). Доступ до низькорівневих операцій, таких як безпосереднє захоплення мережових пакетів, обмежений режимом ядра;
- 2) Суворий контроль доступу до апаратних ресурсів і мережових стеків;
- 3) Стандартний Windows API не надає прямого доступу до пакетів на рівні низькорівневих стеків протоколів;
- 4) Платформа Windows Sockets (WinSock) забезпечує інтерфейс для мережових комунікацій, але працює на високому рівні і не дозволяє перехоплювати або аналізувати трафік на рівнях L2 або L3 моделі OSI;
- 5) Доступ до NDIS-пакетів у користувацькому просторі без спеціальних драйверів обмежений;

- 6) Підтримка RAW Sockets, яка дає можливість захоплення необроблених пакетів, обмежена в Windows, особливо після Windows XP SP2, де доступ обмежено з міркувань безпеки;
- 7) Вбудовані утиліти Windows для діагностики мережі, наприклад netsh trace, можуть захоплювати трафік, але не забезпечують такої гнучкості і потужності обробки пакетів, як Npcap;
- 8) Стандартні засоби не дозволяють ефективно фільтрувати трафік на рівні пакетів або застосовувати правила BPF (Berkeley Packet Filter), які є стандартом для аналізу мереж;
- 9) Без спеціалізованих драйверів обробка мережевих пакетів не може бути настільки ефективною, оскільки стандартні засоби не здатні обробляти великий обсяг даних із високою швидкістю.

Wireshark це один з найпоширеніших інструментів для аналізу мережевого трафіку. Він покладається на Npcap/WinPcap для захоплення трафіку (див. рисунок 2.1).

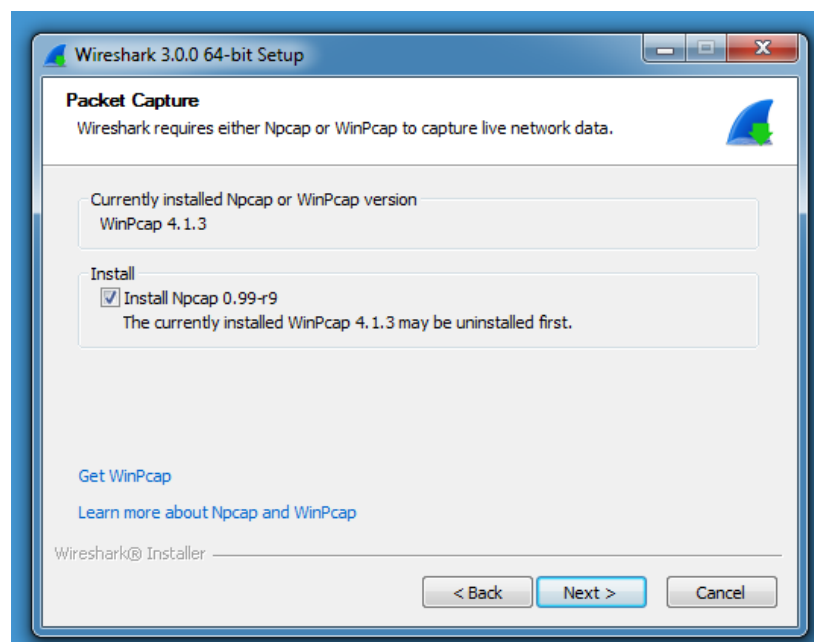


Рисунок 2.1 – Інсталяція Wireshark

Джерело: запропоновано автором

Без спеціалізованого драйвера взаємодія з мережевими інтерфейсами на низькому рівні неможлива. Npcap дозволяє обходити обмеження стандартної моделі безпеки Windows та стандартних API, надаючи прямий доступ до мережеских пакетів на низькому рівні, що необхідно для ефективного аналізу трафіку. Використання таких драйверів, як Npcap, є критично важливим для завдань, що потребують детального перехоплення та аналізу мережеских даних на Windows.

Основні функції та переваги Npcap:

- 1) Висока продуктивність — Npcap оптимізована для ефективного захоплення та передачі мережеских пакетів, що особливо актуально для роботи в високошвидкісних мережах.
- 2) Сумісність із сучасними ОС — бібліотека постійно оновлюється та підтримується для роботи з останніми версіями Windows, включаючи Windows 10 та Windows Server 2019.
- 3) Гнучкість режимів роботи — Npcap може функціонувати як у привілейованому (адміністраторському), так і в непривілейованому режимі, що підвищує безпеку та зручність її використання в різних сценаріях.
- 4) Режим моніторингу (Monitor Mode) — дозволяє захоплювати пакети, не адресовані конкретному хосту, що особливо корисно для аналізу бездротових мереж.
- 5) Доступ до низькорівневих даних — бібліотека забезпечує роботу на канальному рівні, надаючи можливість детального аналізу структури мережеских пакетів.
- 6) Енергоефективність — вдосконалене управління енергоспоживанням зменшує навантаження на систему під час тривалого захоплення трафіку.

Програмний інтерфейс Npcap використовується широким спектром мережеских інструментів для аналізу, діагностики, забезпечення безпеки та моніторингу. До класичних програм, які спираються на Npcap, належать:

- 1) аналізатори мереж і протоколів;
- 2) мережеві монітори;

- 3) реєстратори трафіку;
- 4) генератори трафіку;
- 5) користувачські мости та маршрутизатори;
- 6) системи виявлення вторгнень у мережу;
- 7) мережеві сканери;
- 8) інструменти для забезпечення безпеки.

Nrpsar приймає та передає пакети незалежно від протоколів хост-системи, таких як TCP/IP. Це означає, що бібліотека не може блокувати, фільтрувати або змінювати трафік, згенерований іншими програмами на тому ж комп'ютері — вона лише «перехоплює» пакети, що проходять через мережу. Через це Nrpsar не забезпечує функціональність, необхідну для застосунків типу шейперів трафіку, планувальників QoS або персональних брандмауерів.

2.2 Принцип роботи nrpsar

NDIS (Network Driver Interface Specification) — це стандарт, що регламентує взаємодію між мережевим адаптером та драйверами протоколів. Головна функція NDIS полягає в тому, щоб виконувати роль проміжного шару, який забезпечує можливість надсилання та отримання мережевих пакетів драйверами протоколів без залежності від конкретного адаптера або операційної системи.

По-перше, система, що здійснює захоплення пакетів, повинна мати можливість отримувати доступ до сирих мережевих даних, минаючи стандартний стек протоколів операційної системи. Для цього необхідний компонент, що працює на рівні ядра та взаємодіє безпосередньо з драйверами мережевого інтерфейсу. У нашому випадку таку роль виконує драйвер Netgroup Packet Filter, який забезпечує як базові можливості (перехоплення й ін'єкцію пакетів), так і розширені — програмований фільтр трафіку та механізм моніторингу. Фільтраційний модуль дозволяє обмежувати обсяг захоплюваних даних визначеною частиною трафіку, тоді як моніторинговий двигун забезпечує простий, але потужний інструмент для отримання статистичних показників.

По-друге, система захоплення повинна надавати інтерфейс, доступний для прикладного програмного забезпечення. Npcap реалізує це через дві бібліотеки — `packet.dll` та `wpcap.dll`.

`packet.dll` надає низькорівневий API, який дозволяє взаємодіяти з функціональністю ядрового драйвера незалежно від версії Windows.

`wpcap.dll` пропонує більш високорівневий набір засобів захоплення, сумісних із *libpcap* — відомою Unix-орієнтованою бібліотекою для перехоплення трафіку. Використання цієї бібліотеки дає змогу здійснювати захоплення пакетів без прив'язки до конкретного мережевого обладнання чи ОС.

NDIS підтримує три основні типи мережевих драйверів: драйвери NIC, проміжні драйвери та транспортні драйвери (драйвери протоколів).

Драйвери NIC безпосередньо керують мережевими платами (Network Interface Cards, NIC). Вони підключаються до апаратного забезпечення на нижньому рівні та надають зверху інтерфейс, через який верхні рівні системи можуть надсилати пакети, обробляти переривання, скидання та зупинку NIC, а також читати й змінювати експлуатаційні параметри драйвера. Драйвери NIC можуть бути реалізовані як мініпорти або як застарілі повні драйвери.

Драйвери мініпорту виконують лише специфічні апаратні операції, необхідні для керування NIC, зокрема передачу та прийом даних. Усі стандартні функції низького рівня для драйверів NIC, такі як синхронізація, забезпечуються NDIS. Мініпорт не відстежує прив'язки пакетів — він передає їх до NDIS, який гарантує доставку пакетів до відповідних протоколів.

Повні драйвери NIC були розроблені для виконання всіх операцій, пов'язаних із апаратним забезпеченням, а також синхронізації та постановки пакетів у чергу, які зазвичай обробляє NDIS. Вони, наприклад, зберігають власну інформацію про прив'язку, що дозволяє коректно обробляти отримані дані.

Проміжні драйвери виконують роль посередника між драйверами верхнього рівня, такими як драйвери протоколів, і мініпортами. Для драйвера верхнього рівня проміжний драйвер виглядає як мініпорт, а для мініпорта — як драйвер протоколу.

Проміжні драйвери можуть наслідуватися один на одного, хоча таке накладання може впливати на продуктивність системи.

Головною метою проміжного драйвера є трансляція мережевих медіа між застарілим драйвером протоколу та мініпортом, який керує NIC для нового типу медіа, невідомого драйверу протоколу. Наприклад, проміжний драйвер може перетворювати дані з протоколу LAN на протокол ATM. Проміжний драйвер не взаємодіє з програмами користувацького режиму, працюючи виключно з іншими драйверами NDIS.

Драйвер протоколу реалізує стек мережевих протоколів, таких як IPX/SPX або TCP/IP, і надає свої послуги через одну або декілька мережевих карт. Він обслуговує клієнтів прикладного рівня на верхньому краю та підключається до одного або кількох драйверів NIC чи проміжних драйверів NDIS на нижньому рівні.

NPF реалізований як драйвер протоколу. Хоча це не оптимальне рішення з точки зору продуктивності, такий підхід забезпечує гнучку незалежність від рівня MAC і дає повний доступ до необробленого мережевого трафіку.

На Рисунку 2.2 показано структуру NDIS із двома стеками захоплення пакетів, прив'язаними до одного мережевого адаптера.

Взаємодія NPF з операційною системою зазвичай відбувається асинхронно. Це означає, що драйвер надає набір функцій зворотного виклику, які система виконує, коли NPF потрібно здійснити певну операцію. NPF експортує ці функції для основних операцій вводу/виводу програм, включно з відкриттям, закриттям, читанням та записом даних.

Взаємодія з NDIS також асинхронна: події, такі як надходження нового пакету, повідомляються через функції зворотного виклику NPF. Крім того, взаємодія з NDIS та драйвером NIC здійснюється за допомогою неблокуючих викликів: коли NPF звертається до функції NDIS, виклик повертається негайно, а після завершення обробки NDIS повідомляє NPF через відповідний зворотний виклик.

Драйвер надає зворотні виклики для будь-яких низькорівневих операцій, таких як передача пакетів або читання/запис параметрів NIC.

На Рисунку 2.3 наведено структуру стеку захоплення пакетів із зверненням до драйвера NPF.

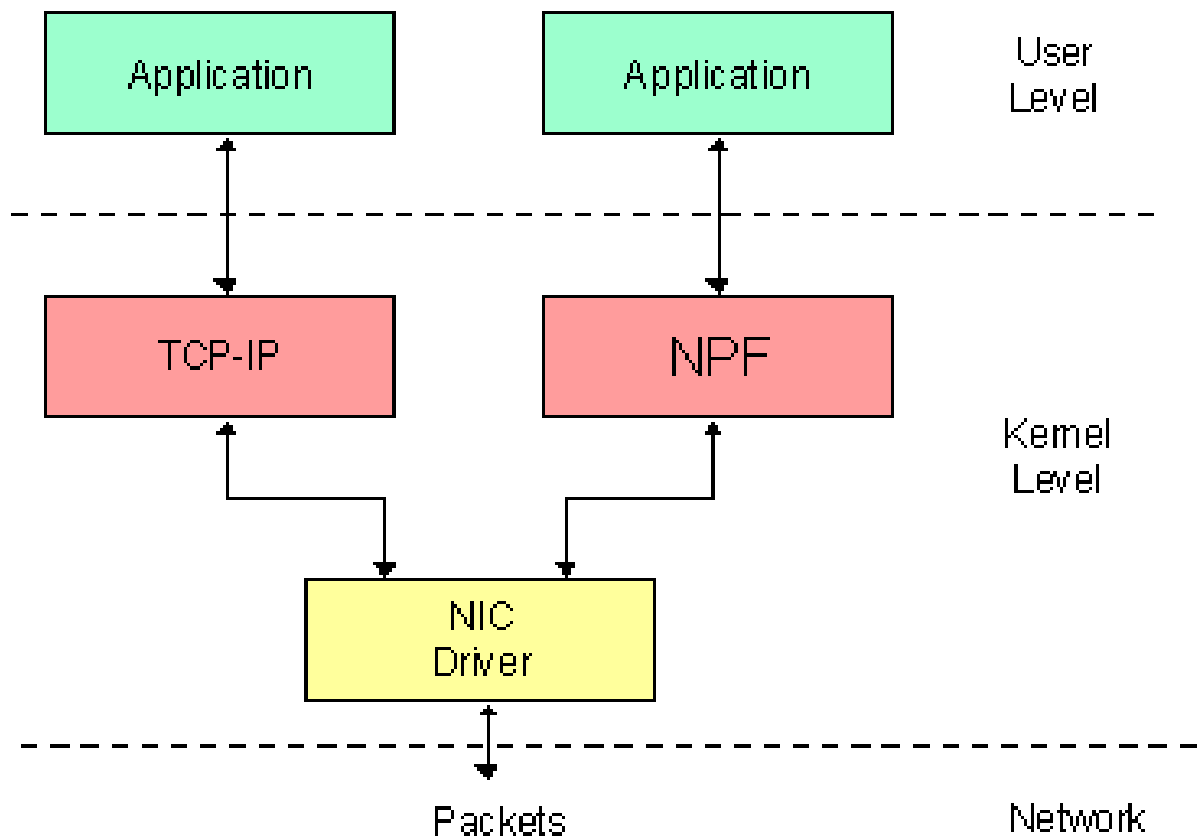


Рисунок 2.2 – Структури NDIS з двома стеками захоплення пакетів

«NPF driver internals manual»

Джерело: [25]

NPF підтримує виконання різноманітних операцій, таких як захоплення пакетів, моніторинг трафіку, запис даних на диск та впровадження пакетів у мережу.

Найважливішою функцією NPF є захоплення пакетів. Під час цієї операції драйвер «перехоплює» пакети через мережевий інтерфейс і передає їх у недоторканому вигляді програмам у користувацькому просторі для подальшого аналізу.

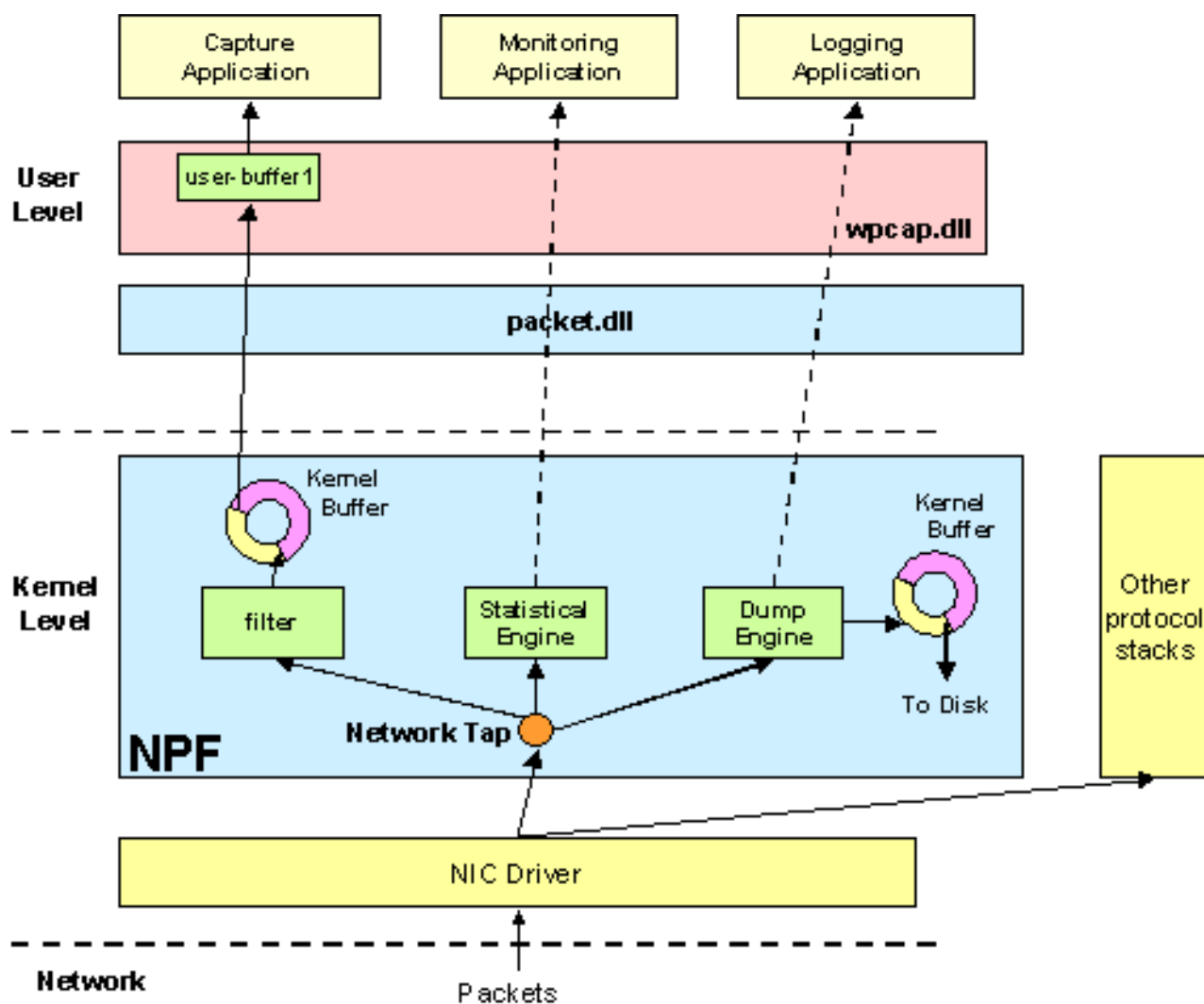


Рисунок 2.3 – Структура стеку захоплення пакетів
«NPF driver internals manual»

Джерело: [25]

2.3 Flet

Flet — це сучасна Python-бібліотека для створення інтерактивних веб- та десктопних застосунків без необхідності глибоких знань фронтенд-розробки. Вона дозволяє швидко будувати інтерфейси користувача, використовуючи звичний синтаксис Python, і автоматично транслює їх у веб-клієнт або десктопний додаток.

Flet дозволяє будувати UI на основі компонентів:

- 1) Компоненти включають кнопки, поля введення, таблиці, графіки та інші елементи;

- 2) Користувач може комбінувати компоненти в контейнери та створювати складні макети;
- 3) Підтримується гнучке розміщення та стилізація елементів без написання HTML/CSS.

Flet спрощує процес розробки UI, дозволяючи зосередитися на логіці застосунку, а не на фронтенді.

Flet забезпечує роботу з подіями та взаємодією користувача:

- 1) Кнопки, поля введення та інші елементи підтримують обробку подій;
- 2) Можна підключати функції Python як обробники кліків, зміни значень або інших дій користувача;
- 3) Зміни стану елементів UI автоматично оновлюються на клієнтській стороні.
- 4) Бібліотека дозволяє легко додавати інтерактивність без складного JS-коду, роблячи UI реактивним.

Flet підтримує різні платформи:

- 1) Веб-браузери (через WebSockets);
- 2) Десктопні додатки на Windows, macOS і Linux;
- 3) Мобільні пристрої (експериментальна підтримка через веб-клієнт).

Flet дозволяє створювати один код для різних платформ, що економить час і ресурси розробки.

Flet тісно інтегрується з екосистемою Python:

- 1) Можна використовувати будь-які Python-бібліотеки для обробки даних, машинного навчання, баз даних тощо;
- 2) UI і логіка програми пишуться виключно на Python;
- 3) Забезпечується простий зв'язок між компонентами UI та бізнес-логікою застосунку.

Розробник може повністю працювати у Python, не потребуючи знань фронтенду, і легко інтегрувати зовнішні бібліотеки.

2.4 time

`time` — це стандартна бібліотека Python для роботи з часом і таймерами. Вона дозволяє отримувати поточний час, вимірювати інтервали, затримувати виконання програм і виконувати інші операції, пов'язані з часом.

Бібліотека `time` надає функції для отримання поточного часу:

- 1) `time.time()` повертає кількість секунд з початку епохи (Unix Epoch, 1 січня 1970 року);
- 2) `time.localtime()` перетворює час у структурований локальний час;
- 3) `time.gmtime()` повертає час у форматі UTC.

Ці функції дозволяють точно визначити поточний час та конвертувати його у зручні формати для подальшого використання.

`time` дозволяє тимчасово призупиняти виконання програми:

- 1) `time.sleep(seconds)` зупиняє виконання на заданий інтервал у секундах;
- 2) Можна використовувати дробові значення для більш точних затримок.
- 3) Функція `sleep` корисна для створення пауз, реалізації таймерів або регулювання частоти виконання циклів у програмах.

Бібліотека дозволяє точно вимірювати час виконання операцій:

- 1) `time.perf_counter()` повертає максимально точний таймер для вимірювання інтервалів;
- 2) `time.process_time()` вимірює час, витрачений лише процесором на виконання програми.

Ці інструменти дозволяють аналізувати продуктивність програм та оптимізувати обчислювальні процеси.

`time` надає можливості перетворення часу у зручні формати:

- 1) `time.strftime(format, t)` перетворює час на рядок у заданому форматі;
- 2) `time.strptime(string, format)` перетворює рядок у структурований час.

Форматування часу спрощує роботу з датами та забезпечує сумісність із різними системами та протоколами.

2.5 Psutil

Psutil це бібліотека Python, яка надає інтерфейс для збору інформації про системні ресурси і їх моніторинг. Вона дозволяє отримувати дані про використання процесора, пам'яті, дискових ресурсів, мережевих інтерфейсів і процесів. psutil є потужним інструментом для розробників, системних адміністраторів і аналітиків, які потребують інформації про стан системи для моніторингу, оптимізації та діагностики.

psutil дозволяє отримувати детальну інформацію про активні процеси:

- 1) Перелік усіх процесів у системі та їхні PID;
- 2) Використання CPU, пам'яті, відкриті файли та дескриптори сокетів;
- 3) Створення об'єктів процесів для подальшого управління (завершення, зміна пріоритету).

psutil спрощує відстеження процесів та ресурсів, що дозволяє ефективно управляти роботою системи та програм.

Бібліотека надає детальну інформацію про основні ресурси:

- 1) `psutil.cpu_percent()` і `psutil.cpu_times()` для відстеження завантаження CPU;
- 2) `psutil.virtual_memory()` та `psutil.swap_memory()` для аналізу використання оперативної пам'яті;
- 3) `psutil.disk_usage()` і `psutil.disk_io_counters()` для контролю стану дисків.

Можна швидко визначати стан системи, виявляти перевантаження та оптимізувати ресурси.

psutil дозволяє отримувати інформацію про мережеві з'єднання та трафік:

- 1) Перелік відкритих мережевих з'єднань і сокетів;
- 2) Лічильники переданих і отриманих байтів для кожного інтерфейсу;
- 3) Виявлення активних з'єднань і перевірка використання портів.

Це робить psutil корисним інструментом для моніторингу мережевої активності та діагностики проблем із з'єднаннями.

psutil підтримує всі основні операційні системи: Windows, macOS, Linux та інші UNIX-подібні системи.

- 1) Бібліотека надає однаковий API для різних платформ;
- 2) Легко інтегрується з Python-скриптами та сторонніми бібліотеками для збору даних або побудови систем моніторингу.

Один код працює на різних системах без змін, що забезпечує універсальність і зручність використання.

Основні можливості psutil:

- отримання загальної і пер-процесорної завантаженості;
- збір інформації про частоту процесора;
- використання фізичної (оперативної) пам'яті;
- інформація про swap (віртуальну) пам'ять;
- інформація про дискові розділи і файлові системи;
- вхідний/вихідний дисковий трафік;
- дані про використання дискового простору;
- збір статистики по мережевих інтерфейсах;
- дані про вхідний/вихідний мережевий трафік;
- інформація про мережеві з'єднання (TCP/UDP);
- інформація про всі активні процеси;
- можливість зупинки, відновлення, завершення процесів;
- збір метрик про процеси: використання CPU, пам'яті, відкриті файли і сокети;
- інформація про температуру, батарею (де підтримується).

Особливості psutil:

- працює на різних операційних системах (Windows, macOS, Linux, FreeBSD, SunOS);
- простий інтерфейс для доступу до системних метрик;
- підтримка безлічі функцій для різних аспектів системного моніторингу.

Обмеження psutil:

- деякі функції можуть вимагати прав адміністратора (root) на системі;
- надмірне використання psutil може вплинути на продуктивність системи, особливо при частому виклику функцій.

Psutil це універсальна бібліотека для моніторингу і збору інформації про системні ресурси в Python. Вона забезпечує доступ до важливої інформації про стан системи, що робить її незамінною для розробки скриптів для управління системами, моніторингу ресурсів і діагностики проблем.

2.6 sqlite3

sqlite3 — це стандартна бібліотека Python для роботи з базами даних SQLite. Вона дозволяє створювати, читати, змінювати та видаляти дані у вбудованих реляційних базах без необхідності встановлювати окремий сервер баз даних.

sqlite3 дозволяє легко підключатися до бази даних:

- 1) `sqlite3.connect("filename.db")` створює або відкриває базу даних у файлі;
- 2) Після підключення створюється об'єкт `connection`, через який виконується вся робота з базою;
- 3) Підтримується контекстний менеджер для автоматичного закриття з'єднання.

Просте та швидке підключення до файлу бази даних без додаткових налаштувань.

Бібліотека дозволяє виконувати будь-які SQL-запити:

- 1) `cursor.execute()` для створення таблиць, вставки, оновлення та видалення даних;
- 2) Підтримка параметризованих запитів для безпечної роботи з користувацькими даними;
- 3) `cursor.fetchall()` і `cursor.fetchone()` для отримання результатів запитів.
- 4) sqlite3 дозволяє працювати з даними в базі на рівні SQL, забезпечуючи високу гнучкість.

sqlite3 підтримує роботу з транзакціями:

- 1) За замовчуванням зміни не зберігаються автоматично; використовується `connection.commit()` для підтвердження змін;
- 2) `connection.rollback()` дозволяє відкотити транзакцію у випадку помилки;

3) Це забезпечує цілісність даних і контроль над змінами.

Можна надійно керувати змінами в базі та уникати втрати даних через помилки.

Бібліотека дозволяє безпосередньо працювати з типами Python:

- 1) Підтримуються int, float, str, bytes, а також None;
- 2) Дані автоматично конвертуються між SQLite і Python;
- 3) Можна визначати власні адаптери та конвертери для складних типів.
- 4) Зручно зберігати та отримувати дані у форматі Python без додаткових конвертацій.

sqlite3 не вимагає окремого сервера та працює на всіх популярних ОС:

- 1) Windows, macOS, Linux;
- 2) Підходить для скриптів, невеликих веб- та десктопних застосунків;
- 3) Легка інтеграція з іншими бібліотеками Python, наприклад, pandas.
- 4) Один і той самий код працює на різних платформах без додаткових налаштувань, що робить роботу простою і універсальною.

2.7 collections

collections — це стандартна бібліотека Python, яка надає розширені структури даних, доповнюючи стандартні типи списків, словників, кортежів і множин. Вона дозволяє ефективніше зберігати та обробляти дані завдяки спеціалізованим контейнерам.

deque (double-ended queue) — це двостороння черга для ефективного додавання та видалення елементів:

- 1) Можна швидко додавати або видаляти елементи з обох кінців (append, appendleft, pop, popleft);
- 2) Підходить для реалізації черг, стеків та обмежених буферів;
- 3) Має високу продуктивність для великих колекцій, де стандартний список буде повільним.

deque зручний для структур даних, що часто змінюються на початку або в кінці.

2.8 threading

threading — це стандартна бібліотека Python для роботи з потоками (threads). Вона дозволяє виконувати кілька завдань одночасно в межах одного процесу, керувати потоками та синхронізувати доступ до спільних ресурсів.

threading дозволяє легко створювати паралельні потоки:

- 1) Потоки створюються через клас Thread, в який передається функція або метод для виконання;
- 2) thread.start() запускає потік асинхронно;
- 3) Можна запускати кілька потоків одночасно, щоб виконувати різні завдання паралельно.

threading дозволяє виконувати кілька завдань одночасно, не блокуючи основний потік програми.

Бібліотека надає механізми для контролю доступу до спільних ресурсів:

- 1) Lock — простий замок, який гарантує, що лише один потік використовує ресурс у певний момент;
- 2) RLock — рекурсивний замок, який можна захоплювати декілька разів у межах одного потоку;
- 3) Event, Condition, Semaphore — більш складні механізми синхронізації та взаємодії потоків.

Синхронізація потоків дозволяє уникати конфліктів при доступі до спільних ресурсів і забезпечує безпечну роботу програми.

threading дозволяє контролювати стан потоків:

- 1) thread.join() дозволяє основному потоку чекати завершення іншого потоку;
- 2) thread.is_alive() перевіряє, чи працює потік;
- 3) Потоки можна позначати як демонські (daemon=True), щоб вони завершувалися разом із головним потоком.

Можна ефективно контролювати виконання та завершення потоків у програмі.

Потоки застосовуються для:

- 1) Паралельної обробки даних;
- 2) Роботи з мережею або файловими операціями без блокування основного потоку;
- 3) Реалізації таймерів і періодичних завдань;
- 4) Підвищення продуктивності у програмах з I/O-затримками.

Потоки дозволяють оптимізувати виконання завдань та підвищити відгук програми без складної асинхронної логіки.

2.9 Засоби розробки програмного забезпечення

Для розробки програмного забезпечення було обрано мову програмування Python. Вона є однією з найпопулярніших мов для роботи з інтернет-трафіком завдяки простому та зрозумілому синтаксису, великій кількості готових бібліотек і активній спільноті розробників. Python надає широкий спектр інструментів для ефективного захоплення, аналізу та обробки мережевих пакетів, що робить його ідеальним вибором для створення мережевих застосунків.

Python — високорівнева мова програмування загального призначення, яка відзначається зрозумілим та лаконічним синтаксисом, великою кількістю готових бібліотек і доступністю для програмістів різного рівня досвіду.

Основні характеристики Python:

- Python має простий і зрозумілий синтаксис, що сприяє читабельності коду. Наприклад, використовується відступ (індентифікація) для визначення блоків коду;
- Python є інтерпретованою мовою;
- Python підтримує динамічну типізацію, де типи змінних визначаються під час виконання;

- Мова пропонує високий рівень абстракції над апаратними та системними ресурсами, що дозволяє зосередитися на логіці вирішення завдань;
- Python підтримує об'єктно-орієнтоване, процедурне та функціональне програмування.

Переваги Python:

- простота і зрозумілість;
- широка стандартна бібліотека;
- підтримка різних парадигм програмування;
- велика і активна спільнота.

Недоліки Python:

- порівняно низька швидкодія для завдань, що вимагають високої продуктивності;
- глобальна блокування інтерпретатора, що обмежує продуктивність багатопоточних додатків.

Для створення програмного забезпечення було обрано середовище розробки PyCharm. PyCharm — це інтегроване середовище розробки для мови Python, розроблене компанією JetBrains. Воно надає широкий набір інструментів для написання коду, налагодження, тестування та управління проектами на Python (див. рисунок 2.4). PyCharm відзначається розширеними функціями, які роблять процес розробки більш ефективним і зручним [13].

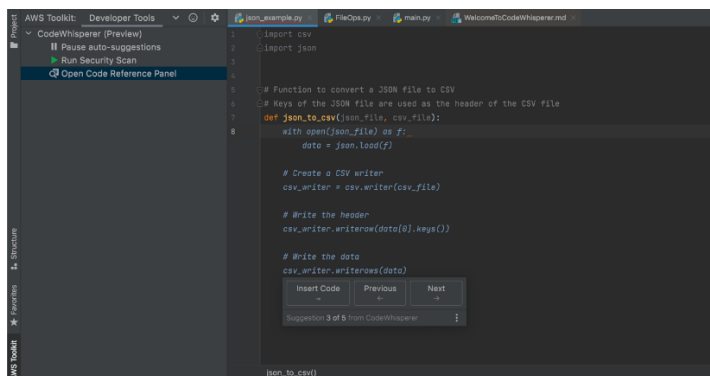


Рисунок 2.4 – Середовище розробки PyCharm

Джерело: запропоновано автором

Основні особливості PyCharm:

- підтримка підсвічування коду Python і інших мов;
- PyCharm автоматично перевіряє код на наявність помилок і дає рекомендації (див. рисунок 2.5);
- пропозиції автодоповнення коду на основі контексту (див. рисунок 2.6);
- зручна навігація по проекту, включаючи перехід до визначення, пошук використань і швидке переміщення між файлами (див. рисунок 2.7);
- інструменти для рефакторингу коду, включаючи перейменування, зміну підпису функцій і розділення класів (див. рисунок 2.8);
- інструменти для підвищення продуктивності;
- PyCharm виконує аналіз коду і дає рекомендації для покращення;
- підтримка шаблонів і фрагментів коду для швидкого написання часто використовуваних конструкцій;
- можливість налаштування середовища розробки, включаючи підтримку різних стилів кодування;
- вбудована система підтримки Git, SVN, Mercurial та інших систем контролю версій;
- візуальні інструменти для порівняння файлів і вирішення конфліктів;
- підтримка повноцінного налагодження з точками зупинки, покроковим виконанням та переглядом значень змінних (див. рисунок 2.9);
- інструменти для профілювання продуктивності додатків;
- інтеграція з популярними базами даних (PostgreSQL, MySQL, Oracle тощо), включаючи можливості перегляду, редагування і виконання SQL-запитів;
- підтримка роботи з Jupyter Notebook, що дозволяє виконувати інтерактивні обчислення і візуалізувати результати прямо в PyCharm;
- підтримка створення і управління віртуальними середовищами (venv, Conda) для ізоляції залежностей проектів;
- інтеграція з популярними веб-фреймворками, такими як Django, Flask, FastAPI.

- підсвічування синтаксису, автодоповнення і налагодження коду для веб-технологій;
- інтеграція з науковими бібліотеками, такими як NumPy, Pandas, Matplotlib (див. рисунок 2.10);
- інтерактивний режим консолі з підтримкою Python (див. рисунок 2.11).

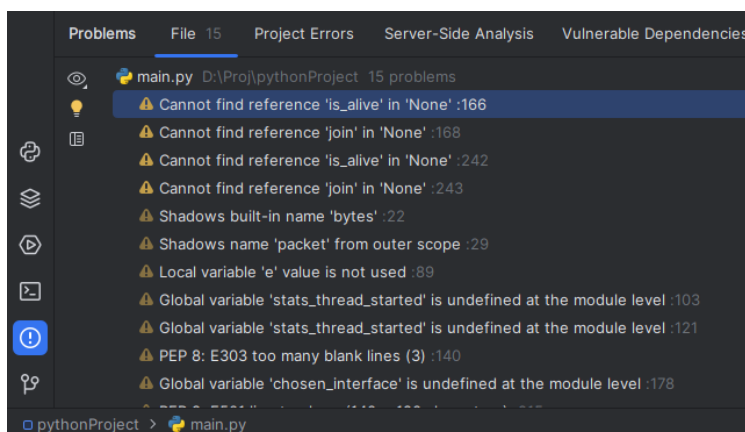


Рисунок 2.5 – Автоматична перевірка коду на наявність помилок і надання рекомендацій

Джерело: запропоновано автором

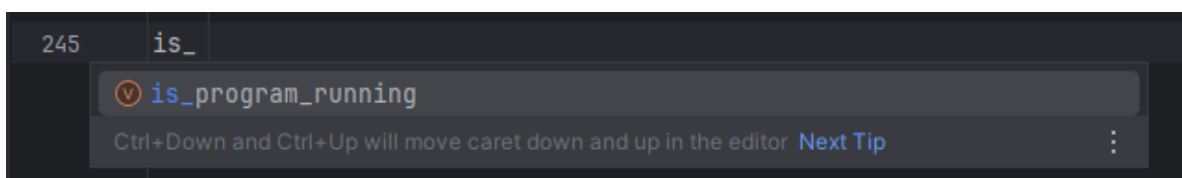


Рисунок 2.6 – Рекомендація на основі контексту

Джерело: запропоновано автором

Основною причиною вибору PyCharm став його розширений редактор коду. Серед ключових можливостей: інтелектуальне автодоповнення, підсвічування синтаксису, перевірка помилок у реальному часі та швидке їх виправлення. PyCharm дозволяє швидко переходити до визначень, використань, класів, методів

та файлів, а також підтримує розширений рефакторинг, включно з перейменуванням і витягом змінних, методів, класів та інших структур.

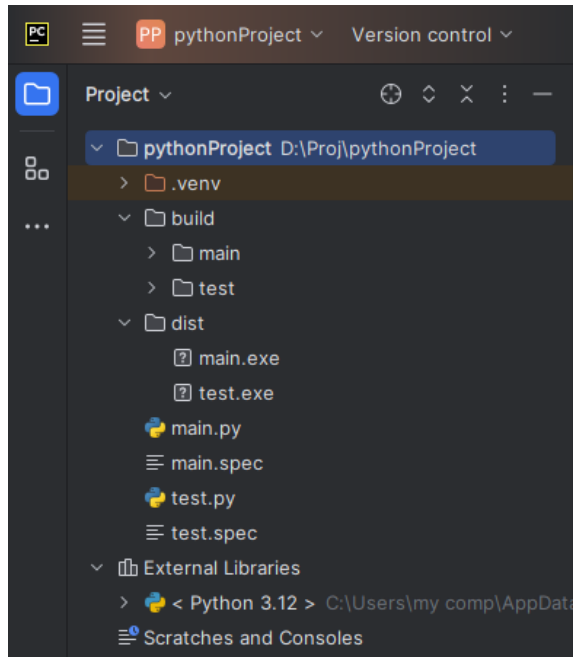


Рисунок 2.7 – Навігація по проєкту

Джерело: запропоновано автором

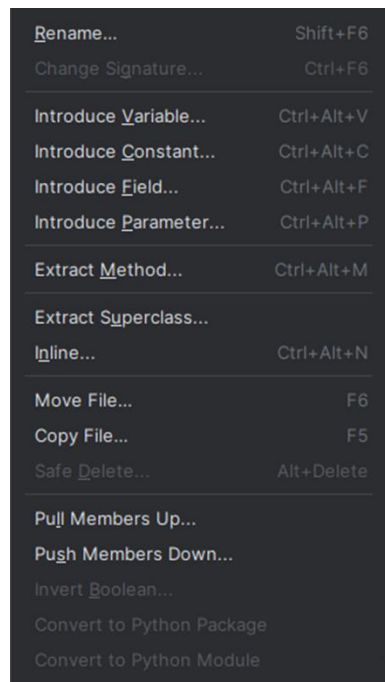


Рисунок 2.8 – Інструменти для рефакторингу

Джерело: запропоновано автором

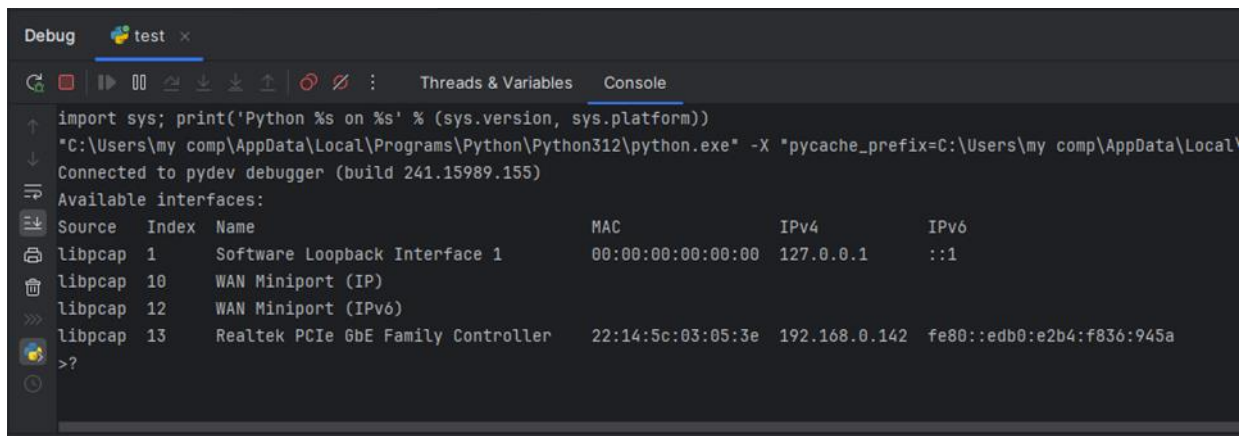


Рисунок 2.9 – Налаштування з точками зупинки, покроковим виконанням та переглядом значень змінних
Джерело: запропоновано автором

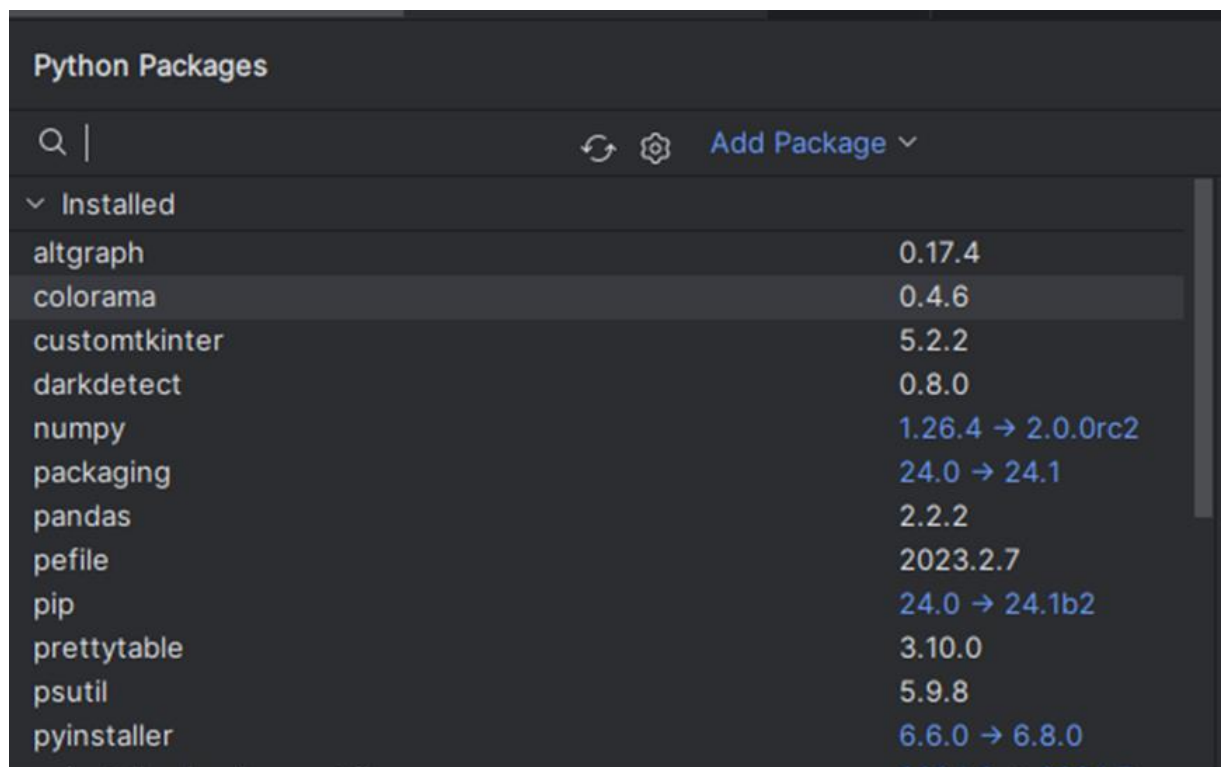


Рисунок 2.10 – Бібліотеки PyCharm
Джерело: запропоновано автором



Рисунок 2.11 – Консоль Python

Джерело: запропоновано автором

РОЗДІЛ 3

ПРОЕКТУВАННЯ ТА РЕАЛІЗАЦІЯ ВЕБІНФОРМАЦІЙНОЇ ТЕХНОЛОГІЇ ДЛЯ МОНІТОРИНГУ ТРАФІКУ ІНТЕРНЕТ-ПРОВАЙДЕРА

3.1 Проектування

Ця вебінформаційна технологія призначена для моніторингу мережевого трафіку, асоціювання трафіку з процесами, що працюють у системі, та надання актуальних даних через графічний інтерфейс користувача (GUI). Користувач може обрати мережевий інтерфейс, запускати моніторинг, переглядати зведення пакетів та статистику трафіку у режимі реального часу, переглядати історію трафіку за певний період.

У цій програмі використані наступні бібліотеки Python:

- 1) flet використовується для: створення графічного інтерфейсу користувача (GUI), текстових елементів (ft.Text), кнопок (ft.ElevatedButton, ft.IconButton), таблиць (ft.DataTable), графіків (ft.LineChart), контейнерів (ft.Container), управління вкладками тощо.
- 2) Psutil використовується для: збору інформації про системні ресурси і мережеву активність, отримання трафіку мережі (psutil.net_io_counters), активних процесів (psutil.Process), використання CPU і RAM, мережевих з'єднань (psutil.net_connections).
- 3) Time використовується для управління паузами в циклах оновлення (time.sleep(1));
- 4) sqlite3 використовується для збереження та отримання даних про трафік у базі даних, створення бази (CREATE TABLE), вставки даних (INSERT) та оновлення (UPDATE).
- 5) collections (конкретно deque) використовується зберігання історії трафіку у вигляді кільцевого буфера (download_history, upload_history) з обмеженою довжиною;

- 6) Threading використовується для запуску циклу оновлення даних у окремому потоці (threading.Thread) без блокування GUI;
- 7) Datetime використовується при роботі з датою та часом (datetime.now(), date.isoformat()), збереженні статистики та відображенні дати на графіках і в таблицях.

На рисунках 3.1 – 3.3 наведений високорівневий псевдокод, що підсумовує проектування.

Проектування програми мережевого моніторингу охоплює аспекти обробки мережевого трафіку, асоціації даних з процесами, створення інтуїтивно зрозумілого GUI та забезпечення ефективної роботи через багатопоточність. Правильний підхід до кожного з цих аспектів дозволяє створити надійну та функціональну програму, що задовольняє вимоги користувачів. Цей дизайн охоплює основні аспекти програми, забезпечуючи баланс між функціональністю, продуктивністю та зручністю.

```

Функція format_bytes(size):
    Перетворити байти у КБ/МБ/ГБ/ТБ
    Повернути форматований рядок

Функція init_db():
    Підключитися до SQLite "traffic.db"
    Створити таблиці daily_traffic та hourly_stats, якщо не існують
    Закрити базу

Функція add_traffic_to_db(upload, download, packets):
    Отримати поточну дату і годину
    Підключитися до SQLite
    Оновити або вставити денну статистику
    Оновити або вставити погодинну статистику
    Закрити базу

Функція main():
    Створити GUI сторінку (Flet)
    Встановити параметри вікна
    Викликати init_db()

--- Головний екран ---
Створити текстові елементи для ↑ та ↓ трафіку
Створити таблицю активних процесів
Створити live графік (останні 60 секунд)
Додати кнопку переходу до історії

```

Рисунок 3.1 – Перша частина високорівневого псевдокоду вебзастосунку

Джерело: запропоновано автором

```

--- Екран історії ---
Створити таблицю днів з трафіком
Створити графік погодинної історії
Додати кнопку назад на головний екран

Функція switch_tab(tab_name):
    Показати/сховати відповідний екран
    Якщо історія → завантажити дані з бази і оновити графік

Функція update_history_chart(date):
    Отримати погодинні дані з бази для date
    Підготувати точки для графіка
    Оновити графік та його шкали

Функція load_history_data():
    Отримати дані з daily_traffic
    Оновити таблицю днів
    Викликати update_history_chart для останнього дня

Додати обидва екрани у стек GUI

```

Рисунок 3.2 – Друга частина високорівневого псевдокоду

Джерело: запропоновано автором

```

--- Потік оновлення даних ---
Функція update_data_loop():
    Зчитати початкові лічильники мережі
    Створити словник процесів

    while True:
        Затримка 1 секунда
        Зчитати нові лічильники мережі
        Обчислити ↑/↓ трафік
        Оновити текстові елементи
        Додати дані у базу через add_traffic_to_db
        Додати дані у deque для live графіка
        Оновити live графік

        Отримати список активних процесів через psutil
        Зібрати дані: PID, назва, RAM, CPU, пакети, хости, статус
        Оновити таблицю процесів

        Оновити GUI

Запустити update_data_loop у окремому потоці

Запустити Flet GUI

Якщо __name__ == "__main__":
    Викликати main()

```

Рисунок 3.3 – Третя частина високорівневого псевдокоду

Джерело: запропоновано автором

3.2 Опис функціоналу вебінформаційної технології

Програма здійснює моніторинг мережевого трафіку на обраному інтерфейсі та відображає його у прив'язці до процесів, що виконуються на комп'ютері. У процесі розробки було використано так функції:

- 1) Функція `format_bytes(size)` перетворює розмір у байтах у більш читабельний формат. Вона отримує число байтів, ділить на 1024, поки не буде знайдено відповідну одиницю та повертає відформатоване значення. Ця функція використовується у таблиці процесів, вікні з інформацією про трафік та у статистичних таблицях.
- 2) Функція `init_db()` створює локальну SQLite-базу та записує в неї загальну статистику по днях та по кожній годині. Це необхідно для перегляду історії навіть після перезапуску програми.
- 3) Функція `add_traffic_to_db(upload, download, packets)` зберігає дані про трафік у базу даних.
- 4) Функція `load_history_data()` завантажує всі дні, коли був записаний трафік, і відображає їх у таблиці історії.
- 5) Функція `update_history_chart(selected_date)` оновлює графік погодинного трафіку. Вона зчитує дані з таблиці та малює графік у Flet LineChart. Викликається при відкритті вкладки «Історія» та при виборі іншої дати.
- 6) Функція `switch_tab(tab_name)` перемикає між головним екраном та історією.
- 7) Функція `update_data_loop()` виконує ключову роль. Вона нескінченно оновлює графік, таблицю процесів, статистику у базі даних, інтерфейс користувача, графік трафіку та з'єднання. Ця функція працює в окремому потоці, що гарантує стабільність інтерфейсу.
- 8) Функція `main()` створює застосунок, інтерфейс, графіки та запускає оновлення. Саме ця функція створює основні UI-елементи, під'єднує логіку `switch_tab()`, створює окремий потік для `update_data_loop()`, запускає Flet GUI, викликає `init_db()` та налаштовує вікно Flet.

3.3 Опис програмного забезпечення

Перед початком роботи з системою слід переконатися, що на комп'ютері присутні мережеві інтерфейси, вони підключені до мережі та правильно налаштовані. Крім того, необхідно перевірити наявність клієнтів у мережі. Для запуску додатку існують обов'язкові умови:

- мережевий інтерфейс, що підтримує режим прослуховування мережі;
- версія Nrcar не нижче 1.79.

Початок роботи з вебзастосунком:

1) Запуск програми

Після запуску програма автоматично почне моніторити трафік. На головному екрані користувач буде бачити: поточну швидкість завантаження та вивантаження, живий графік трафіку за останні 60 секунд, таблицю активних мережевих процесів.

На цьому екрані(див. Рисунок 3.3) користувач може:

- переглядати швидкість інтернету у двох великих цифрових панелях;
- спостерігати live-графік, який показує трафік за останню хвилину, що постійно оновлюється;
- переглядати активні мережеві процеси у таблиці, яка дозволяє бачити хто саме використовує інтернет, чи є підозрілі процеси, які програми створюють найбільше навантаження.

2) Перехід до історії використання трафіку

У верхній частині є кнопка "Історія". Після натискання користувач переходить на інший екран (див. Рисунок 3.4). На екрані історії користувач бачить велику таблицю, в якій кожен рядок — це окремий день, у який була активність. На цьому екрані можна побачити сумарний трафік за день, кількість використаного трафіка за певний період, загальну кількість мережевих пакетів.

3) Перегляд детальної погодинної статистики

При виборі будь-якого дня користувач бачить графік за 24 години цього дня, де є: швидкість завантаження за годину, швидкість вивантаження, кількість

пакетів. Це дає змогу відстежити пікові години навантаження, знаходити аномальну активність у мережі. Графік будується автоматично після вибору дати.

4) Робота у фоновому режимі (автоматично)

Користувач цього не бачить, але програма кожну секунду:

- записує трафік до локальної бази;
- оновлює графіки;
- збирає інформацію про процеси;
- логічно зберігає історію;
- визначає за якою годинаю і днем йде запис.

Програма працює без втручання користувача.

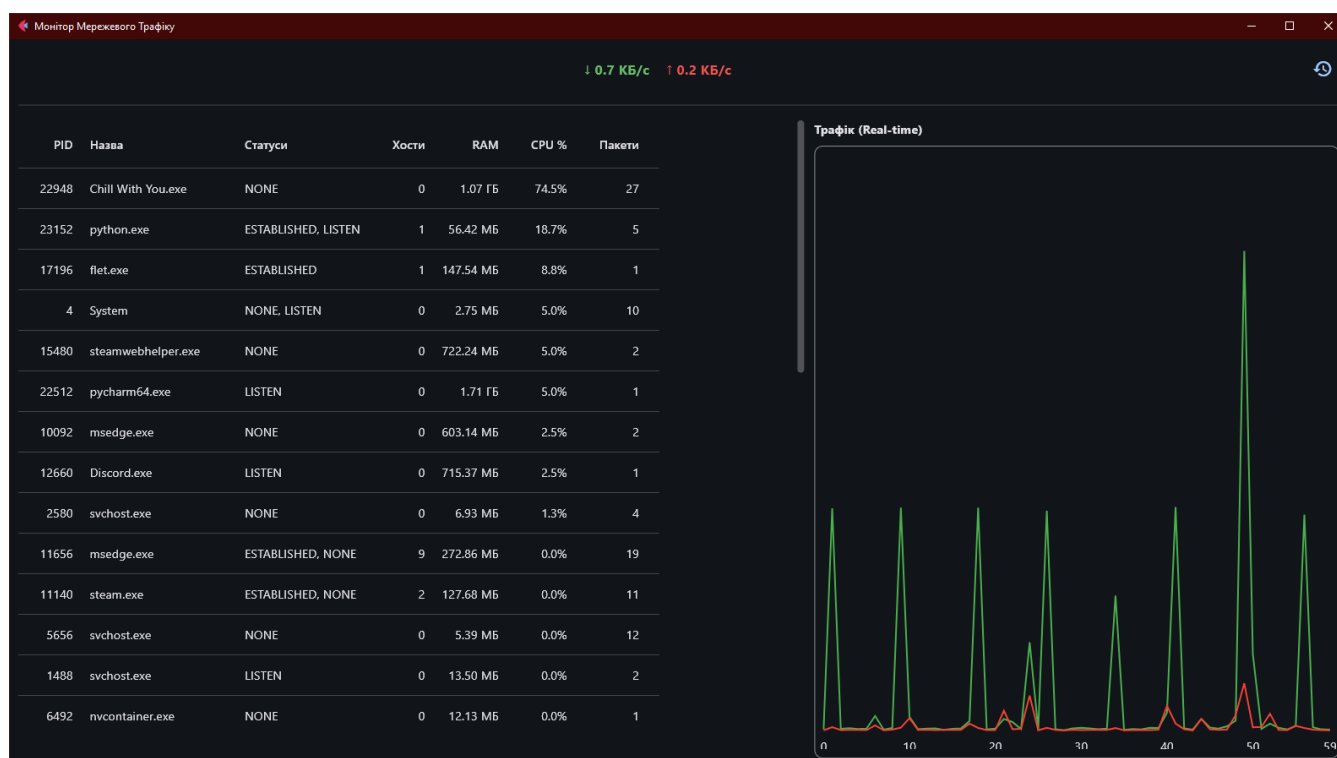


Рисунок 3.3 – Головний екран

Джерело: запропоновано автором

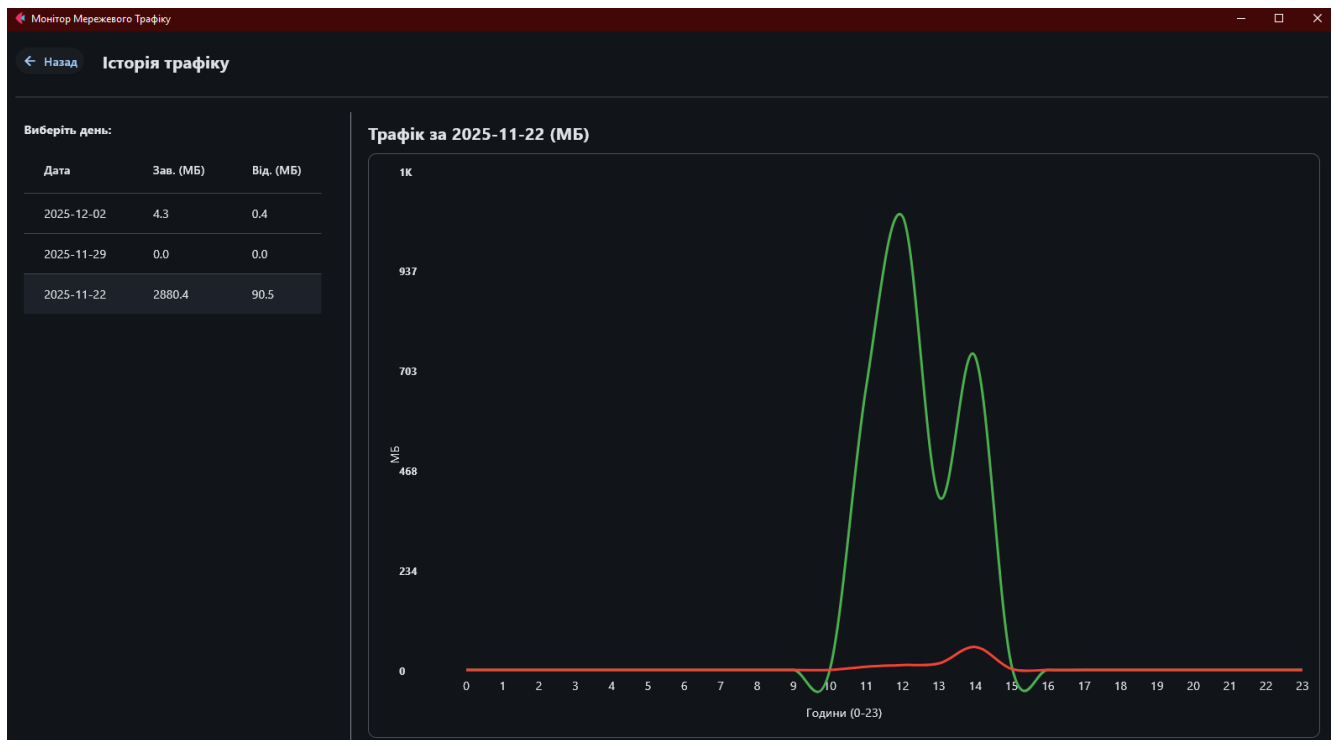


Рисунок 3.4 – Історія трафіку
Джерело: запропоновано автором

ВИСНОВКИ

У процесі виконання цієї роботи було розглянуто ключові методи програмного перехоплення мережових пакетів та аналізу трафіку. Розроблена вебінформаційна технологія забезпечує моніторинг трафіку на вибраному мережевому інтерфейсі та дозволяє пов'язувати отримані дані з відповідними процесами на комп'ютері.

В роботі були вирішені такі задачі як:

- 1) Повний моніторинг мережі в реальному часі
- 2) Аналіз активних процесів і їхньої мережевої поведінки
- 3) Графік live-трафіку з оновленням 1 раз/сек
- 4) Локальну базу даних зі збереженням історії
- 5) Погодинні та денні графіки
- 6) Зручний інтерфейс на Flet
- 7) Незалежний фоновий потік збору даних

Подальші дослідження полягають у вдосконаленні та розширенні функціональних можливостей. Можливі напрямки розвитку:

- розширення можливостей захоплення трафіку;
- інтеграція інструментів для аналізу інтернет трафіку.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Кодекс академічної доброчесності Сумського НАУ. Рішення Вченої ради Сумського НАУ № 7 від 28.10.2024. URL: <https://drive.google.com/file/d/142uwVZKHnDYNQJ6TKFxaaPcr7rT5qTN8/view>;
2. Методичні рекомендації для закладів вищої освіти з підтримки принципів академічної доброчесності. URL: https://snau.edu.ua/wp-content/uploads/2020/02/Methods_2019-Final.pdf;
3. Положення про запобігання та виявлення академічного плагіату в Сумському НАУ № 6 від 26.12.2017р.) URL: https://drive.google.com/file/d/1broXTIWw07C5S1_tCyCJUyUPXvIUgkq2/view;
4. Положення про організацію навчального процесу в Сумському національному аграрному університеті. №112 від 08.04. 2015 р. URL: <https://snau.edu.ua/wp-content/uploads/2020/02/>;
5. Положення про порядок перевірки академічних та наукових текстів на унікальність в Сумському національному аграрному університеті. Рішення Вченої ради №22 від 30.06.2025 р. URL: <https://drive.google.com/file/d/1TBRpjmNNvmalhbl8n1TTc7-yU8NpZNdS/view>;
6. Положення про кваліфікаційні роботи в Сумському національному аграрному університеті. Рішення Вченої ради № 19 від 02.05.2025 р. URL: https://drive.google.com/file/d/1airE3eGrg6hMrvfnfsrk4fAwa_xGj91f/view;
7. Правила оформлення студентських робіт: методичні вказівки для викладачів та студентів денної і заочної форм навчання. Суми: Сумський національний аграрний університет, 2025. 47 с.;
8. Feldmann A. Internet traffic analysis at scale. Proceedings of the VLDB Endowment. 2021. Vol. 14, no. 13. P. 3415. URL: <https://doi.org/10.14778/3484224.3484237>;
9. Bhandari, Aishwarya, et al. "Packet Sniffing and Network Traffic Analysis Using TCP – A New Approach." Advances in Electronics, Communication and Computing; Springer, Singapore, 2020;
10. Explainable Internet Traffic Classification / C. Callegari et al. Applied Sciences. 2021. Vol. 11, no. 10. P. 4697. URL: <https://doi.org/10.3390/app111046974>;
11. Cardwell K. Capturing Network Traffic. Tactical Wireshark. Berkeley, CA, 2023. P. 27-51. URL: https://doi.org/10.1007/978-1-4842-9291-4_2;
12. Cardwell K. Customization of the Wireshark Interface. Tactical Wireshark. Berkeley, CA, 2023. P. 1-26. URL: https://doi.org/10.1007/978-1-4842-9291-4_1;
13. Guznova E. S., Kazantsev M. A. Zabbix monitoring system for analyzing and predicting events. Achievements of Modern Radioelectronics. 2023. URL: <https://doi.org/10.18127/j20700784-202312-17>;

14. B R., K V. Deploying Nagios for Service Monitoring and Reporting. International Journal of Innovative Research in Engineering. 2023. P. 39-41. URL: <https://doi.org/10.59256/ijire.20230405005>;
15. Held G. Working with NT's Performance Monitor. Local Area Network. 2020. P. 599-607. URL: <https://doi.org/10.1201/9781003069393-c51>;
16. A Systematic Approach of Analysing Network Traffic Using Packet Sniffing with Scapy Framework / S. H. Brahmanand et al. Computer Networks and Inventive Communication Technologies. Singapore, 2021. P. 811-820. URL: https://doi.org/10.1007/978-981-16-3728-5_60;
17. Unpingco J. Pandas. Python Programming for Data Analysis. Cham, 2021. P. 127-156. URL: https://doi.org/10.1007/978-3-030-68952-0_5;
18. Nguyen Q. Hands-On Application Development with Pycharm: Accelerate Your Python Applications Using Practical Coding Techniques in Pycharm. Packt Publishing, Limited, 2019;
19. Wireshark · Go Deep. Wireshark. URL: <https://www.wireshark.org/>;
20. Zabbix :: The Enterprise-Class Open Source Network Monitoring Solution. URL: <https://www.zabbix.com>;
21. Nagios Open Source | Nagios Open Source. Nagios Open Source. URL: <https://www.nagios.org>;
22. Cisco Security Products and Solutions. Cisco. URL: <https://www.cisco.com/site/us/en/products/security/index.html>;
23. Network Performance Monitor - Onsite & Remote Monitoring | SolarWinds. Observability and IT Management Platform | SolarWinds. URL: <https://www.solarwinds.com/network-performance-monitor>;
24. Cacti® - The Complete RRDTool-based Graphing Solution. Cacti® - The Complete RRDTool-based Graphing Solution. URL: <https://www.cacti.net>;
25. WinPcap: NPF driver internals manual. WinPcap - Home. URL: https://www.winpcap.org/docs/docs_412/html/group__NPF.html.

ДОДАТКИ

ДОДАТОК А

ТЕХНІЧНЕ ЗАВДАННЯ

Розробка вебзастосунку для моніторингу трафіку інтернет-провайдера

Погоджено:

Кандидат ф.-м.наук, доцент кафедри
кібернетики та інформатики

Сергій БРАТУШКА _____

Студент групи ІСТ 2204 с.т.

Василь КОРЕЦЬКИЙ _____

СУМИ 2025

1. Призначення й мета створення вебзастосунку

1.1 Призначення вебзастосунку

Вебінформаційна технологія призначена для моніторингу та підвищення ефективності використання трафіку.

1.2 Мета створення вебзастосунку

Розробка вебінформаційної технології моніторингу трафіку для моніторингу та оптимізації використання трафіку.

1.3 Цільова аудиторія

Цільова аудиторія вебзастосунку це системні адміністратори, звичайний користувач, невеликі офіси.

2. Вимоги до вебзастосунку

2.1 Вимоги до вебзастосунку в цілому

2.1.1 Вимоги до структури й функціонування вебзастосунку

Вебінформаційна технологія має бути реалізована у вигляді застосунку. Застосунок повинен складатися із розділів із чітко розділеними функціями.

2.1.2 Вимоги до персоналу

Від персоналу не має вимагатися особливих технічних навичок для підтримки й експлуатації вебінформаційної технології, окрім загальних навичок роботи з персональним комп'ютером.

2.2 Структура вебзастосунку

2.2.1 Навігація

На головному екрані користувач буде бачити: поточну швидкість завантаження та вивантаження, живий графік трафіку за останні 60 секунд, таблицю активних мережевих процесів.

2.3 Вимоги до функціонування системи

2.3.1 Потреби користувача

Можливість перегляду історії трафіку за певний період, його графічне відображення.

2.3.2 Функціональні вимоги

На головній сторінці будуть представлені такі елементи:

- поточна швидкість завантаження;
- поточна швидкість вивантаження;
- живий графік трафіку за останні 60 секунд;
- таблиця активних мережевих процесів;
- активні мережеві процеси;

2.3.3 Системні вимоги

Системні вимоги відповідно до потреб користувачів:

- перехід до історії використання трафіку;
- запис історії в базу даних.

2.4 Вимоги до видів забезпечення

2.4.1 Вимоги до інформаційного забезпечення

Реалізація вебзастосунку відбувається з використанням:

- flet
- Psutil
- Time
- sqlite3
- collections (конкретно deque)
- Threading
- Datetime

2.4.2 Вимоги до лінгвістичного забезпечення

Вебінформаційна технологія має бути виконаний українською мовою.

2.4.3 Вимоги до програмного забезпечення

Програмне забезпечення клієнтської частини повинне задовольняти наступним вимогам:

- прсар 1.79 та вище;
- мережевий інтерфейс, що підтримує режим прослуховування мережі.

3. Склад і зміст робіт зі створення вебзастосунку

Докладний опис етапів роботи зі створення вебінформаційної технології наведено на рисунку А.1.

Режим завдань	Назва задачі	Тривалість	Початок	Завершені	Попередники	Імена ресурсів
★	▲ Розробка вебінформаційної технології моніторингу інтернет трафіку	118 днів	Ср 08.01.25	Пт 20.06.25		
→	▲ Підготовка проєкту	11 днів	Ср 08.01.25	Ср 22.01.25		
→	Визначення стратегії проєкту	2 днів	Ср 08.01.25	Чт 09.01.25		Developer
→	Встановлення комунікаційних процесів	1 день	Пт 10.01.25	Пт 10.01.25	3	Tester
→	Аналіз конкурентів	6 днів	Пн 13.01.25	Пн 20.01.25	4	Developer
→	Визначення технічних вимог	2 днів	Вт 21.01.25	Ср 22.01.25	5	Tester
→	▲ Збір вимог	16 днів	Чт 23.01.25	Чт 13.02.25		
→	Визначення вимог	6 днів	Чт 23.01.25	Чт 30.01.25	6	Developer
→	Розробка варіантів вимог	2 днів	Пт 31.01.25	Пн 03.02.25	8	Developer
→	Відстеження змін вимог	5 днів	Вт 04.02.25	Пн 10.02.25	9	Developer
→	Перевірка вимог на реалізованість	3 днів	Вт 11.02.25	Чт 13.02.25	10	Developer
→	▲ Архітектура та дизайн	26 днів	Пт 14.02.25	Пт 21.03.25		
→	Розробка технічного проєкту	12 днів	Пт 14.02.25	Пн 03.03.25	11	Developer
→	Створення варіантів дизайну	7 днів	Вт 04.03.25	Ср 12.03.25	13	Developer
→	Розробка сценаріїв взаємодії	7 днів	Чт 13.03.25	Пт 21.03.25	14	Developer
→	▲ Розробка	35 днів	Пн 24.03.25	Пт 09.05.25		
→	Написання коду	25 днів	Пн 24.03.25	Пт 25.04.25	15	Developer
→	Оптимізація продуктивності	10 днів	Пн 28.04.25	Пт 09.05.25	17	Developer
→	▲ Тестування	17 днів	Пн 12.05.25	Вт 03.06.25		
→	Створення тест-кейсів	10 днів	Пн 12.05.25	Пт 23.05.25	18	Tester
→	Виконання функціональних тестів	2 днів	Пн 26.05.25	Вт 27.05.25	20	Tester
→	Проведення тестів з навантаженням	5 днів	Ср 28.05.25	Вт 03.06.25	21	Tester
→	▲ Підтримка	13 днів	Ср 04.06.25	Пт 20.06.25		
→	Моніторинг і аналіз функціонування системи	8 днів	Ср 04.06.25	Пт 13.06.25	22	Tester
→	Виявлення та виправлення помилок	5 днів	Пн 16.06.25	Пт 20.06.25	24	Developer;Tester

Рисунок А.1 – Етапи створення вебінформаційної технології

Джерело: запропоновано автором

4. Вимоги до складу й змісту робіт із введення вебзастосунку в експлуатацію

Для того, щоб вебзастосунок могли користуватися робітники, необхідно на робочій станції користувача встановити прсар останньої версії та мати мережевий інтерфейс, що підтримує режим прослуховування мережі.

ДОДАТОК Б

ПЛАНУВАННЯ РОБІТ

Діаграма Ганта – це графічний інструмент, що використовується для візуалізації послідовності та тривалості завдань у проекті. Вона названа на честь інженера Генріка Ганта, який вперше її застосував в сфері виробництва.

Мета діаграми Ганта – показати послідовність завдань і етапів проекту в графічній формі. Це корисно для планування проекту і для візуалізації взаємозв'язків і взаємодії між завданнями в часі.

Діаграма Ганта відображає часову шкалу горизонтально, а завдання представлені у вигляді прямокутників, розташованих на цій шкалі. Довжина прямокутників відповідає тривалості завдань, а їх положення показує їх послідовність. Також, за допомогою діаграми Ганта можна відображати залежності між завданнями, ресурси, виконавців та мілістоуні.

Основні елементи діаграми Ганта:

1) список завдань: зазвичай розташований зліва від діаграми. Кожен рядок представляє окреме завдання або етап проекту.

2) шкала часу: розташована горизонтально в верхній частині діаграми. Вона може бути поділена на дні, тижні, місяці або інші часові інтервали в залежності від тривалості проекту.

3) смуги завдань: горизонтальні смуги, що відображають тривалість кожного завдання. Початок смуги відповідає даті початку завдання, а кінець – даті завершення.

4) взаємозв'язки між завданнями: Лінії або стрілки, які показують залежності між завданнями, наприклад, які завдання повинні бути завершені перед початком інших.

Діаграма Ганта є важливим інструментом управління проектами, який надає велику користь в розумінні послідовності робіт, визначенні тривалості завдань та

контролі прогресу проекту. Основні висновки, що можна зробити з використанням діаграми Ганта, включають:

- послідовність робіт: діаграма Ганта допомагає зорієнтуватися в послідовності завдань, необхідних для виконання проекту. Вона показує,
- які завдання потрібно виконати перед іншими, що допомагає зорієнтуватися в хронологічному порядку робіт;
- тривалість завдань: діаграма Ганта надає інформацію про тривалість кожного завдання, що дозволяє оцінити загальний час виконання проекту. Завдяки цьому можна планувати ресурси, визначати критичні завдання і визначати терміни виконання проекту;
- залежності між завданнями: діаграма Ганта візуалізує залежності між різними завданнями, показуючи, як одне завдання впливає на початок або завершення інших. Це допомагає визначити критичні шляхи та визначити, які завдання можуть вплинути на тривалість проекту;
- контроль прогресу: діаграма Ганта дозволяє відстежувати прогрес виконання завдань і проекту в цілому. Вона надає можливість позначати завдання, що виконані, та спостерігати за затримками або відхиленнями від плану.

На рисунках Б.1 – Б.4 наведена діаграма Ганта та використання завдань по створенню вебзастосунку для моніторингу трафіка інтернет-провайдера. Сформований проєкт надасть значну кількість інформації про систему, оскільки він описує загальні функції системи. Користувачі, представники клієнтів, менеджери проєктів, аналітики, розробники, фахівці з контролю якості і всі, хто зацікавлений у системі.

Функції можна легко описати простою мовою. Описи функцій складаються з коротких фраз, і рідко коли функції детально розробляються. Кожна функція пов'язана з елементами даних, які надають додаткову інформацію про цю функцію. Ці елементи даних називаються атрибутами функції.

Режим задачі	Назва задачі	Тривалість	Початок	Завершені	Попередники	Імена ресурсів
	Розробка вебінформаційної технології моніторингу інтернет трафіку	118 днів	Ср 08.01.25	Пт 20.06.25		
	Підготовка проєкту	11 днів	Ср 08.01.25	Ср 22.01.25		
	Визначення стратегії проєкту	2 днів	Ср 08.01.25	Чт 09.01.25		Developer
	Встановлення комунікаційних процесів	1 день	Пт 10.01.25	Пт 10.01.25	3	Tester
	Аналіз конкурентів	6 днів	Пн 13.01.25	Пн 20.01.25	4	Developer
	Визначення технічних вимог	2 днів	Вт 21.01.25	Ср 22.01.25	5	Tester
	Збір вимог	16 днів	Чт 23.01.25	Чт 13.02.25		
	Визначення вимог	6 днів	Чт 23.01.25	Чт 30.01.25	6	Developer
	Розробка варіантів вимог	2 днів	Пт 31.01.25	Пн 03.02.25	8	Developer
	Відстеження змін вимог	5 днів	Вт 04.02.25	Пн 10.02.25	9	Developer
	Перевірка вимог на реалізованість	3 днів	Вт 11.02.25	Чт 13.02.25	10	Developer
	Архітектура та дизайн	26 днів	Пт 14.02.25	Пт 21.03.25		
	Розробка технічного проєкту	12 днів	Пт 14.02.25	Пн 03.03.25	11	Developer
	Створення варіантів дизайну	7 днів	Вт 04.03.25	Ср 12.03.25	13	Developer
	Розробка сценаріїв взаємодії	7 днів	Чт 13.03.25	Пт 21.03.25	14	Developer
	Розробка	35 днів	Пн 24.03.25	Пт 09.05.25		
	Написання коду	25 днів	Пн 24.03.25	Пт 25.04.25	15	Developer
	Оптимізація продуктивності	10 днів	Пн 28.04.25	Пт 09.05.25	17	Developer
	Тестування	17 днів	Пн 12.05.25	Вт 03.06.25		
	Створення тест-кейсів	10 днів	Пн 12.05.25	Пт 23.05.25	18	Tester
	Виконання функціональних тестів	2 днів	Пн 26.05.25	Вт 27.05.25	20	Tester
	Проведення тестів з навантаженням	5 днів	Ср 28.05.25	Вт 03.06.25	21	Tester
	Підтримка	13 днів	Ср 04.06.25	Пт 20.06.25		
	Моніторинг і аналіз функціонування системи	8 днів	Ср 04.06.25	Пт 13.06.25	22	Tester
	Виявлення та виправлення помилок	5 днів	Пн 16.06.25	Пт 20.06.25	24	Developer;Tester

Рисунок Б.1 – Інформаційна частина діаграми Ганта

Джерело: запропоновано автором

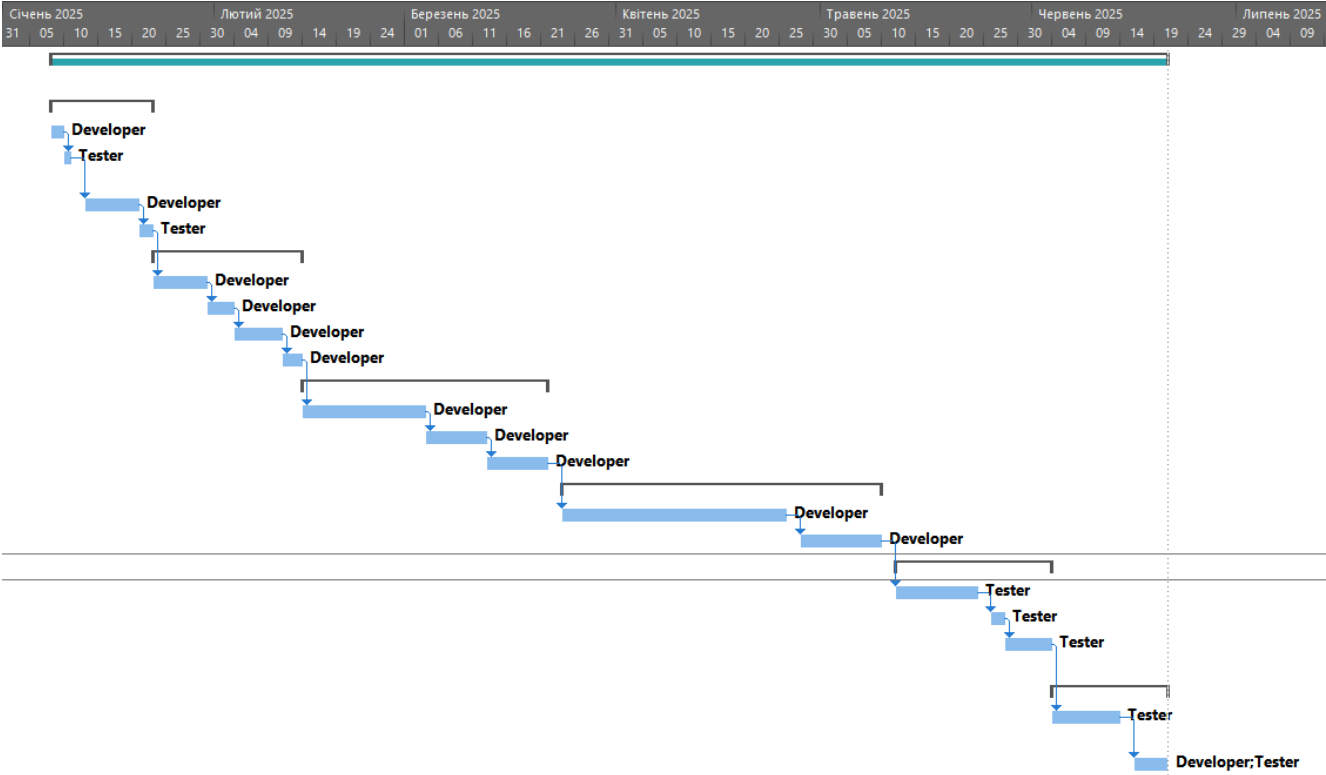


Рисунок Б.2 – Діаграма Ганта
Джерело: запропоновано автором

	Режим завдань	Ім'я завдання	Робота	Тривалість	Початок	Завершенн
		Розробка вебінформаційної технології моніторингу інтернет трафіку	2 784 г	118 днів	Ср 08.01.25	Пт 20.06.25
		Developer	944 г		Ср 08.01.25	Пт 20.06.25
		Підготовка проєкту	88 г	11 днів	Ср 08.01.25	Ср 22.01.25
		Визначення стратегії проєкту	16 г	2 днів	Ср 08.01.25	Чт 09.01.25
		Developer	16 г		Ср 08.01.25	Чт 09.01.25
		Встановлення комунікаційних процесів	8 г	1 день	Пт 10.01.25	Пт 10.01.25
		Tester	8 г		Пт 10.01.25	Пт 10.01.25
		Аналіз конкурентів	48 г	6 днів	Пн 13.01.25	Пн 20.01.25
		Developer	48 г		Пн 13.01.25	Пн 20.01.25
		Визначення технічних вимог	16 г	2 днів	Вт 21.01.25	Ср 22.01.25
		Tester	16 г		Вт 21.01.25	Ср 22.01.25
		Збір вимог	256 г	16 днів	Чт 23.01.25	Чт 13.02.25
		Developer	128 г		Чт 23.01.25	Чт 13.02.25
		Визначення вимог	48 г	6 днів	Чт 23.01.25	Чт 30.01.25
		Developer	48 г		Чт 23.01.25	Чт 30.01.25
		Розробка варіантів вимог	16 г	2 днів	Пт 31.01.25	Пн 03.02.25
		Developer	16 г		Пт 31.01.25	Пн 03.02.25
		Відстеження змін вимог	40 г	5 днів	Вт 04.02.25	Пн 10.02.25
		Developer	40 г		Вт 04.02.25	Пн 10.02.25
		Перевірка вимог на реалізованість	24 г	3 днів	Вт 11.02.25	Чт 13.02.25
		Developer	24 г		Вт 11.02.25	Чт 13.02.25
		Архітектура та дизайн	416 г	26 днів	Пт 14.02.25	Пт 21.03.25
		Developer	208 г		Пт 14.02.25	Пт 21.03.25
		Розробка технічного проєкту	96 г	12 днів	Пт 14.02.25	Пн 03.03.25
		Developer	96 г		Пт 14.02.25	Пн 03.03.25
		Створення варіантів дизайну	56 г	7 днів	Вт 04.03.25	Ср 12.03.25
		Developer	56 г		Вт 04.03.25	Ср 12.03.25
		Розробка сценаріїв взаємодії	56 г	7 днів	Чт 13.03.25	Пт 21.03.25
		Developer	56 г		Чт 13.03.25	Пт 21.03.25

Рисунок Б.3 – Використання завдань

Джерело: запропоновано автором

		Розробка	560 г	35 днів	Пн 24.03.25	Пт 09.05.25
		Developer	280 г		Пн 24.03.25	Пт 09.05.25
		Написання коду	200 г	25 днів	Пн 24.03.25	Пт 25.04.25
		Developer	200 г		Пн 24.03.25	Пт 25.04.25
		Оптимізація продуктивності	80 г	10 днів	Пн 28.04.25	Пт 09.05.25
		Developer	80 г		Пн 28.04.25	Пт 09.05.25
		Тестування	272 г	17 днів	Пн 12.05.25	Вт 03.06.25
		Developer	136 г		Пн 12.05.25	Вт 03.06.25
		Створення тест-кейсів	80 г	10 днів	Пн 12.05.25	Пт 23.05.25
		Tester	80 г		Пн 12.05.25	Пт 23.05.25
		Виконання функціональних тестів	16 г	2 днів	Пн 26.05.25	Вт 27.05.25
		Tester	16 г		Пн 26.05.25	Вт 27.05.25
		Проведення тестів з навантаженням	40 г	5 днів	Ср 28.05.25	Вт 03.06.25
		Tester	40 г		Ср 28.05.25	Вт 03.06.25
		Підтримка	248 г	13 днів	Ср 04.06.25	Пт 20.06.25
		Developer	104 г		Ср 04.06.25	Пт 20.06.25
		Моніторинг і аналіз функціонування системи	64 г	8 днів	Ср 04.06.25	Пт 13.06.25
		Tester	64 г		Ср 04.06.25	Пт 13.06.25
		Виявлення та виправлення помилок	80 г	5 днів	Пн 16.06.25	Пт 20.06.25
		Developer	40 г		Пн 16.06.25	Пт 20.06.25
		Tester	40 г		Пн 16.06.25	Пт 20.06.25

Рисунок Б.4 – Використання завдань

Джерело: запропоновано автором

Атрибути використовуються для зв'язування функцій з іншими функціями та інформацією в проекті. В переліку атрибутів немає жорстких обмежень, тому розробник формує їх на свій розсуд.

Управління ризиками проекту – це процес виявлення, аналізу та реагування на будь-який ризик, що виникає протягом життєвого циклу проекту. Крім того, управління ризиками є частиною процесу планування, що необхідно для своєчасного з'ясування можливих ризиків, і контролю цих ризиків, якщо вони дійсно виникають. Ризик – це все, що потенційно може вплинути на терміни, продуктивність або бюджет проекту. Процес ідентифікації ризиків передбачає встановлення та документування характеристик ризиків, які можуть вплинути на проект. Цей процес є ітеративним, адже з розвитком проекту можуть з'являтися нові ризики, що потребуватимуть ідентифікації та реагування.

Перелічимо можливі ризики проекту, визначимо відносну критичність, а

також можливі способи їх уникнення. Введемо умовну шкалу рівнів критичності ризиків (див. табл. Б.1):

Таблиця Б.1 – Рівні критичності ризиків

1	Незначний
2	Низький
3	Середній
4	Високий
5	Критичний

Джерело: запропоновано автором

Ризики проекту, визначені в результаті ідентифікації, а також їх потенційний вплив та ймовірність виникнення наведені у таблиці Б.2.

Таблиця Б.2 – Ризики проекту

№	Опис	Ймовірність	Вплив
1	Втрата актуальності проекту	1	4
2	Зміна вимог на пізніх етапах розробки	2	4
3	Незаплановані роботи, що потребують обов'язкового виконання	2	3
4	Вимушене зміщення термінів виконання завдань або паралельне виконання цих завдань	3	3
5	Нездійснені або важкі задачі	3	5
6	Пошкодження або відсутність обладнання	1	5
7	Непрацездатність розробника	1	5
8	Низьке залучення замовника	2	4

Джерело: запропоновано автором

Таблиця Б.3 – Матриця «Ймовірність - Втрати»

Ймовірність\Втрати	1	2	3	4	5
1				1	6,7
2			3	2,8	
3			4		5
4					

Джерело: запропоновано автором

ДОДАТОК В

КОПІЇ ПУБЛІКАЦІЙ

Proceedings of the XVIII International scientific and practical conference «Information technologies and automation– 2025»

**Міністерство освіти і науки України
Одеський національний технологічний університет
Інститут комп'ютерної інженерії, автоматизації,
робототехніки та програмування ім.П.Н.Платонова**

«ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ І АВТОМАТИЗАЦІЯ – 2025»

***МАТЕРІАЛИ
XVIII МІЖНАРОДНОЇ НАУКОВО-ПРАКТИЧНОЇ КОНФЕРЕНЦІЇ***



30-31 ЖОВТНЯ 2025 р.

м.Одеса

Вінницький національний технічний університет (Україна)	
АВТОМАТИЗАЦІЯ ВИЯВЛЕННЯ ХИБНО ЗАХИЩЕНИХ ОБЧИСЛЕНЬ. Колчин О.В., Руденко Д.Б. Інститут кібернетики ім. В.М. Глушкова НАН України	120
ПРИНЦИП РОБОТИ ЗАСОБІВ АНАЛІЗУ ІНТЕРНЕТ-ТРАФІКУ. Корецький В.А. Сумський національний аграрний університет (Україна)	122
МЕТОДИ ОПТИМІЗАЦІЇ CASE-СИСТЕМ ПРИ ПРОЄКТУВАННІ ІНФОРМАЦІЙНО-АНАЛІТИЧНИХ ПЛАТФОРМ. Корнієнко О. О. Інститут програмних систем Національної академії наук України (Україна)	124
МАТЕМАТИЧНЕ МОДЕЛЮВАННЯ ПРОЦЕСІВ ДЕФЕКТОУТВОРЕННЯ У ГРАФЕНОВИХ КВАНТОВИХ ТОЧКАХ ІЗ ВИКОРИСТАННЯМ МЕТОДІВ ШТУЧНОГО ІНТЕЛЕКТУ. Коротиш В.О. Криворізький державний педагогічний університет (Україна)	126
ОБГРУНТУВАННЯ ШЛЯХІВ ВИРІШЕННЯ ПРОБЛЕМИ ОБОРОННОГО ПЛАНУВАННЯ. Лактіонов О.І. Національний університет «Полтавська політехніка імені Юрія Кондратюка» (Україна)	129
МАТЕМАТИЧНЕ МОДЕЛЮВАННЯ ГЛОБАЛЬНИХ ТА ЛОКАЛЬНИХ ОПТИМУМІВ У ЗАДАЧАХ ОПТИМІЗАЦІЇ. Литвиненко О. Є., Кашкевич С.О. Державний університет «Київський авіаційний інститут» (Україна)	131
ГІБРИДНИЙ ПІДХІД ДО ПРОГНОЗУВАННЯ ВАРТОСТІ АКЦІЙ НА ОСНОВІ LSTM-МЕРЕЖ ТА СЕНТИМЕНТ-АНАЛІЗУ НОВИН. Маковій А.С., Лютенко І.В. Національний технічний університет «Харківський політехнічний інститут» (Україна)	133
ВИКОРИСТАННЯ НЕЧІТКОЇ КЛАСТЕРИЗАЦІЇ ДЛЯ ІДЕНТИФІКАЦІЇ ГРУП СТУДЕНТІВ У ОСВІТНІХ ДАНИХ. Матвійчук О., Мамчич Т. І. Волинський національний університет імені Лесі Українки (Україна)	134
ЗАДАЧА ЗАСТОСУВАННЯ НИЗКИ МЕТОДІВ РОЗРАХУНКУ РІВНЯ ОСВІТЛЕННЯ ТА КІЛЬКОСТІ, ПОТУЖНОСТІ ТА РОЗТАШУВАННЯ ДЖЕРЕЛ СВІТЛА. Мельников О. Ю., Малюкін О. В. Донбаська державна машинобудівна академія (Україна)	136
APPLICATION OF SIR MODELS TO PREDICT THE SPREAD OF CONTENT ON SOCIAL NETWORKS USING MACHINE LEARNING TECHNOLOGIES. Mysiuk I.V. Ivan Franko National University of Lviv (Ukraine)	139
МОДЕЛЮВАННЯ ПРОЦЕСІВ ЗАПОВНЕННЯ РІДИНОЮ РЕЗЕРВУАРІВ З ВИКОРИСТАННЯМ НЕЛІНІЙНИХ ПРИСТРОЇВ УПРАВЛІННЯ. Михайлов А. Г. Національний аерокосмічний університет «Харківський авіаційний інститут» (Україна)	140
ПРОГРАМНИЙ КОМПЛЕКС ДЛЯ МОДЕЛЮВАННЯ ТА ВІЗУАЛІЗАЦІЇ ДИНАМІКИ FPV-ДРОНІВ ІЗ ВИКОРИСТАННЯМ MATLAB GUI ТА КВАТЕРНІОННОГО ІНТЕГРАТОРА. Нікулін С. С., Заболотний О. В. Національний аерокосмічний університет «Харківський авіаційний інститут» (Україна)	142
ОСОБЛИВОСТІ ВИКОРИСТАННЯ НЕЧІТКОЇ ЛОГІКИ ДЛЯ ОЦІНЮВАННЯ ЯКОСТІ ЦИФРОВИХ ЗОБРАЖЕНЬ. Оліярник Т. І. Національний університет «Львівська політехніка» (Україна)	145
МЕТОДИ ТА АЛГОРИТМИ РОЗПІЗНАВАННЯ ОБ'ЄКТІВ У ХІМІЧНОМУ АНАЛІЗІ ТА РОЗПІЗНАВАННІ РЕАКТИВІВ. Павленко Р.О. Сумський національний аграрний університет (Україна)	147
МЕТОД ІНТЕРПІНАЦІЇ ДЛЯ 3D РЕКОНСТРУКЦІЇ ПОЛІВ КОНЦЕНТРАЦІЙ ТЯЖКИХ МЕТАЛІВ: МОДЕЛЬ, ВАЛІДАЦІЯ ТА ПОРІВНЯННЯ З МЕТОДАМИ ШЕПАРДА. Першина Ю.І., Ковтун А.В. Національний технічний університет «Харківський політехнічний інститут», Харківський національний університет радіоелектроніки (Україна)	149
ДОСЛІДЖЕННЯ РУХУ КОНВЕЄРНОГО ПОЇЗДА. Поляков В. О. Інститут транспортних систем і технологій НАН України (Україна)	151
МЕТОДИ ПІДВИЩЕННЯ ЕНТРОПІЇ ГЕНЕРАТОРІВ ПСЕВДОВИПАДКОВИХ ЧИСЕЛ ШЛЯХОМ ІНТЕГРАЦІЇ СЕНСОРНИХ ДАНИХ. Поперешняк С.В. Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського» (Україна)	154
МАТЕМАТИЧНЕ ТА КОМП'ЮТЕРНЕ МОДЕЛЮВАННЯ НЕГАУСІВСЬКИХ	157

комбінованими підходами статичного аналізу, що поєднують досяжність вузлів CFG із відстеженням залежностей між умовами та використаннями змінних.

Висновки. Описана патологія не завжди призводить до аварійних станів, використання підходу має стосуватись в першу чергу критичних обчислень. Методи автоматизації пошуку мають класичні проблеми пов'язані з ефективністю, об'ємом, генерацією хибно-позитивних або хибно-негативних результатів. Слід зауважити, що наслідки описаної патології можуть призводити до неналежної валідації вхідних даних, неправильної індексації тощо — вразливостей, які належать до топ-списку CWE [5].

Список використаної літератури

1. J. J. Chilenski, Investigation of Three Forms of the Modified Condition Decision Coverage (MC/DC) Criterion, U.S. Federal Aviation Administration, Office of Aviation Research, Tech. Rep. DOT/FAA/AR-01/18, 2001.
2. S. Rapps and E. Weyuker, "Data flow analysis techniques for test data selection," in *Proc. Int. Conf. Software Engineering (ICSE)*, 1982, pp. 272–277.
3. Kolchin and S. Potiyenko, "Extending data flow coverage to test constraint refinements," in *Lecture Notes in Computer Science*, vol. 13274, 2022. doi: 10.1007/978-3-031-07727-2_17
4. G. Piskachev, J. Späth, I. Budde, and others, "Fluently specifying taint-flow queries with FLUENTTQL," *Empirical Software Engineering*, vol. 27, no. 104, 2022. doi: 10.1007/s10664-022-10165-y
5. Common Weakness Enumeration (CWE), "2024 CWE Top 25 Most Dangerous Software Weaknesses." [Online]. Available: https://cwe.mitre.org/top25/archive/2024/2024_kev_list.html

ПРИНЦИП РОБОТИ ЗАСОБІВ АНАЛІЗУ ІНТЕРНЕТ-ТРАФІКУ

Корецький В.А. (koretckiyvasily@gmail.com)

Сумський національний аграрний університет (Україна)

Постановка проблеми. В умовах зростаючої складності мережевих інфраструктур (гібридні хмари, шифрування трафіку, високі швидкості) та постійних кіберзагроз постійно зростає необхідність забезпечення ефективного, комплексного та деталізованого аналізу інтернет-трафіку. Отже ця тема є актуальною.

Викладення основного матеріалу. Засоби для аналізу інтернет-трафіку (або мережевого трафіку) можна розділити на дві основні категорії: аналізатори мережевих пакетів (сніфери) і засоби для аналізу потоків даних (flow-based analysis), а також окрему категорію інструментів для аналізу веб-трафіку (відвідуваності сайтів).[1]

Аналізатори мережевих пакетів (Сніфери) — це програма чи пристрій, який перехоплює та записує мережеві пакети, що проходять через мережевий інтерфейс. Принцип його роботи можна розділити на три основні етапи: захоплення пакетів, фільтрація, декодування та аналіз.

В етапі захоплення пакетів сніфер переводить мережеву карту у нерозбірливий режим. У звичайному режимі мережева карта ігнорує всі пакети, адресовані іншим пристроям. У нерозбірливому режимі вона приймає та передає на обробку операційній системі всі пакети, які фізично проходять через сегмент мережі (це особливо актуально в застарілих мережах з хабами). У сучасних комутованих мережах (з використанням світців) для захоплення трафіку, призначеного іншим пристроям, адміністратори використовують функцію дзеркалювання портів (SPAN/Port Mirroring) на комутаторі.[2]

На етапі фільтрації через великий обсяг трафіку, сніфери застосовують фільтри захоплення (Capture Filters), щоб зберегти лише потрібні пакети (наприклад, тільки HTTP-трафік або пакети з певної IP-адреси).

Під час декодування та аналізу сніфер розбирає вміст кожного пакета, застосовуючи знання про мережеві протоколи згідно з моделлю TCP/IP або OSI (канальний, мережевий, транспортний,

прикладний рівні). Він відокремлює заголовки протоколів (Ethernet, IP, TCP/UDP) від даних і відображає їх у зручному для читання форматі.[3]

Засоби для аналізу потоків даних (Flow-Based Analysis) є більш високорівневим і масштабованим порівняно з пакетним аналізом. Він базується на концепції потоків. Принцип його роботи можна розділити на три основні етапи: Визначення потоку (Flow Definition), Генерація записів (Flow Export), Аналіз на колекторі (Collector Analysis).[4]

Потік (Flow) — це односпрямована послідовність пакетів між двома кінцевими точками, які мають спільний набір ключових параметрів (так званий "ключ потоку"). Стандартний ключ NetFlow v5 включає 7 основних значень: вхідний інтерфейс (Ingress interface), IP-адреса джерела, IP-адреса призначення, протокол IP (TCP, UDP, ICMP тощо), порт джерела, порт призначення, тип обслуговування (ToS).

Генерація записів (Flow Export) ведеться за допомогою мережевого пристрою (маршрутизатор, комутатор, міжмережевий екран), який з підтримкою протоколу (наприклад, NetFlow, sFlow, IPFIX) веде облік усіх активних потоків. Коли потік завершується (за таймаутом або за пакетами TCP FIN/RST), пристрій генерує запис потоку (flow record) — метадані, що містять статистику (обсяг трафіку, кількість пакетів, час початку/завершення), але не містять фактичного вмісту даних. Цей запис експортується на колектор потоків (Flow Collector).[5]

У наступному етапі колектор отримує величезну кількість зведених записів від різних мережевих пристроїв. Колектор агрегує та аналізує ці дані для: визначення найбільш завантажених хостів і додатків ("пожирачі" пропускну здатності), побудови матриці трафіку (хто з ким спілкується), виявлення аномалій, які можуть свідчити про атаки (наприклад, SYN flood або високий відсоток коротких сесій).

Засоби для аналізу веб-трафіку (Веб-аналітика) орієнтовані на маркетинговий аналіз та вивчення поведінки користувачів на веб-сайті, а не на низькорівневий мережевий аналіз. Принцип його роботи можна розділити на три основні етапи: збір даних (Tracking Code), ідентифікація та сесія, аналіз ключових метрик.

Під час збору даних на кожному сторінку сайту додається невеликий фрагмент JavaScript-коду (наприклад, код Google Analytics). Коли користувач завантажує сторінку, цей код виконується і відправляє інформацію про відвідувача та його дії (так званий хіт або подія) на сервери аналітичної системи.[6]

На етапі ідентифікації та сесії система використовує файли cookie для ідентифікації унікальних відвідувачів та відстеження їхніх дій протягом однієї сесії (періоду активності на сайті). Аналітична система обробляє зібрані дані для розрахунку ключових показників: кількість унікальних та загальних відвідувачів, джерела трафіку, показник відмов, середній час на сторінці, сторінки за сесію, досягнення цільових дій (купівля, підписка, заповнення форми), поведінка користувачів.

Аналіз інтернет-трафіку є критично важливим для забезпечення безпеки, діагностики мережі та оптимізації бізнес-процесів. Жоден із представлених вище засобів не є універсальним. Ефективний і всебічний аналіз інтернет-трафіку вимагає комплексного індивідуального підходу, що залежить від мети аналізу трафіку.

Список використаної літератури

- [1] Jakkani A. K. Real Time Network Traffic Analysis and Anomaly Detection to Enhance Network Security and Performance: Machine Learning Approaches. June-July 2024. 2024. No. 44. P. 32–44. [Online]. Available: <https://doi.org/10.55529/jecnam.44.32.44> [date of access: 07.10.2025].
- [2] Alserhani F. Analysis of Encrypted Network Traffic for Enhancing Cyber-security in Dynamic Environments. Applied Artificial Intelligence. 2024. Vol. 38, no. 1. [Online]. Available: <https://doi.org/10.1080/08839514.2024.2381882> [date of access: 07.10.2025].
- [3] Artificial Intelligence-Based Approaches for Anomaly Detection / A. K. Cherukuri et al. Encrypted Network Traffic Analysis. Cham. 2024. P. 73–99. [Online]. Available: https://doi.org/10.1007/978-3-031-62909-9_5 [date of access: 07.10.2025].
- [4] Koumar J., Hynek K., Čejka T. Network Traffic Classification Based on Single Flow Time Series Analysis. 2023 19th International Conference on Network and Service Management (CNSM), Niagara Falls, ON, Canada, 30 October – 2 November 2023. 2023. [Online]. Available: <https://doi.org/10.23919/cnsm59352.2023.10327876> [date of access: 07.10.2025].

- [5] Coutinho A. C., Araújo L. V. d. Network Data Flow Collection Methods for Cybersecurity: A Systematic Literature Review. *Computers*. 2025. Vol. 14, no. 10. P. 407. [Online]. Available: <https://doi.org/10.3390/computers14100407> [date of access: 07.10.2025].
- [6] ML-Based Traffic Classification in an SDN-Enabled Cloud Environment / O. Belkadi et al. *Electronics*. 2023. Vol. 12, no. 2. P. 269. [Online]. Available: <https://doi.org/10.3390/electronics12020269> [date of access: 07.10.2025].

УДК 004.415.2:004.42:004.89

МЕТОДИ ОПТИМІЗАЦІЇ CASE-СИСТЕМ ПРИ ПРОЄКТУВАННІ ІНФОРМАЦІЙНО-АНАЛІТИЧНИХ ПЛАТФОРМ

Корнієнко О. О. (oleksiikornilenko@gmail.com)

Інститут програмних систем Національної академії наук України (Україна)

У роботі розглянуто проблему підвищення ефективності проєктування інформаційно-аналітичних платформ шляхом оптимізації функціонування CASE-систем (Computer-Aided Software Engineering). Сучасні CASE-засоби забезпечують підтримку життєвого циклу програмних систем, проте їх продуктивність і узгодженість моделей значною мірою залежать від рівня автоматизації процесів оновлення, інтеграції та контролю змін. Недоліком більшості комерційних CASE-систем є статичність структур, дублювання артефактів та відсутність зворотного зв'язку між бізнес-вимогами і технічними моделями.

Постановка проблеми. У сучасних умовах цифрової трансформації організацій спостерігається зростання вимог до швидкості, якості та гнучкості розробки інформаційно-аналітичних систем. Одним із ключових інструментів, що забезпечують ці властивості, є CASE-системи (Computer-Aided Software Engineering), які підтримують життєвий цикл програмного забезпечення — від формування вимог до тестування й супроводу.

Однак ефективність CASE-засобів суттєво залежить від рівня їх інтеграції з бізнес-процесами, узгодженості моделей проєктування та оптимальності обробки вимог. Традиційні CASE-системи часто мають обмеження у продуктивності, надмірну формалізацію моделей і недостатню підтримку автоматичного аналізу взаємозв'язків між артефактами. Це знижує ефективність розробки складних інформаційно-аналітичних платформ, особливо у великих корпоративних середовищах.

Тому актуальним є завдання розроблення методів оптимізації CASE-систем, спрямованих на підвищення продуктивності, гнучкості та аналітичної здатності інструментів при проєктуванні інформаційно-аналітичних платформ.

Метою дослідження є розроблення методів оптимізації CASE-систем, що забезпечують підвищення ефективності процесів проєктування інформаційно-аналітичних платформ шляхом автоматизації аналізу вимог, оптимізації моделей та зниження трудомісткості етапів розробки.

Завдання дослідження

1. Проаналізувати сучасні підходи до використання CASE-засобів у проєктуванні інформаційно-аналітичних систем та визначити їхні обмеження.
2. Визначити критерії ефективності CASE-систем — продуктивність, узгодженість моделей, час обробки вимог, глибину інтеграції з бізнес-процесами.
3. Розробити модель оптимізації CASE-середовища на основі аналізу зв'язків між артефактами розробки, сценаріями змін і залежностями між компонентами.
4. Провести експериментальну перевірку ефективності запропонованих методів на прикладі корпоративної платформи управління аналітичними даними.

Суть дослідження. CASE-системи (Computer-Aided Software Engineering) виступають ключовим інструментом у процесі створення інформаційно-аналітичних систем, забезпечуючи підтримку всіх етапів життєвого циклу — від збору вимог до супроводу.

(Україна)	
ЕВОЛЮЦІЯ КОНЦЕПЦІЙ РЕФАКТОРИНГУ ТА ЇЇ АДАПТАЦІЯ ДО ОНТОЛОГІЧНИХ СИСТЕМ. Карповський Д.О., Шинкаренко В.І. Український державний університет науки та технологій(Україна)	271
ПІДХОДИ ДО МОДЕЛЮВАННЯ ЗАГРОЗ І ФРЕЙМОВОРКИ УПРАВЛІННЯ РИЗИКОМ В ІНФОРМАЦІЙНІЙ БЕЗПЕЦІ ПІДПРИЄМСТВА. Кіщак М.М. Коробейнікова Т.І. Національний університет «Львівська політехніка» (Україна)	272
АВТОМАТИЗОВАНИЙ АНАЛІЗ БЕЗПЕКИ СМАРТКОНТРАКТІВ НА ПЛАТФОРМІ ETHEREUM. Ковальчук О.Я., Іваницький Р.І. Західноукраїнський національний університет (Україна), Тернопільський національний педагогічний університет імені В. Гнатюка (Україна)	274
АНАЛІЗ СУЧАСНИХ ЗАСОБІВ ВІЯВЛЕННЯ МЕРЕЖЕВИХ АТАК. Корецький В.А. Сумський національний аграрний університет (Україна)	277
ЗАСТОСУВАННЯ МЕТОДУ МОНТЕ-КАРЛО ДЛЯ ОЦІНКИ ДЕСТРУКТИВНИХ ФАКТОРІВ У БЕЗПЕЦІ ІНФОРМАЦІЙНИХ СИСТЕМ. Кравченко Д. В., Косенко С. І. Національний університет «Одеська політехніка» (Україна)	279
АНАЛІЗ ПРОБЛЕМАТИКИ ВІЯВЛЕННЯ ДЕСТРУКТИВНИХ ФАКТОРІВ У БЕЗПЕЦІ СУЧАСНИХ ІНФОРМАЦІЙНИХ СИСТЕМ. Кравченко Д. В., Косенко С. І. Національний університет «Одеська політехніка» (Україна)	281
ІНТЕЛЕКТУАЛЬНІ МЕТОДИ АНАЛІЗУ ТА КОНТРОЛЮ ПОТОКОВИХ ДАНИХ. Кравченко Р.В. Інститут програмних систем Національної академії наук України (Україна)	283
ПОВЕДІНКОВІ ІНДИКАТОРИ ЯК ФАКТОР РАНЬОГО ПОПЕРЕДЖЕННЯ КІБЕРЗАГРОЗ. Крупа Н. О. Інститут програмних систем Національної академії наук України (Україна)	285
ОПТИМІЗАЦІЯ АРХІТЕКТУРИ ЛОГУВАННЯ ЯК СКЛАДОВА ПІДВИЩЕННЯ КІБЕРБЕЗПЕКИ ІНФОРМАЦІЙНИХ СИСТЕМ. Кузик Я.М. Харківський національний університет радіоелектроніки (Україна)	288
СТІЙКІ ДО ЛЮДСЬКОГО ФАКТОРУ СИСТЕМИ КІБЕРЗАХИСТУ. Кузик Я.М. Харківський національний університет радіоелектроніки (Україна)	290
СИСТЕМИ ВІЯВЛЕННЯ ТА ПОПЕРЕДЖЕННЯ КІБЕРАТАК У КООРПЕРАТИВНИХ МЕРЕЖАХ. Кур'ята О. С., Кічак Б.В. Ірпінський фаховий коледж НУБіП України (Україна)	292
ФОРМУВАННЯ ДЕРЖАВНОЇ ПОЛІТИКИ У СФЕРІ ІНФОРМАЦІЙНОЇ БЕЗПЕКИ: ТЕОРЕТИКО-МЕТОДОЛОГІЧНИЙ АСПЕКТ. Кушніков В.В. Національний аерокосмічний університет «Харківський авіаційний інститут» (Україна)	293
ОБМЕЖЕННЯ КЛАСИЧНИХ РЕЛЯЦІЙНИХ БАЗ ДАНИХ ТА АЛЬТЕРНАТИВНІ ПІДХОДИ ДЛЯ МАШИННОГО НАВЧАННЯ. Кушнір В.М., Вовк Р.Б. Івано-Франківський національний технічний університет нафти і газу (Україна)	295
АЛГОРИТМИ ЗАБЕЗПЕЧЕННЯ ВІДМОВОСТІЙКОСТІ В ХМАРНИХ ІНФРАСТРУКТУРАХ. Лисков В.І. Національний технічний університет «Харківський політехнічний інститут» (Україна)	298
SELECTION OF DATA STORAGE METHODS FOR WEATHER FORECAST BIG DATA COLLECTION. Liutenko Iryna, Yaroslav Kravets. National Technical University «Kharkiv Polytechnic Institute» (Ukraine)	299
ЗАСТОСУВАННЯ GOOGLE AUTHENTICATOR У СУЧАСНИХ СИСТЕМАХ ЕЛЕКТРОННОГО ДОКУМЕНТООБІГУ. Матвеев Р.В., Терейковська Л.О., Київський національний університет будівництва і архітектури (Україна)	302
ДОСЛІДЖЕННЯ ТЕХНОЛОГІЙ ДЛЯ ЗАБЕЗПЕЧЕННЯ АНОНІМНОСТІ ТА БЕЗПЕКИ В ПРОЦЕСІ ОБМІНУ КРИПТОВАЛЮТ. Овчаренко О.В., Котлик С.В. Одеський національний технологічний університет (Україна)	303
ПЕРСОНАЛІЗАЦІЯ КОРИСТУВАЧЬКОГО ДОСВІДУ ЗАСОБАМИ РЕКОМЕНДАЦІЙНИХ АЛГОРИТМІВ. Опанасюк М. С. Національний технічний університет «Харківський політехнічний інститут» (Україна)	305
ДОСЛІДЖЕННЯ ЗАСОБІВ ПОБУДОВИ ПРОЦЕСУ ТЕСТУВАННЯ МОБІЛЬНИХ	307

процесом. Для 30% контрактів тип залишився невідомим, що вказує на обмеження запропонованого методу у визначенні структури або специфіки певних контрактів. Це може бути пов'язано зі складністю аналізу деяких фрагментів коду або відсутністю відповідних маркерів у вихідному коді.

Проведене дослідження підтверджує ефективність комплексного автоматизованого підходу до аналізу цілісності та безпеки смартконтрактів в екосистемі Ethereum. Запропонована методологія дозволяє швидко ідентифікувати статус контрактів, їх прозорість та архітектурні особливості, що значно підвищує рівень безпеки в DeFi. Унікальність розробленого рішення полягає у поєднанні масштабованої автоматизації, інтеграції з відкритими API та можливості систематичного аналізу великих масивів контрактів, що робить інструмент перспективним для застосування в аудитах, освітніх програмах та попередньому оцінюванні ризиків. Подальшими напрямками дослідження є вдосконалення алгоритмів класифікації, інтеграція методів штучного інтелекту для виявлення прихованих вразливостей та розширення функціоналу у векторі мультимедійного аналізу, що сприятиме зниженню ризиків у DeFi та підвищенню довіри користувачів до смартконтрактів.

References

- [1] Alsunaidi S. J., Aljamaan H., & Hammoudeh M. MultiTagging: A vulnerable smart contract labeling and evaluation framework. *Electronics*, 13(23), 4616, 2024, DOI: 10.3390/electronics13234616.
- [2] Guo J., Lu L., & Li J. (2024). Smart contract vulnerability detection based on multi-scale encoders. *Electronics*, 13(3), 489, 3034, DOI: 10.3390/electronics13030489.
- [3] Lashkari B., & Musilek P. (2023). Evaluation of smart contract vulnerability analysis tools: A domain-specific perspective. *Information*, 14(10), 533, 2023, DOI: 10.3390/info14100533.
- [4] Kovalchuk O., Shevchu, R., & Banak, S. Cryptocurrency crime risks modeling: Environment, e-commerce, and cybersecurity issue. *IEEE Access*, 12, 50673-50688, 2024, 10.1109/ACCESS.2024.3386428.
- [5] Taherdoost H. Smart contracts in blockchain technology: A critical review. *Information*, 14(2), 117, 2023. <https://doi.org/10.3390/info14020117>.

АНАЛІЗ СУЧАСНИХ ЗАСОБІВ ВИЯВЛЕННЯ МЕРЕЖЕВИХ АТАК

Корецький В.А. (koretskyvasily@gmail.com)

Сумський національний аграрний університет (Україна)

Постановка проблеми. У сучасному кіберпросторі кількість, складність та швидкість поширення кібератак невідомо зростає. Зловмисники дедалі частіше використовують Zero-Day уразливості, складні стійкі загрози (APT), а також шифрування та поліморфізм для приховування своїх дій. У таких умовах старі, сигнатурні системи виявлення вторгнень (IDS) просто не справляються. Саме тому ця тема є надзвичайно актуальною.

Викладення основного матеріалу. Аналізуючи сучасні методи боротьби з мережевими атаками, ми неминуче приходимо до Систем Виявлення Вторгнень (СВВ, або IDS) та Систем Запобігання Вторгнень (СЗВ, або IPS). Це ключові інструменти для моніторингу, аналізу та реагування на підозрілу активність у мережах та комп'ютерних системах. Це основні класи засобів, призначених для моніторингу, аналізу та реагування на підозрілу активність у комп'ютерних системах і мережах.

Системи СВВ/СЗВ класифікуються за двома основними ознаками: за місцем моніторингу та за методикою аналізу. За місцем моніторингу такі системи діляться на мережеві (NIDS/NIPS), хостові (HIDS/HIPS) та гібридні.

Системи мережевого (NIDS/NIPS) типу моніторять трафік, що проходить через певний сегмент мережі (аналізує копію трафіку або стоїть "у розриві"). З переваг цього типу можна відзначити захист всієї підмережі та виявлення атак до досягнення цілі. В недоліки цього типу систем входять неможливість бачити шифрований трафік, недостатня швидкість обробки для високошвидкісних мереж, не здатність бачити локальних атак на хості.[1]

Системи хостового (HIDS/HIPS) типу працюють на окремому кінцевому пристрої (хості), аналізуючи системні журнали, файли аудиту, зміни конфігурації, системні виклики та активність користувачів. З переваг цього типу можна відзначити можливість бачити локальні атаки (навіть після обходу мережевого захисту), можливість аналізувати зашифрований трафік. Найбільшими слабкими місцями цього типу є потреба встановлення на кожному хості та легкість виводу системи з ладу при компрометації хоста.[2]

Системи гібридного типу комбінують NIDS і HIDS, об'єднуючи дані з різних джерел для більш повного аналізу. Такий тип системи має високу ефективність, здатність виявляти складні та багатоступінні атаки, але має велику складність впровадження та адміністрування.

За методикою аналізу системи CBV/СЗВ можна поділити на такі типи: на основі сигнатур (Signature-based) та на основі аномалій (Anomaly-based).

Системи на основі сигнатур працюють по принципу порівняння мережевого трафіку/системних подій з базою відомих шаблонів (сигнатур) атак. Це надає високу точність виявлення відомих атак, низький відсоток хибних спрацювань (False Positive). Але така система вимагає постійного оновлення бази сигнатур та не здатна виявляти нові (zero-day) атаки та модифіковані версії відомих атак.

Системи на основі аномалій створюють "профіль" нормальної поведінки (трафіку, користувачів, системи) і виявляють будь-які суттєві відхилення від цього профілю. Такий тип системи здатний виявляти раніше невідомі (zero-day) атаки, але має можливість високого відсотка хибних спрацювань (False Positive) та вимагає періоду "навчання" для формування нормального профілю.[3]

Сучасні засоби виявлення атак активно інтегрують передові технології для подолання обмежень традиційних методів. Штучний інтелект та Машинне навчання використовуються в системах на основі аномалій для більш точного моделювання нормальної поведінки та виявлення складних, прихованих атак, які не мають чітких сигнатур (наприклад, повільні DDoS-атаки, прихований трафік). Нейромережні технології (багатошарові перцептрони) демонструють високу ефективність у визначенні певних категорій атак.[4]

Network Detection and Response (NDR) – це еволюція NIDS/NIPS, яка використовує поведінковий аналіз для виявлення аномалій у мережевому трафіку, забезпечує контекстний аналіз і автоматизоване реагування. Extended Detection and Response (XDR) об'єднує дані з мережі, кінцевих точок (EDR), електронної пошти та хмарних середовищ для комплексного аналізу інцидентів і реагування на них в усій IT-інфраструктурі.[5]

Аналіз на рівні додатків (Application-based IDS) фокусується на аналізі подій, пов'язаних із конкретними додатками (наприклад, веб-додатками), дозволяючи виявляти атаки, спрямовані на вразливість ПЗ.

Технології "Пасток" (Honeypots) – це імітація вразливих систем або сервісів для залучення зловмисників які дозволяють вивчати тактики атак і збирати розвідувальну інформацію, не наражаючи на небезпеку основні ресурси.

Сучасні засоби виявлення мережевих атак еволюціонують від простих сигнатурних систем до комплексних, інтелектуальних платформ (NDR, XDR), які використовують машинне навчання та поведінковий аналіз для виявлення невідомих (аномальних) загроз. Найефективнішим підходом вважається гібридне використання різних типів систем (NIDS/HIDS/NDR/EDR) та методів (сигнатурний + аномальний) у рамках стратегії глибокоєшелюваного захисту.

Список використаної літератури

- [1] Intelligent Techniques for Detecting Network Attacks: Review and Research Directions / M. Aljabri et al. *Sensors*. 2021. Vol. 21, no. 21. P. 7070. [Online]. Available: <https://doi.org/10.3390/s21217070> [date of access: 01.10.2025].
- [2] A Survey of Intrusion Detection Systems Leveraging Host Data / R. A. Bridges et al. *ACM Computing Surveys*. 2020. Vol. 52, no. 6. P. 1–35. [Online]. Available: <https://doi.org/10.1145/3344382> [date of access: 03.10.2025].
- [3] Verma A., Paneru E., Baaniya B. Anomaly-Based Network Intrusion Detection System. *Journal of Lumbini Engineering College*. 2022. Vol. 4, no. 1. P. 38–42. [Online]. Available: <https://doi.org/10.3126/lecj.v4i1.49364> [date of access: 02.10.2025].

- [4] Kocher G., Kumar G. Machine learning and deep learning methods for intrusion detection systems: recent developments and challenges. *Soft Computing*. 2021. Vol. 25, no. 15. P. 9731–9763. [Online]. Available: <https://doi.org/10.1007/s00500-021-05893-0> [date of access: 03.10.2025].
- [5] Rozendaal K., Dissanayake-Mohottalalage T., Mailewa A. Neural Network Assisted IDS/IPS: An Overview of Implementations, Benefits, and Drawbacks. *International Journal of Computer Applications*. 2022. Vol. 184, no. 12. P. 21–28. [Online]. Available: <https://doi.org/10.5120/ijca2022922098> [date of access: 02.10.2025].

УДК 004.65

ЗАСТОСУВАННЯ МЕТОДУ МОНТЕ-КАРЛО ДЛЯ ОЦІНКИ ДЕСТРУКТИВНИХ ФАКТОРІВ У БЕЗПЕЦІ ІНФОРМАЦІЙНИХ СИСТЕМ

Кравченко Д. В., Косенко С. І.

(kravshenko_dys@gmail.com, studylearnerstudy@gmail.com)

Національний університет «Одеська політехніка» (Україна)

В тезах розглянуто застосування методу Монте-Карло для кількісної оцінки деструктивних факторів, що впливають на безпеку інформаційних систем. Запропоновано стохастичну модель, яка враховує технічні, організаційні та поведінкові загрози з урахуванням їх взаємозалежності. Проведено моделювання з використанням 100 000 ітерацій, що дозволило отримати статистично стабільні показники ризику для різних сценаріїв. Результати експериментів продемонстрували, що найбільший вплив на рівень небезпеки мають поведінкові та організаційні чинники, які формують асиметричний розподіл ризику. Метод Монте-Карло довів ефективність у прогнозуванні критичних станів системи та визначенні порогових значень стійкості. Зроблено висновок про перспективність використання цього підходу для підтримки управлінських рішень у сфері кібербезпеки.

Безпека інформаційних систем сьогодні є однією з ключових складових цифрової інфраструктури, оскільки зростання обсягів даних, швидкості їх обробки та складності взаємодій підвищує ризики виникнення деструктивних впливів. Класичні підходи до оцінки безпеки часто базуються на детермінованих моделях, які не враховують стохастичну природу реальних загроз [1]. Невизначеність параметрів, мінливість зовнішнього середовища, випадковість появи вразливостей і поведінка зломисників роблять традиційні методи малоефективними для реалістичного прогнозування ризиків. Тому застосування методів статистичного моделювання, зокрема методу Монте-Карло, відкриває нові можливості для оцінювання деструктивних факторів у системах безпеки [2]. Цей підхід дозволяє відтворювати широкий спектр випадкових сценаріїв, оцінювати ймовірність критичних подій та розробляти стратегії реагування. Мета даної роботи полягає у розробленні концептуальної моделі оцінки деструктивних факторів на основі методу Монте-Карло та аналізі його ефективності в контексті ризик-орієнтованої безпеки інформаційних систем.

Метод Монте-Карло ґрунтується на статистичному моделюванні процесів за допомогою випадкових чисел для оцінювання складних стохастичних систем [3]. У контексті інформаційної безпеки цей підхід дозволяє оцінювати не лише ймовірність реалізації загроз, а й вплив різних комбінацій факторів на загальний рівень стійкості системи [4]. У межах дослідження розглянуто три групи деструктивних факторів: технічні, організаційні та поведінкові. До технічних належать відмови обладнання, помилки конфігурації, мережеві атаки, до організаційних — недотримання політик безпеки та людські помилки, а до поведінкових — внутрішні порушення, соціальна інженерія та зломисні дії користувачів. Для кожного фактору визначено параметри розподілу ймовірностей виникнення та інтенсивності наслідків. Вхідними змінними моделі виступають частота подій, рівень впливу, час відновлення і вартість втрат. На основі цих параметрів генеруються тисячі ітерацій, що імітують можливі стани системи протягом певного періоду часу. Результатом моделювання є розподіл можливих значень збитків або порушення доступності

ДОДАТОК Г

ЛІСТИНГ ПРОГРАМ

```

import flet as ft
import psutil
import time
import sqlite3
from collections import deque
import threading
from datetime import datetime, date
import math

def format_bytes(size):
    power = 2 ** 10
    n = 0
    power_labels = {0: '', 1: 'K', 2: 'M', 3: 'G', 4: 'T'}
    while size > power:
        size /= power
        n += 1
    return f'{size:.2f} {power_labels.get(n, "")}B'

def init_db():
    conn = sqlite3.connect("traffic.db")
    cursor = conn.cursor()

    cursor.execute("""
        CREATE TABLE IF NOT EXISTS daily_traffic (

```

```

        day TEXT PRIMARY KEY,
        upload REAL,
        download REAL,
        packets INTEGER
    )
    """

```

```

cursor.execute("""
    CREATE TABLE IF NOT EXISTS hourly_stats (
        day TEXT,
        hour INTEGER,
        upload REAL,
        download REAL,
        PRIMARY KEY (day, hour)
    )
    """)

```

```

conn.commit()
conn.close()

```

```

def add_traffic_to_db(upload_kb, download_kb, packets):
    now = datetime.now()
    today_str = now.date().isoformat()
    current_hour = now.hour

    conn = sqlite3.connect("traffic.db")
    cursor = conn.cursor()

```

```
cursor.execute("SELECT upload, download, packets FROM daily_traffic WHERE
day=?", (today_str,))
```

```
row = cursor.fetchone()
```

```
if row:
```

```
    cursor.execute("""
```

```
        UPDATE daily_traffic
```

```
        SET upload = upload + ?, download = download + ?, packets = packets + ?
```

```
        WHERE day = ?
```

```
    """, (upload_kb, download_kb, packets, today_str))
```

```
else:
```

```
    cursor.execute("""
```

```
        INSERT INTO daily_traffic(day, upload, download, packets)
```

```
        VALUES (?, ?, ?, ?)
```

```
    """, (today_str, upload_kb, download_kb, packets))
```

```
cursor.execute("SELECT upload, download FROM hourly_stats WHERE day=?
AND hour=?", (today_str, current_hour))
```

```
h_row = cursor.fetchone()
```

```
if h_row:
```

```
    cursor.execute("""
```

```
        UPDATE hourly_stats
```

```
        SET upload = upload + ?, download = download + ?
```

```
        WHERE day = ? AND hour = ?
```

```
    """, (upload_kb, download_kb, today_str, current_hour))
```

```
else:
```

```
    cursor.execute("""
```

```
        INSERT INTO hourly_stats(day, hour, upload, download)
```

VALUES (?, ?, ?, ?)

""", (today_str, current_hour, upload_kb, download_kb))

conn.commit()

conn.close()

def main(page: ft.Page):

page.title = "Монітор Мережевого Трафіку"

page.theme_mode = ft.ThemeMode.DARK

page.window_width = 1600

page.window_height = 900

page.window.width = 1600

page.window.height = 900

init_db()

CPU_COUNT = psutil.cpu_count()

txt_upload = ft.Text("↑ 0 КБ/с", size=16, weight=ft.FontWeight.BOLD,
color=ft.Colors.RED_400)

txt_download = ft.Text("↓ 0 КБ/с", size=16, weight=ft.FontWeight.BOLD,
color=ft.Colors.GREEN_400)

sort_state = {"index": 5, "ascending": False}

def get_sort_key(row: ft.DataRow, index: int):

cell_value = row.cells[index].content.value

if index == 0: return int(cell_value)

```

if index in [3, 6]: return int(cell_value)
if index == 4:
    try:
        return float(cell_value.split()[0])
    except:
        return 0
if index == 5:
    return float(cell_value.replace("%", ""))
return cell_value

columns_list = [
    ft.DataColumn(ft.Text("PID"), numeric=True),
    ft.DataColumn(ft.Text("Назва")),
    ft.DataColumn(ft.Text("Статуси")),
    ft.DataColumn(ft.Text("Хости"), numeric=True),
    ft.DataColumn(ft.Text("RAM"), numeric=True),
    ft.DataColumn(ft.Text("CPU %"), numeric=True),
    ft.DataColumn(ft.Text("Пакети"), numeric=True),
]

data_table = ft.DataTable(columns=columns_list, rows=[], column_spacing=20)

def handle_sort(e: ft.DataColumnSortEvent):
    col = e.column_index
    if sort_state["index"] != col:
        sort_state["index"] = col
        sort_state["ascending"] = False
    else:
        sort_state["ascending"] = not sort_state["ascending"]

```

```

data_table.rows.sort(
    key=lambda r: get_sort_key(r, sort_state["index"]),
    reverse=not sort_state["ascending"]
)
page.update()

for col in data_table.columns:
    col.on_sort = handle_sort

max_points = 60
download_history = deque([0] * max_points, maxlen=max_points)
upload_history = deque([0] * max_points, maxlen=max_points)

live_chart = ft.LineChart(
    data_series=[
        ft.LineChartData(data_points=[], color=ft.Colors.GREEN, stroke_width=2),
        ft.LineChartData(data_points=[], color=ft.Colors.RED, stroke_width=2),
    ],
    expand=True,
    min_y=0,
    bottom_axis=ft.ChartAxis(labels_interval=10),
    tooltip_bgcolor=ft.Colors.with_opacity(0.8, ft.Colors.BLACK),
)

live_chart_container = ft.Container(
    content=live_chart,
    border=ft.border.all(1, ft.Colors.OUTLINE),
    border_radius=10,

```

```
padding=10,
expand=True
)
```

```
history_chart_data_dl = ft.LineChartData(data_points=[], color=ft.Colors.GREEN,
stroke_width=3, curved=True)
```

```
history_chart_data_up = ft.LineChartData(data_points=[], color=ft.Colors.RED,
stroke_width=3, curved=True)
```

```
history_chart = ft.LineChart(
    data_series=[history_chart_data_dl, history_chart_data_up],
    min_y=0,
    min_x=0,
    max_x=23,
    expand=True,
    bottom_axis=ft.ChartAxis(
        labels_interval=1,
        labels_size=30,
    ),
    left_axis=ft.ChartAxis(
        labels_size=80,
        title=ft.Container(
            content=ft.Text("МБ"),
            padding=ft.padding.only(right=50)
        ),
    ),
    tooltip_bgcolor=ft.Colors.with_opacity(0.8, ft.Colors.BLACK),
)
```

```
history_chart_title = ft.Text("Виберіть дату для перегляду графіка", size=20,
weight=ft.FontWeight.BOLD)
```

```
history_table = ft.DataTable(
    columns=[
        ft.DataColumn(ft.Text("Дата")),
        ft.DataColumn(ft.Text("Зав. (МБ)")),
        ft.DataColumn(ft.Text("Від. (МБ)")),
    ],
    rows=[],
    show_checkbox_column=False,
)
```

```
def update_history_chart(date_str):
    conn = sqlite3.connect("traffic.db")
    cursor = conn.cursor()

    hours_data = {h: {'up': 0, 'dl': 0} for h in range(24)}

    cursor.execute("SELECT hour, upload, download FROM hourly_stats WHERE
day=?", (date_str,))

    rows = cursor.fetchall()
    conn.close()

    max_val = 10

    for r in rows:
        h, up, dl = r
```

```

up_mb = up / 1024
dl_mb = dl / 1024
hours_data[h] = {'up': up_mb, 'dl': dl_mb}
max_val = max(max_val, up_mb, dl_mb)

points_dl = []
points_up = []
for h in range(24):
    points_dl.append(
        ft.LineChartDataPoint(h, hours_data[h]['dl'], tooltip=f'{h}:00\n↓
{hours_data[h]['dl']:.1f} MB'))
    points_up.append(
        ft.LineChartDataPoint(h, hours_data[h]['up'], tooltip=f'{h}:00\n↑
{hours_data[h]['up']:.1f} MB'))

history_chart_data_dl.data_points = points_dl
history_chart_data_up.data_points = points_up

chart_max_y = max_val * 1.1
if chart_max_y == 0: chart_max_y = 10
history_chart.max_y = chart_max_y

num_steps = 5
step_size = chart_max_y / num_steps

history_chart.left_axis.labels_interval = step_size

custom_labels = []
for i in range(num_steps + 1):

```

```
val = i * step_size
```

```
if val >= 1000:
```

```
    label_text = f"{int(round(val / 1000))}K"
```

```
else:
```

```
    label_text = f"{int(round(val))}"
```

```
custom_labels.append(
```

```
    ft.ChartAxisLabel(
```

```
        value=val,
```

```
        label=ft.Text(label_text, size=12, weight=ft.FontWeight.BOLD)
```

```
    )
```

```
)
```

```
history_chart.left_axis.labels = custom_labels
```

```
history_chart_title.value = f"Τραфик за {date_str} (МБ)"
```

```
page.update()
```

```
def on_history_row_select(e):
```

```
    for row in history_table.rows:
```

```
        row.selected = False
```

```
    e.control.selected = True
```

```
date_str = e.control.cells[0].content.value
```

```
update_history_chart(date_str)
```

```
def load_history_data():
```

```
    conn = sqlite3.connect("traffic.db")
```

```

cursor = conn.cursor()

cursor.execute("SELECT day, upload, download FROM daily_traffic ORDER BY
day DESC")

rows = cursor.fetchall()

conn.close()


new_rows = []
latest_date = None


for i, r in enumerate(rows):
    day_str = r[0]
    if i == 0: latest_date = day_str


    new_rows.append(
        ft.DataRow(
            cells=[
                ft.DataCell(ft.Text(day_str)),
                ft.DataCell(ft.Text(f'{r[2] / 1024:.1f}')),
                ft.DataCell(ft.Text(f'{r[1] / 1024:.1f}')),
            ],
            on_select_changed=on_history_row_select,
            selected=(i == 0)
        )
    )

history_table.rows = new_rows


if latest_date:
    update_history_chart(latest_date)

```

```

else:
    history_chart_title.value = "Немає даних"

main_screen_content = ft.Column([
    ft.Row([
        ft.Container(
            content=ft.Row(
                [txt_download, txt_upload],
                spacing=20,
                alignment=ft.MainAxisAlignment.CENTER
            ),
            expand=True
        ),
        ft.IconButton(
            icon=ft.Icons.HISTORY,
            tooltip="Історія",
            on_click=lambda e: switch_tab("history")
        ),
    ], height=50),
    ft.Divider(),
    ft.Row([
        ft.Container(
            content=ft.Column([data_table], scroll=ft.ScrollMode.ADAPTIVE,
expand=True),
            expand=3,
            margin=ft.margin.only(bottom=45)
        ),
        ft.Column([
            ft.Text("Трафік (Real-time)", weight="bold"),

```

```

        live_chart_container
    ], expand=2),
    ], expand=True)
], expand=True)

history_screen_content = ft.Column([
    ft.Row([
        ft.ElevatedButton("Назад", icon=ft.Icons.ARROW_BACK, on_click=lambda e:
switch_tab("main")),
        ft.Text(" Історія трафіку", size=20, weight="bold"),
    ], height=50),
    ft.Divider(),
    ft.Row([
        ft.Container(
            content=ft.Column([
                ft.Text("Виберіть день:", weight="bold"),
                ft.Column([history_table], scroll=ft.ScrollMode.AUTO, expand=True)
            ]),
            width=400,
            border=ft.border.only(right=ft.border.BorderSide(1, ft.Colors.OUTLINE)),
            padding=10,
        ),
        ft.Container(
            content=ft.Column([
                history_chart_title,
                ft.Container(
                    content=ft.Column([
                        history_chart,
                        ft.Container(

```

```

        content=ft.Text("Годины (0-23)"),
        alignment=ft.alignment.center,
        padding=ft.padding.only(top=10)
    )
], spacing=0, expand=True),
padding=20,
border=ft.border.all(1, ft.Colors.with_opacity(0.5, ft.Colors.OUTLINE)),
border_radius=10,
expand=True
)
]),
expand=True,
padding=10
)
], expand=True)
], visible=False, expand=True)

```

```

def switch_tab(tab_name):
    if tab_name == "history":
        load_history_data()
        main_screen_content.visible = False
        history_screen_content.visible = True
    else:
        history_screen_content.visible = False
        main_screen_content.visible = True
    page.update()

```

```

page.add(ft.Stack([main_screen_content, history_screen_content], expand=True))

```

```

def update_data_loop():
    last_io = psutil.net_io_counters()
    process_objects = {}

    while True:
        time.sleep(1)

        cur = psutil.net_io_counters()
        up = cur.bytes_sent - last_io.bytes_sent
        dn = cur.bytes_recv - last_io.bytes_recv
        last_io = cur

        up_kb = up / 1024
        dn_kb = dn / 1024

        txt_upload.value = f"↑ {up_kb:.1f} КБ/с"
        txt_download.value = f"↓ {dn_kb:.1f} КБ/с"

        add_traffic_to_db(up_kb, dn_kb, int(up / 100 + dn / 100))

        download_history.append(dn_kb)
        upload_history.append(up_kb)

        live_chart.data_series[0].data_points = [
            ft.LineChartDataPoint(i, v, tooltip=f"{v:.1f} КБ") for i, v in
            enumerate(download_history)
        ]
        live_chart.data_series[1].data_points = [

```

```

        ft.LineChartDataPoint(i, v, tooltip=f'{v:.1f} KB') for i, v in
enumerate(upload_history)
    ]

```

```

current_max = max(max(upload_history), max(download_history), 10)
live_chart.max_y = current_max * 1.2

```

```

processes_data = {}

```

```

active_pids = set()

```

```

try:

```

```

    for conn in psutil.net_connections(kind="inet"):

```

```

        if conn.pid:

```

```

            active_pids.add(conn.pid)

```

```

            if conn.pid not in processes_data:

```

```

                try:

```

```

                    proc = process_objects.get(conn.pid) or psutil.Process(conn.pid)

```

```

                    process_objects[conn.pid] = proc

```

```

                raw_cpu = proc.cpu_percent(interval=None)

```

```

                normalized_cpu = raw_cpu / CPU_COUNT

```

```

                if normalized_cpu > 100: normalized_cpu = 100.0

```

```

                processes_data[conn.pid] = {

```

```

                    "name": proc.name(),

```

```

                    "statuses": {conn.status},

```

```

                    "hosts": {conn.raddr[0]} if conn.raddr else set(),

```

```

                    "ram": proc.memory_info().rss,

```

```

        "cpu": normalized_cpu,
        "packets": 1
    }
except:
    pass
else:
    processes_data[conn.pid]["packets"] += 1
    if conn.status: processes_data[conn.pid]["statuses"].add(conn.status)
    if conn.raddr: processes_data[conn.pid]["hosts"].add(conn.raddr[0])
except:
    pass

for pid in list(process_objects.keys()):
    if pid not in active_pids: del process_objects[pid]

rows = []
for pid, d in processes_data.items():
    rows.append(ft.DataRow(cells=[
        ft.DataCell(ft.Text(str(pid))),
        ft.DataCell(ft.Text(d["name"])),
        ft.DataCell(ft.Text(", ".join(list(d["statuses"][:2])))),
        ft.DataCell(ft.Text(str(len(d["hosts"])))),
        ft.DataCell(ft.Text(format_bytes(d["ram"]))),
        ft.DataCell(ft.Text(f'{d["cpu"]:.1f}%')),
        ft.DataCell(ft.Text(str(d["packets"]))),
    ]))

rows.sort(key=lambda r: get_sort_key(r, sort_state["index"]), reverse=not
sort_state["ascending"])

```

```
data_table.rows = rows
```

```
try:
```

```
    page.update()
```

```
except:
```

```
    break
```

```
threading.Thread(target=update_data_loop, daemon=True).start()
```

```
if __name__ == "__main__":
```

```
    ft.app(target=main)
```