Міністерство освіти і науки України
Державний університет «Житомирська політехніка»
Факультет інформаційно-комп'ютерних технологій
Кафедра комп'ютерної інженерії та кібербезпеки

ПОЯСНЮВАЛЬНА ЗАПИСКА

до випускної кваліфікаційної роботи бакалавра на тему: «Проєкт програмного комплексу для збору, обробки та візуалізації статистики використання файлових ресурсів»

Виконав: студент 4-го курсу групи КІ-22-1
спеціальності 123 «Комп'ютерна інженерія»
_____ Ілля УСТИМЕНКО

Керівник: старший викладач кафедри КІ та КБ
_____ Ігор ФАЛЬКОВСЬКИЙ

Рецензент: Доцент кафедри КІ та КБ, к.т.н., доцент
_____ Юрій РОССІНСЬКИЙ

Житомир – 2026

Державний університет «Житомирська політехніка»
Факультет інформаційно-комп'ютерних технологій
Кафедра комп'ютерної інженерії та кібербезпеки
Спеціальність 123 «Комп'ютерна інженерія»
Освітня програма «Комп'ютерна інженерія»

ЗАТВЕРДЖУЮ
Завідувач
кафедри комп'ютерної
інженерії та кібербезпеки
Андрій Єфіменко
2 лютого 2026 р.

ЗАВДАННЯ
на випускну кваліфікаційну роботу бакалавра

Студент: Устименко Ілля Ігорович

Тема роботи: «Проєкт програмного комплексу для збору, обробки та візуалізації статистики використання файлових ресурсів»

затверджена Наказом університету від р. 2 лютого 2026 № 57/с

Термін здачі студентом закінченої роботи 23 червня 2026 р.

Вихідні дані роботи (зазначається предмет і об'єкт дослідження): процеси адміністрування, автоматизованого моніторингу та управління розподіленими мережевими сховищами у корпоративних обчислювальних системах під керуванням операційних систем сімейства Windows Server.

Обладнання та програмне забезпечення:

Обладнання та програмне забезпечення:

1. Персональний комп'ютер із підтримкою віртуалізації під управлінням ОС Windows 10 Pro (Intel Core i7, 16 Gb RAM).
2. Клієнтська операційна система Windows 10 Pro.
3. Система управління базами даних MS SQL Server 2022 Developer, SQL Server Management Studio (SSMS) 19.2.
4. Програмне забезпечення аналізу та візуалізації даних Microsoft Power BI Desktop 2.128.751.
5. Програмне середовище Windows PowerShell 5.1 (інтерфейс .NET API, компонент System.IO.FileSystemWatcher).

Клієнтське програмне забезпечення оверлейної мережі Tailscale 1.62.0 (протокол WireGuard).

Консультанти з випускної кваліфікаційної роботи із зазначенням розділів, що їх стосуються

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв
1	Ігор Фальковський	02.02.2026	02.02.2026
2	Ігор Фальковський	02.02.2026	02.02.2026
3	Ігор Фальковський	02.02.2026	02.02.2026

Студент _____ Ілля Устименко

Керівник _____ Ігор Фальковський

Календарний план

№ з/п	Назви етапів випускної кваліфікаційної роботи	Термін виконання етапів роботи	Примітка
1	Постановка завдання	02.02.2026	виконано
2	Аналіз літературних джерел	14.03.2026	виконано
3	Аналіз об’єкта дослідження	29.03.2026	виконано
4	Постановка задачі дослідження	16.04.2026	виконано
5	Розробка структурної моделі	27.04.2026	виконано
6	Проєктування архітектури системи моніторингу та БД	18.05.2026	виконано
7	Програмна реалізація компонентів та дашборду	22.05.2026	виконано
8	Тестування системи та оцінка її стабільності	28.05.2026	виконано
9	Оформлення тексту пояснювальної записки	14.06.2026	виконано
10	Захист		

Студент _____ Ілля Устименко

Керівник _____ Ігор Фальковський

АНОТАЦІЯ

Устименко І. І. Проєкт програмного комплексу для збору, обробки та візуалізації статистики використання файлових ресурсів. – Кваліфікаційна робота бакалавра за спеціальністю 123 «Комп'ютерна інженерія» / Кафедра комп'ютерної інженерії та кібербезпеки. – Державний університет «Житомирська політехніка», Житомир, 2026. Пояснювальна записка складається зі вступу, 3 розділів, висновків, списку використаної літератури та додатків. Загальний обсяг роботи становить 71 сторінку. Текст ілюстровано 14 рисунками, містить 5 таблиць. Список використаних джерел містить 20 найменувань та розміщений на 3 сторінках.

Метою кваліфікаційної роботи є підвищення ефективності системного адміністрування та оптимізація використання дискового простору корпоративних мережевих сховищ шляхом проєктування та програмно-технічної реалізації розподіленого комплексу асинхронного збору, транспортування та інтерактивної візуалізації метрик файлової активності в режимі реального часу.

У межах кваліфікаційної роботи отримано такі основні результати:

• визначено інженерні критерії та вимоги до автоматизованого збору системної телеметрії без створення критичного I/O навантаження на накопичувачі файлових серверів;

• проаналізовано архітектуру вбудованих засобів логування Windows Server, виявлено їхні обмеження щодо роботи з великими масивами сирих даних та обґрунтовано винесення аналітичного шару за межі продуктивного вузла;

• розглянуто особливості організації оверлейних P2P-мереж із використанням клієнтського програмного забезпечення Tailscale на базі протоколу WireGuard для побудови стабільних міжсерверних каналів зв'язку;

• показано розробку трирівневої структури програмного комплексу, що включає асинхронний фоновий агент на базі PowerShell (FileSystemWatcher), централізований репозиторій логів у СУБД MS SQL Server та аналітичну модель даних;

• спроєктовано інтерактивний дашборд у середовищі Microsoft Power BI Desktop у режимі DirectQuery для live-моніторингу інтенсивності операцій, розподілу навантаження та типів даних;

• наведено результати тестування та імітаційного моделювання системи за допомогою автоматизованого скрипта-генератора випадкових подій, що підтвердили її відмовостійкість та низьке споживання апаратних ресурсів.

Ключові слова: КОМП'ЮТЕРНА ІНЖЕНЕРІЯ, СИСТЕМНЕ АДМІНІСТРУВАННЯ, ФАЙЛОВІ СЕРВЕРИ, АСИНХРОННИЙ МОНІТОРИНГ, POWERSHELL, FILESYSTEMWATCHER, MICROSOFT SQL SERVER, POWER BI, TAILSCALE, WIREGUARD.

ANNOTATION

Ustymenko I. I. Project of a software system for collecting, processing, and visualizing file resource usage statistics. – Bachelor's thesis in specialty 123 "Computer Engineering" / Department of Computer Engineering and Cybersecurity. – Zhytomyr Polytechnic State University, Zhytomyr, 2026.

The explanatory note consists of an introduction, 3 chapters, conclusions, list of references, and appendices. The total volume of the work is 71 pages. The text is illustrated with 14 figures and contains 5 tables. The bibliography includes 20 titles and is located on 3 pages.

The purpose of the thesis is to increase the efficiency of system administration and optimize storage capacity allocation within corporate network shares by designing and implementing a distributed system for asynchronous real-time collection, transport, and interactive visualization of file activity metrics.

Key results obtained within the framework of the bachelor's thesis:

• defined engineering criteria and requirements for automated system telemetry collection without causing critical I/O performance overhead on production server storage;

• analyzed the architecture of native Windows Server logging facilities, revealed their performance bottlenecks, and justified the decoupling of the analytical layer from production nodes;

• considered features of overlay P2P network configuration using Tailscale software driven by the WireGuard protocol to establish stable and reliable inter-server connections;

• demonstrated development of a 3-tier software architecture including a lightweight background collection agent via PowerShell (FileSystemWatcher), a centralized log repository in MS SQL Server, and an analytical data model;

• designed an interactive dashboard inside the Microsoft Power BI Desktop ecosystem using DirectQuery mode for live-monitoring of operational intensity, load balancing, and data distribution;

• presented results of testing and simulation modeling using an automated random event generator, which verified the system's fault tolerance and low hardware resource consumption.

Keywords: COMPUTER ENGINEERING, SYSTEM ADMINISTRATION, FILE SERVERS, ASYNCHRONOUS MONITORING, POWERSHELL, FILESYSTEMWATCHER, MICROSOFT SQL SERVER, POWER BI, TAILSCALE, WIREGUARD.

ЗМІСТ

ВСТУП 10

РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ОБҐРУНТУВАННЯ АРХІТЕКТУРИ ПРОГРАМНОГО КОМПЛЕКСУ 12

1.1. Аналіз сучасного стану та обґрунтування актуальності збору статистики використання файлових ресурсів 12

1.2. Огляд існуючих рішень для моніторингу та аудиту файлових систем 17

1.3. Статистика використання файлових ресурсів 23

1.4. Інструментальні засоби реалізації програмного продукту та вимоги до апаратних засобів 27

РОЗДІЛ 2. ПРОЄКТУВАННЯ ТА ПРОГРАМНА РЕАЛІЗАЦІЯ КОМПОНЕНТІВ СИСТЕМИ 37

2.1. Загальна структура та архітектура програмного комплексу 37

2.2. Реалізація збору та обробки статистики доступу до файлових ресурсів 39

2.3. Проєктування та розробка реляційної структури бази даних 41

РОЗДІЛ 3 ДОСЛІДЖЕННЯ ТА ОЦІНКА РЕЗУЛЬТАТІВ РОБОТИ ПРОГРАМНОГО КОМПЛЕКСУ 46

3.1 Підготовка середовища до експлуатації 46

3.2 Імітаційне моделювання навантаження та верифікація збору телеметрії 50

3.3 Аналітична обробка та візуалізація метрик безпеки в Power BI 55

ВИСНОВКИ 64

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ 64

ДОДАТКИ 69

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

БД — база даних

ПЗ — програмне забезпечення

ПК — програмний комплекс

ПКІ — підсистема комп'ютерної інженерії

СУБД — система управління базами даних

API — інтерфейс прикладного програмування (Application Programming Interface)

DAX — мова виразів аналізу даних (Data Analysis Expressions)

I/O — операції введення-виведення (Input/Output)

IP — міжмережевий протокол (Internet Protocol)

MS SQL Server — система управління реляційними базами даних від Microsoft

OS — операційна система (Operating System)

P2P — однорангова комп'ютерна мережа (Peer-to-Peer)

RAM — оперативна пам'ять (Random Access Memory)

UI — інтерфейс користувача (User Interface)

VPN — віртуальна приватна мережа (Virtual Private Network)

ВСТУП

Актуальність теми. На сучасному етапі розвитку інформаційних технологій та комп'ютерної інженерії ефективне керування корпоративною інфраструктурою збереження даних є одним із ключових завдань системного адміністрування. Стрімке зростання обсягів цифрового контенту та розгортання розподілених мережевих сховищ вимагають безперервного контролю дискового простору та аналізу інтенсивності доступу до файлових ресурсів.

Традиційні вбудовані засоби логування операційних систем часто створюють надлишкове навантаження на підсистему введення-виведення (I/O) продуктивних серверів та не надають інструментів для інтерактивного live-аналізу масивів сирих даних. Це призводить до затримок у прийнятті адмінрішень щодо оптимізації дискових масивів та виявлення аномальних процесів використання ресурсів. Відтак, розробка розподілених програмно-технічних комплексів асинхронного збору, захищеного транспортування та інтерактивної візуалізації метрик файлової активності в режимі реального часу є актуальним практичним завданням для оптимізації сучасних комп'ютерних систем.

Метою кваліфікаційної роботи є підвищення ефективності системного адміністрування та оптимізація використання дискового простору корпоративних мережевих сховищ шляхом проєктування та програмно-технічної реалізації розподіленого комплексу асинхронного збору, транспортування та інтерактивної візуалізації метрик файлової активності в режимі реального часу.

Для досягнення поставленої мети визначено та вирішено такі завдання:

Визначити інженерні критерії та вимоги до автоматизованого збору системної телеметрії без створення критичного I/O навантаження на накопичувачі файлових серверів.

Проаналізувати архітектуру вбудованих засобів логування Windows Server, виявити їхні обмеження щодо роботи з великими масивами даних та обґрунтувати винесення аналітичного шару за межі продуктивного вузла.

Розглянути особливості організації оверлейних P2P-мереж із використанням клієнтського програмного забезпечення Tailscale на базі протоколу WireGuard для побудови стабільних міжсерверних каналів зв'язку.

Розробити трирівневу структуру програмного комплексу, що включає асинхронний фоновий агент на базі PowerShell (FileSystemWatcher), централізований репозиторій логів у СУБД MS SQL Server та аналітичну модель даних.

Спроєктувати інтерактивний дашборд у середовищі Microsoft Power BI Desktop у режимі DirectQuery для live-моніторингу інтенсивності операцій, розподілу навантаження та типів даних.

Провести тестування та імітаційне моделювання системи за допомогою автоматизованого скрипта-генератора випадкових подій для перевірки її відмовостійкості, оцінки споживання апаратних ресурсів та фіксації патернів аномальної поведінки.

Об’єкт дослідження – процеси адміністрування, автоматизованого моніторингу та управління розподіленими мережевими сховищами у корпоративних обчислювальних системах під керуванням операційних систем сімейства Windows Server.

Предмет дослідження – архітектурні підходи, методи, мережеві протоколи та програмно-технічні засоби асинхронного перехоплення, криптографічного транспортування, структурування та інтерактивної візуалізації телеметрії файлової активності файлових серверів.

Методи дослідження. Для вирішення поставлених завдань використано методи системного аналізу комп'ютерних мереж, теорії реляційних баз даних, інженерії програмного забезпечення, а також статистичні методи обробки даних та імітаційне моделювання для фіксації навантаження й відхилень у системі.

Практичне значення отриманих результатів полягає у створенні розподіленого програмного комплексу, який дозволяє адміністраторам автоматизувати моніторинг інфраструктури сховищ, оперативно аналізувати структуру дискового простору за допомогою інтерактивних дашбордів реального часу та оптимізувати використання апаратних ресурсів серверів.

РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ОБҐРУНТУВАННЯ АРХІТЕКТУРИ ПРОГРАМНОГО КОМПЛЕКСУ

1.1. Аналіз сучасного стану та обґрунтування актуальності збору статистики використання файлових ресурсів

Сучасні корпоративні інформаційні системи характеризуються постійним зростанням обсягів даних, які зберігаються, обробляються та передаються в межах локальних і глобальних мереж [1]. Фундаментальною складовою будь-якої IT-інфраструктури залишаються файлові сервери, які забезпечують централізоване зберігання та спільний доступ до інформаційних ресурсів підприємства [4]. Організація ефективного управління цими ресурсами є одним із головних завдань у галузі комп'ютерної інженерії та системного адміністрування.

З інженерної точки зору, забезпечення стабільної роботи файлових серверів вимагає не лише налаштування прав доступу (наприклад, з використанням списків контролю доступу ACL у файловій системі NTFS), але й безперервного моніторингу того, як саме ці ресурси використовуються користувачами та службами. Розуміння статистики використання файлових ресурсів дозволяє системним адміністраторам вирішувати низку критичних завдань: виявляти пікові навантаження на дискову підсистему, оптимізувати розподіл дискового простору, відстежувати життєвий цикл файлів та своєчасно архівувати неактуальні дані.

Штатні засоби операційних систем сімейства Windows Server, які найчастіше використовуються для побудови корпоративних мереж, надають базові можливості для аудиту доступу до об'єктів. Процес аудиту базується на генерації подій безпеки (наприклад, подія з ідентифікатором 4663), які фіксуються в системному журналі (Security Event Log) [3]. Проте, на практиці, використання стандартних інструментів, таких як Event Viewer, для комплексного аналізу є вкрай неефективним. Під час активної роботи користувачів операційна система генерує тисячі подій щохвилини. Ці журнали містять великий обсяг надлишкової інформації, представленої у незручному для сприйняття текстовому або XML-форматі.

За таких умов системний адміністратор позбавлений можливості оперативно оцінити загальну картину стану файлової системи. Пошук конкретних операцій (наприклад, масового видалення або переміщення файлів) перетворюється на ресурсомістку задачу, яка потребує ручного парсингу логів. Крім того, штатні засоби зберігання журналів подій мають обмежений обсяг і за замовчуванням циклічно перезаписуються, що унеможливлює проведення ретроспективного аналізу використання ресурсів за тривалий проміжок часу.

Отже, виникає об'єктивна потреба у створенні спеціалізованих інструментів, які здатні автоматизувати процес збору «сирих» даних з системних журналів, нормалізувати їх (відкинути зайве та структурувати) та зберігати у надійному централізованому сховищі, такому як реляційна база даних.

Проте, накопичення структурованих даних у базі є лише проміжним етапом. Для прийняття ефективних інженерних та управлінських рішень щодо IT-інфраструктури, зібрана статистика повинна бути трансформована у зрозумілий візуальний формат. Застосування сучасних засобів бізнес-аналітики (Business Intelligence) для побудови інтерактивних графіків, діаграм та дашбордів дозволяє миттєво оцінювати динаміку файлових операцій, виявляти аномальну активність або некоректну роботу програмного забезпечення.

Враховуючи вищезазначене, розробка спеціалізованого програмного комплексу для збору, нормалізації, централізованого зберігання та візуалізації статистики використання файлових ресурсів є актуальним завданням. Створення такого інструменту дозволить значно спростити роботу системних адміністраторів, підвищити прозорість функціонування файлових серверів та оптимізувати використання апаратних ресурсів підприємства.

1.1.1. Архітектурні виклики та апаратні обмеження штатних засобів логування Windows Server

У більшості корпоративних мереж як базова платформа для розгортання файлових служб використовується операційна система Windows Server. Вона містить вбудовану підсистему аудиту, яка функціонує на рівні ядра ОС та взаємодіє із підсистемою локальної безпеки (LSASS) за допомогою викликів спеціалізованих API

[13]. Коли користувач або фоновий процес ініціює операцію з файловим об'єктом (читання, запис, видалення, зміна атрибутів), підсистема аудиту генерує відповідні бінарні структури подій, які записуються у журнал безпеки (Security Event Log) у вигляді унікальних ідентифікаторів подій, таких як Event ID 4663 (фіксація спроби доступу до об'єкта) та Event ID 4656 (запит дескриптора об'єкта).

Проте, практична експлуатація штатних інструментів перегляду журналів (зокрема утиліти Event Viewer) у високонавантажених інфраструктурах виявляє серйозні архітектурні обмеження:

Проблема надлишковості та низької інформативності даних: При одночасній роботі сотень користувачів сервер генерує тисячі записів за хвилину. Кожна подія зберігається у складному бінарному XML-форматі (.evtx), який містить значну кількість службової, нерелевантної інформації (наприклад, технічні токени сесій, внутрішні GUID підсистем), що ускладнює пряме читання.

Обмеження циклічного буфера зберігання: Файли журналів подій мають жорстко лімітований максимальний розмір. За досягнення цього ліміту система діє за принципом кільцевого буфера (FIFO), затираючи старі події новими. Це унеможливлює проведення глибокого ретроспективного аналізу завантаженості файлового сервера за тривалі періоди (квартал, рік).

Високе навантаження на підсистему введення-виведення (I/O Bottleneck): Спроби побудувати аналітичні звіти безпосередньо через штатні механізми запитів WMI (Windows Management Instrumentation) або локальні скрипти парсингу .evtx-файлів призводять до критичного споживання процесорного часу та блокування дискових операцій, що знижує швидкість відклику основного корисного навантаження сервера.

1.1.2. Необхідність нормалізації даних та побудови ETL-конвеєра

Для подолання технологічного бар'єру штатних утиліт постає інженерне завдання розробки автоматизованого конвеєра даних, що реалізує концепцію ETL (Extract, Transform, Load) [2]. Програмний комплекс повинен у фоновому асинхронному режимі виконувати екстракцію "сирих" подій з ядра операційної системи, проводити їх фільтрацію та нормалізацію (очищення від надлишкових XML-тегів,

приведення часових міток до єдиного стандарту UTC, виокремлення сутностей "користувач", "файл", "тип операції") та завантажувати структуровану інформацію у реляційну базу даних (наприклад, під управлінням СУБД MS SQL Server).

Централізоване збереження логів у реляційних таблицях з налаштованими індексами та зв'язками дозволяє виконувати складні аналітичні SQL-запити за лічені мілісекунди, організовувати довготривале архівування історичних даних та гарантувати цілісність інформації без ризику її випадкового циклічного перезапису.

1.1.3. Порівняльний аналіз архітектурних рішень для збору та аналізу інфраструктурних логів

У комп'ютерній інженерії існують різні класи програмного забезпечення, що можуть використовуватися для управління системними логами. У таблиці 1.1 наведено порівняльний аналіз існуючих підходів з точки зору системного адміністрування та використання апаратних ресурсів.

Таблиця 1.1.

Порівняльний аналіз платформ логування

Критерій порівняння	Штатний інструментарій (Event Viewer)	Універсальні платформи логування (Splunk, Elastic)	Спеціалізований комплекс (Проєктоване рішення)
Спосіб збереження даних	Локальні бінарні файли .evtx (обмежений об'єм)	Децентралізовані індексовані NoSQL кластери	Оптимізована реляційна СКБД (MS SQL Server)
Споживання апаратних ресурсів (RAM/CPU)	Мінімальне у спокої, критичне під час ручного пошуку	Високе (потребує виділених серверів для індексації)	Низьке (завдяки фоновому таргетному збору телеметрії)

Критерій порівняння	Штатний інструментарій (Event Viewer)	Універсальні платформи логування (Splunk, Elastic)	Спеціалізований комплекс (Проєктоване рішення)
Глибина ретроспективного аналізу	Низька (зазвичай кілька днів через перезапис)	Висока (обмежена лише місткістю сховища)	Максимальна (налаштовувана політика агрегації в БД)
Можливості візуалізації та звітності	Відсутні (лише лінійний текстовий список)	Складні внутрішні мови запитів (SPL/KQL)	Інтерактивні BI-дашборди (інтеграція з Power BI)

Аналіз таблиці 1.1 чітко вказує, що хоча великі аналітичні платформи (Splunk, Elastic) володіють потужним функціоналом, їхнє впровадження вимагає значних фінансових витрат на ліцензування та розгортання окремих серверних кластерів, що є надлишковим для завдань моніторингу окремої файлової інфраструктури середнього підприємства. Створення спеціалізованого, легковагого рішення є найбільш економічно та інженерно виправданим кроком.

1.1.4. Інтеграція засобів Business Intelligence (BI) як інструмент операційного управління

Накопичення мільйонів рядків структурованих записів у базі даних не вирішує проблему швидкості прийняття рішень, якщо адміністратор змушений аналізувати їх за допомогою сирих SQL-таблиць. Для якісної трансформації технічних даних в інженерні рішення необхідний рівень представлення даних (Presentation Layer), реалізований на базі сучасних систем бізнес-аналітики Business Intelligence (наприклад, Microsoft Power BI або аналогічних інструментів дашбордінгу).

Побудова інтерактивних аналітичних панелей (дашбордів) дозволяє реалізувати такі інженерні сценарії:

Аналіз теплових карт (Heatmaps) активності: Візуалізація розподілу файлових операцій за годинами та днями тижня, що дає змогу точно планувати технологічні вікна для резервного копіювання чи технічного обслуговування накопичувачів у моменти найменшого навантаження на дискову підсистему.

Ідентифікація аномальних спалахів операцій: Миттєве графічне відображення різкого зростання кількості операцій модифікації чи видалення файлів певним обліковим записом, що може свідчити про збій у роботі прикладного ПЗ (наприклад, зациклення скрипта автоматизації документообігу).

Оптимізація ємності (Capacity Planning): Наочне представлення розподілу дискового простору між різними департаментами компанії, виявлення "холодних" даних (файлів, до яких не було звернень понад 6 місяців) для їх архівації та вивільнення високорівневих масивів SSD/NVMe під актуальні бізнес-процеси.

Таким чином, розробка та впровадження спроєктованого програмного комплексу для автоматизованого фонового збору, нормалізації та аналітичної BI-візуалізації статистики використання файлових ресурсів є надзвичайно актуальною науково-практичною задачею комп'ютерної інженерії. Створення такого інструменту дозволить ліквідувати обмеження штатних засобів операційної системи, мінімізувати навантаження на технічний персонал і суттєво підвищити загальну ефективність управління корпоративною IT-інфраструктурою підприємства.

1.2. Огляд існуючих рішень для моніторингу та аудиту файлових систем

Для забезпечення ефективного управління інформаційною інфраструктурою підприємства на сучасному ринку програмного забезпечення існує низка інструментів, призначених для збору логів та моніторингу файлових серверів. Усі наявні програмні продукти можна умовно розділити на дві великі категорії: вбудовані системні утиліти операційних систем та комерційні комплекси від сторонніх розробників. Кожен із цих підходів має свої технічні переваги та суттєві обмеження, які необхідно враховувати під час проєктування та обслуговування мережі.

Базовим інструментом, який постачається разом із серверними операційними системами Windows, є оснастка перегляду подій Event Viewer. Цей системний компонент дозволяє збирати та переглядати записи про доступ до файлів та папок на рівні ядра операційної системи. Проте, з точки зору системного адміністрування, використання виключно цього інструменту є вкрай незручним для щоденного моніторингу. Журнали подій представлені у вигляді великого масиву тексту або XML-даних, не підтримують можливості побудови візуальних звітів та швидко переповнюються в умовах активного використання файлового сервера. Крім того, інший штатний засіб — File Server Resource Manager — надає можливості квотування дискового простору та блокування певних типів файлів, але не дозволяє здійснювати зручний ретроспективний аналіз дій користувачів чи генерувати аналітичні дашборди щодо загального навантаження на дискову підсистему.

Серед корпоративних рішень від сторонніх розробників широкого поширення набули комплексні системи моніторингу інфраструктури, такі як Netwrix Auditor або ManageEngine ADAudit Plus. Зазначені програмні продукти є потужними інструментами, які автоматизують збір логів, їхню обробку та надають розвинені засоби візуалізації. Вони дозволяють детально відстежувати зміни у файловій системі та формувати графіки навантаження. Однак впровадження таких систем супроводжується низкою труднощів, особливо для невеликих та середніх інфраструктур. Насамперед це пов'язано з високою вартістю комерційних ліцензій. Також подібні платформи є структурно важкими та часто надлишковими за своїм функціоналом для вирішення вузьких адміністративних задач. Вони вимагають виділення значних апаратних потужностей для розгортання окремих серверів обробки і баз даних та потребують тривалого часу на інтеграцію й подальше обслуговування.

Аналіз існуючих програмних рішень демонструє наявність суттєвого розриву між базовими вбудованими засобами операційних систем та складними, дорогими комерційними комплексами. Вбудовані утиліти не забезпечують належного рівня нормалізації та візуалізації даних, тоді як корпоративні платформи є фінансово та

ресурсно обтяжливими. Відповідно, розробка власного спеціалізованого програмного комплексу, який поєднує гнучкість скриптових мов для агрегації даних, надійність реляційних баз даних для їхнього зберігання та сучасні аналітичні платформи для візуалізації, є доцільним і технічно обґрунтованим кроком. Такий підхід дозволить отримати ефективний, цільовий інструмент адміністрування без зайвого навантаження на бюджет та апаратні ресурси підприємства.

1.2.1. Архітектурний аналіз та обмеження штатних засобів Microsoft Windows Server

Базовим технологічним шаром, який постачається інтегрованим у серверні операційні системи сімейства Windows, є оснастка Event Viewer (Переглядач подій), що функціонує в синергії з підсистемою локальної безпеки та драйверами файлових систем (наприклад, NTFS.sys) [8]. Цей компонент забезпечує перехоплення подій безпосередньо на рівні ядра ОС. При активації відповідних дескрипторів групових політик (Group Policy Objects — GPO) система починає документувати транзакції у вигляді структурованих записів журналу безпеки.

З погляду системної інженерії, використання Event Viewer як самостійної аналітичної платформи для моніторингу навантажених файлових серверів супроводжується низкою критичних недоліків:

Низька швидкість обробки пошукових запитів: Структура файлів журналів подій з розширенням .evtx є послідовним бінарним XML-деревом. За відсутності класичного механізму індексування (як у реляційних БД), будь-яка спроба виконати фільтрацію логів за певний проміжок часу змушує утиліту здійснювати повне послідовне сканування (Full Scan) файлу на диску. Це викликає різке зростання довжини черги дискової підсистеми (Physical Disk Queue Length) і призводить до тимчасового "заморожування" інтерфейсу та затримок в обслуговуванні клієнтських запитів.

Формат представлення та відсутність агрегації: "Сирі" події (наприклад, Event ID 4663) фіксують атомарні операції. Якщо користувач копіює директорію з 10 000 дрібних файлів, Event Viewer згенерує 10 000 окремих записів, що повністю заповнить буфер, але не дасть адміністратору узагальненої інформації про сутність

транзакції. Відсутність вбудованого математичного апарату для групування, обчислення середніх значень та математичного очікування утилізує людський ресурс на ручну рутинну аналітику.

Іншим штатним інструментом є File Server Resource Manager (FSRM). Це функціональна роль Windows Server, що виконує завдання квотування дискового простору, класифікації інфраструктурних даних та керування блокуванням розширень файлів (наприклад, для захисту від несанкціонованого запису виконуваних скриптів) [14]. FSRM підтримує генерацію статичних звітів за розкладом (наприклад, перелік великих файлів чи файлів, які рідко використовуються), проте він позбавлений динаміки: інструмент не працює в режимі реального часу (Real-time monitoring), не дозволяє відстежувати часові тренди завантаженості шини введення-виведення та не має інтерфейсів для передачі сирої телеметрії у зовнішні аналітичні системи.

1.2.2. Аналіз важковагових комерційних комплексів сторонніх розробників

У великих корпоративних мережах (Enterprise-сегмент) для нівелювання недоліків стандартного логування впроваджують спеціалізовані комерційні екосистеми, серед яких лідерами є Netwrix Auditor та ManageEngine ADAudit Plus. Ці платформи використовують агентську або безагентну (через віддалені виклики RPC/WMI) архітектуру для збору подій, здійснюють їх нормалізацію на власних серверах-колекторах та зберігають дані у пропрієтарних сховищах або зовнішніх СКБД промислового рівня .

Основні переваги таких систем полягають у наявності розвиненого Presentation Layer: адміністратор отримує готові інтерактивні панелі, звіти про відповідність міжнародним стандартам управління IT-інфраструктурою та модулі оповіщення. Проте, їх інтеграція у структуру середніх та малих підприємств пов'язана з декількома серйозними інженерно-економічними бар'єрами:

Критичні вимоги до апаратного забезпечення (Hardware Overhead): Для стабільної роботи систем типу Netwrix Auditor необхідне виділення значних обчислювальних ресурсів — окремих багатоядерних серверів додатків та виділених ліцен-

зованих серверів баз даних (MS SQL Server Enterprise), що виконують постійне важке індексування. В умовах обмеженого бюджету це створює невиправдане навантаження на апаратний стек компанії.

Надлишковість функціоналу (Feature Creep): Ці комплекси проєктуються як універсальні комбайни, що одночасно аналізують Active Directory, Exchange Server, SharePoint, бази даних та мережеве обладнання. Якщо перед інженером стоїть чітке завдання — оптимізація та збір статистики конкретної файлової підсистеми, 80% функцій комерційного продукту залишаться незадіяними, хоча вони потребуватимуть постійного оновлення, конфігурації та оплати ліцензій.

Висока вартість володіння (TCO): Ліцензійна політика сторонніх розробників найчастіше базується на кількості активних облікових записів (Active Users) або об'ємах оброблюваного трафіку логів за добу. Для підприємств із великим штатом співробітників вартість ліцензій стає економічно неефективною порівняно з розробкою власного цільового рішення.

1.2.3. Порівняльний техніко-економічний аналіз систем моніторингу

Для наочного обґрунтування розробки власного програмного комплексу проведено порівняльний аналіз наявних архітектурних підходів за ключовими технічними та експлуатаційними критеріями, результати якого наведено в таблиці 1.2.

Таблиця 1.2.

Порівняльний аналіз функціональних можливостей існуючих аналогів та проєктуваного комплексу

Технічний критерій порівняння	Вбудовані утиліти (Event Viewer / FSRM)	Комерційні системи (Netwrix / ManageEngine)	Проєктований програмний комплекс
Архітектура збору даних	Локальна, на основі бінарних XML-файлів ядра	Централізована, агентська/безагентна (RPC/WMI)	Легковагова таргетна фонова агрегація (скрипти/API)

Технічний критерій порівняння	Вбудовані утиліти (Event Viewer / FSRM)	Комерційні системи (Netwrix / ManageEngine)	Проєктований програмний комплекс
Індексація та оптимізація сховища	Відсутня, послідовне читання диска	Повна важка індексація NoSQL/SQL кластерів	Реляційні індекси, оптимізовані під конкретні типи запитів
Вплив на продуктивність хоста (I/O overhead)	Високий під час пошуку через відсутність індексів	Помірний/Високий через постійні опитування по WMI	Мінімальний завдяки асинхронній потоковій обробці логів
Гнучкість масштабування й налаштування	Низька (обмежена жорсткою логікою розробників ОС)	Середня (конфігурація в межах графічного інтерфейсу)	Максимальна (повний контроль над логікою парсингу та БД)
Фінансові витрати на впровадження	Безкоштовно (входить у вартість ліцензії ОС)	Високі (ліцензування за користувачів/сервери + підтримка)	Мінімальні (витрати лише на етап розробки та проєктування)

Проведений детальний аналіз ринку програмного забезпечення (див. табл. 1.2) наочно демонструє існування вираженого технологічного розриву. З одного боку, базові вбудовані утиліти операційних систем є безкоштовними й легковаговими, проте абсолютно непридатні для глибокого структурного аналізу та швидкої візуалізації інформації через застарілі механізми зберігання й обробки логів. З іншого боку, важкі комерційні комплекси вирішують завдання представлення даних, але створюють колосальне навантаження на обчислювальні вузли підприємства й вимагають значних фінансових вкладень, що часто є критичним для системної архітектури середнього рівня.

Таким чином, розробка власного спеціалізованого програмного комплексу є повністю обґрунтованим, раціональним та технічно доцільним кроком. Створення

легковагового архітектурного рішення, яке базуватиметься на гнучких скриптових мовах автоматизації для таргетного виокремлення подій, надійній і швидкій реляційній базі даних для структурованого накопичення телеметрії та сучасних інтерактивних аналітичних BI-платформах для презентації звітів, дозволить усунути виявлений технологічний дефіцит. Такий підхід забезпечить інженерний персонал підприємства високоефективним, кастомізованим інструментом оперативного адміністрування та планування ємності файлових ресурсів, повністю виключивши надлишкові апаратні й фінансові витрати.

1.3. Статистика використання файлових ресурсів

У сучасних інформаційних системах файлові ресурси є одним із головних засобів зберігання та обробки даних. У процесі роботи користувачі та програмні засоби постійно виконують операції створення, читання, модифікації, копіювання, переміщення та видалення файлів. Сукупність відомостей про такі операції формує статистику використання файлових ресурсів.

Статистика використання файлових ресурсів являє собою набір кількісних та якісних показників, що характеризують інтенсивність і особливості роботи користувачів та програм із файлами та каталогами. Аналіз цих показників дозволяє отримати об'єктивне уявлення про стан файлової інфраструктури, виявити потенційні загрози безпеці та оцінити ефективність використання інформаційних ресурсів організації.

До основних статистичних показників використання файлових ресурсів належать кількість звернень до файлів, частота виконання операцій читання та запису, кількість створених і видалених файлів, обсяг оброблених даних, а також інформація про користувачів, які здійснювали відповідні дії. Особливу цінність мають відомості про час виконання операцій, оскільки вони дозволяють аналізувати активність користувачів та визначати аномальні події.

Збір статистичних даних щодо файлових ресурсів є важливим завданням для забезпечення інформаційної безпеки. Аналіз накопичених даних дозволяє виявляти

спроби несанкціонованого доступу, масове копіювання інформації, видалення критично важливих документів та інші події, що можуть свідчити про порушення політик безпеки або внутрішні загрози.

Крім забезпечення безпеки, статистика використання файлових ресурсів використовується для оптимізації роботи інформаційних систем. На основі отриманих даних можна визначати найбільш використовувані каталоги та файли, оцінювати навантаження на файлові сервери, планувати модернізацію апаратного забезпечення та вдосконалювати політики зберігання інформації.

Для ефективного аналізу великих обсягів статистичної інформації необхідне використання спеціалізованих програмних засобів, які забезпечують автоматизований збір, зберігання та обробку даних. Саме тому актуальним є створення програмного комплексу, здатного автоматично отримувати інформацію про доступ до файлових ресурсів, накопичувати її в базі даних та надавати результати аналізу у зручній для користувача формі.

1.3.1. Класифікація та формалізація низькорівневих файлових метрик

Для побудови ефективного програмного комплексу збору статистики необхідно чітко класифікувати вхідні метрики. З інженерної точки зору їх поділяють на дві основні категорії: метрики інтенсивності (навантаження) та метрики об'єктної структури.

До метрик інтенсивності належать:

Частота дискретизації операцій (IOPS — Input/Output Operations Per Second): Кількість окремих запитів на читання (Read IOPS) або запис (Write IOPS), що надходять до файлової системи за одиницю часу [20].

Пропускна здатність (Throughput): Об'єм даних (у мегабайтах за секунду — MB/s), що фізично переміщується між оперативною пам'яттю (RAM) та дисковими масивами під час файлових операцій.

Таймінг затримок (Latency / Response Time): Час, витрачений операційною системою та контролером диска на виконання одного файлового виклику. Зростання цього показника свідчить про перевантаження дискової шини.

До метрик об'єктної структури (метаданих) належать ідентифікатори суб'єктів (Security Identifiers — SID користувачів, назви процесів), абсолютні шляхи до об'єктів у просторі імен файлової системи, типи файлів за їхніми сигнатурами або розширеннями, а також вектори операцій (створення, модифікація метаданих, видалення, переміщення). Особливу цінність для комп'ютерної інженерії мають точні часові мітки високої роздільної здатності (хронологічні вектори з точністю до мілісекунд), які дозволяють виконувати кореляційний аналіз подій.

1.3.2. Вплив файлових операцій на апаратну архітектуру та продуктивність

Збір статистики використання ресурсів є критичним для запобігання апаратним збоям. Наприклад, масові дрібні операції запису чи модифікації метаданих файлів (типові для некоректно спроєктованих прикладних програм або скриптів автоматизації) створюють специфічний вид навантаження — випадковий запис (Random Write). У сучасних твердотільних накопичувачах (SSD/NVMe) це викликає ефект під назвою Write Amplification (коефіцієнт посилення запису) [5], що призводить до прискореної деградації комірок пам'яті NAND та швидкого вичерпання ресурсу TBW (Total Bytes Written).

Постійний моніторинг статистики дозволяє вчасно виявляти такі деструктивні патерни до того, як вони спричинять фізичну відмову обладнання. Окрім того, аналіз розподілу запитів у часі дає змогу розрахувати оптимальний розмір блоку файлової системи (Cluster Size) під час форматування дискових масивів під конкретний тип профілю навантаження підприємства.

1.3.3. Структурування метрик для реляційного моделювання

Для того щоб програмний комплекс міг ефективно зберігати та обробляти мільйони атомарних подій без ризику експоненційного розростання бази даних, у процесі проєктування необхідно реалізувати чітку схему нормалізації. Сирі дані з журналів операційної системи Windows (наприклад, XML-структури події 4663) мають бути розпарсені та декомпоновані на атрибути реляційної моделі.

У таблиці 1.3 наведено логічне відображення сирих інфраструктурних даних у структуровані атрибути бази даних, що реалізується в проєктованому програмному комплексі.

Таблиця 1.3.

Порівняльний аналіз систем управління базами даних для збереження системної телеметрії

Сирий параметр (Windows Event XML)	Цільовий атрибут БД (Data Type)	Інженерне призначення метрики	Приклад нормалізованого значення
TimeCreated SystemTime	EventTimestamp (DATETIME2)	Побудова часових рядів, фіксація пікових годин	2026-06-19 14:23:11.451
SubjectUserSid / AccountName	UserID (INT - Foreign Key)	Ідентифікація ініціатора навантаження (сервіс/юзер)	i.ustymenko (ID: 42)
ObjectName (ObjectServer)	FilePath (VARCHAR(max))	Локалізація найбільш затребуваних папок/директорій	D:\Shares\Engineering\Doc.docx
Accesses (Mask)	OperationTypeID (TINYINT)	Визначення типу I/O профілю (Read-heavy / Write-heavy)	WriteData (ID: 2)
ProcessName	ProcessID (INT - Foreign Key)	Аналіз програмних джерел файлової активності	explorer.exe (ID: 104)

Завдяки структурі, наведеній у таблиці 1.3, проєктована база даних позбавляється текстової надлишковості. Використання сурогатних ключів (Foreign Keys) для користувачів та процесів дозволяє оптимізувати розмір одного запису в таблиці транзакцій до мінімальних байтових значень. Це забезпечує високу швидкість агрегації та сумісність із концепцією індексованого збереження даних.

1.3.4. Оптимізація інфраструктури на основі статистичного аналізу

У системному адмініструванні накопичена статистика виступає головним інструментом для реалізації стратегій Capacity Planning (планування потужностей) та

оптимізації дискового простору. На основі аналізу довготривалих часових рядів інженери отримують можливість автоматизувати керування життєвим циклом інформації (ILM). Наприклад, виокремлюються так звані "гарячі дані" (файли з високою частотою Read/Write IOPS протягом останніх 30 днів) та "холодні дані" (архіви та документи, до яких не було жодного звернення понад пів року).

Це дає змогу налаштувати автоматичні правила міграції: "холодні" дані безперервно й непомітно для користувачів переміщуються на повільні, енергоефективні та дешеві жорсткі диски (HDD) або в хмарні холодні сховища, тоді як дорогі та швидкісні масиви NVMe SSD звільняються під оперативні бізнес-завдання підприємства. Такий інженерний підхід дозволяє суттєво знизити капітальні витрати (CAPEX) на розширення серверного парку та оптимізувати утилізацію наявного заліза.

Таким чином, створення спеціалізованого програмного комплексу, що автоматизує збір, нормалізацію та зберігання статистики використання файлових ресурсів у реляційній базі даних, є логічним завершенням аналітичного етапу проєктування. Нове програмне рішення дозволить усунути обмеження сирих системних логів, надасть інструменти для математичного розрахунку апаратного навантаження та забезпечить надійну основу для подальшої візуалізації метрик на рівні бізнес-аналітики, що є невід'ємною частиною сучасної комп'ютерної інженерії.

1.4. Інструментальні засоби реалізації програмного продукту та вимоги до апаратних засобів

Важливим етапом під час проєктування будь-якого програмного забезпечення є комплексний аналіз та обґрунтований вибір інструментальних засобів розробки, а також визначення вимог до апаратного забезпечення. У контексті створення системи для збору та аналізу статистики використання файлових ресурсів, вибір технологічного стека має вирішальне значення для забезпечення стабільності, продуктивності та масштабованості кінцевого продукту. Оскільки розроблюваний комплекс інтегрується в існуючу корпоративну IT-інфраструктуру, програмні

інструменти повинні мати високий рівень сумісності з цільовими операційними системами та не створювати конфліктів із наявними службами.

Процес вибору інструментальних засобів базується на архітектурних особливостях розроблюваної системи, яка логічно поділена на рівень збору телеметрії, рівень централізованого зберігання та рівень аналітичної візуалізації. Для кожного з цих рівнів необхідно підібрати оптимальні технології, які найкраще відповідають поставленим інженерним завданням. Ключовими критеріями вибору є мінімізація навантаження на мережеві канали під час передачі логів, забезпечення високої швидкості запису великих масивів даних у базу та можливість гнучкої обробки інформації для побудови графічних звітів у режимі реального часу. Крім того, враховується поріг входження в технологію, наявність офіційної документації та підтримка інструментів з боку компаній-розробників.

Паралельно з вибором програмних платформ необхідно чітко визначити апаратні вимоги для розгортання комплексу. Специфіка системного адміністрування та моніторингу полягає в тому, що інструменти аудиту не повинні негативно впливати на продуктивність основних файлових серверів. Відповідно, програмний агент, який працює безпосередньо на цільовому вузлі, має бути максимально оптимізованим та споживати мінімум процесорного часу і оперативної пам'яті. Водночас серверна частина, яка відповідає за обробку бази даних та роботу аналітичної платформи, потребує виділення достатніх обчислювальних потужностей для забезпечення транзакційної цілісності та швидкої генерації дашбордів.

Враховуючи зазначені критерії та архітектурні вимоги, подальший аналіз присвячено детальному обґрунтуванню вибору конкретної мови сценаріїв для агрегації системних подій, системи управління реляційними базами даних для накопичення історичної статистики та платформи бізнес-аналітики для візуалізації результатів. Також формалізуються конкретні кількісні показники мінімально необхідного апаратного забезпечення для кожного вузла розроблюваної системи.

1.4.1. Обґрунтування вибору мови сценаріїв для збору даних

Першочерговим завданням розроблюваного програмного комплексу є отримання доступу до системних журналів операційної системи та вилучення з них цільової інформації про файлові транзакції. Для виконання цього завдання на рівні цільового сервера необхідно обрати інструмент, який забезпечує гнучкість налаштування програмної логіки, мінімально навантажує обчислювальні ресурси та не вимагає встановлення громіздких сторонніх залежностей. Оскільки об’єктом моніторингу виступають сервери під управлінням операційних систем сімейства Windows, інструмент збору має бути максимально адаптованим до їхньої внутрішньої архітектури.

Як базову технологію для розробки модуля збору та нормалізації даних було обрано об'єктно-орієнтовану мову сценаріїв PowerShell. Вибір на користь цього інструменту зумовлений його глибокою системною інтеграцією та наявністю вбудованого середовища виконання у всіх сучасних версіях Windows Server. На відміну від використання універсальних мов програмування, наприклад Python, або розробки компільованих програмних агентів на C++, застосування PowerShell не потребує інсталяції додаткових інтерпретаторів, компіляторів чи сторонніх бібліотек на кожен цільовий вузол. Такий підхід повністю відповідає інженерним принципам мінімізації втручання в роботу продуктивних серверів та значно спрощує процес розгортання системи в масштабах корпоративної мережі.

Ключовою технічною перевагою PowerShell у контексті обробки логів є використання об'єктно-орієнтованого підходу до конвеєризації даних. Традиційні оболонки командного рядка оперують виключно неструктурованими текстовими потоками, що змушує розробників застосовувати складні регулярні вирази для синтаксичного аналізу виводу. Натомість середовище PowerShell побудоване поверх платформи .NET і передає між командами конвеєра повноцінні структуровані об'єкти з набором визначених властивостей та методів [15]. Це дозволяє гарантовано і безпомилково вилучати необхідні параметри системних подій (наприклад, часову мітку, обліковий запис користувача, тип операції та повний шлях до файлу), оминаючи етап ресурсомісткого ручного парсингу тексту.

Додатковим фактором, що обґрунтовує використання цієї мови сценаріїв, є наявність оптимізованих вбудованих командлетів для прямої взаємодії з журналами подій операційної системи та інструментарієм управління Windows (WMI/CIM). Це дає змогу формувати точні запити до ядра системи і відфільтровувати надлишкові інформаційні повідомлення безпосередньо під час зчитування, значно економлячи процесорний час. Крім того, PowerShell має нативні засоби для взаємодії з реляційними базами даних, що дозволяє організувати безперебійну передачу нормалізованої телеметрії до централізованого сховища в автоматичному фоновому режимі, забезпечуючи високу надійність роботи першого архітектурного рівня комплексу.

1.4.2. Вибір системи управління базами даних

Накопичення, структурування та довгострокове зберігання великих обсягів нормалізованої телеметрії, що надходить від модуля збору даних, вимагає впровадження надійної, відмовостійкої та високопродуктивної архітектури збереження інформації. У практиці комп'ютерної інженерії та системного адміністрування традиційні підходи, засновані на збереженні логів у плоских текстових файлах, форматах XML або CSV, швидко демонструють свою неефективність у разі масштабування інфраструктури. Такі файли створюють високе навантаження на дискову підсистему при одночасному записі, не підтримують індексацію для швидкого пошуку, позбавлені механізмів контролю цілісності та є вкрай незручними для прямої інтеграції з сучасними аналітичними платформами. Відповідно, виникає потреба в організації централізованого сховища на базі повноцінної системи управління базами даних, яка здатна забезпечити високу швидкість обробки транзакцій, безпеку збереження історичних відомостей та підтримку складних вибірок.

Для реалізації централізованого сховища розроблюваного програмного комплексу було обрано систему управління реляційними базами даних Microsoft SQL Server. Цей вибір є цілком обґрунтованим з інженерної та архітектурної точок зору, оскільки зазначена платформа демонструє найвищу ефективність і стабільність під час роботи в екосистемі операційних систем сімейства Windows. Нативна синергія між Microsoft SQL Server, мовою сценаріїв PowerShell та аналітичним середовищем

Power BI дозволяє побудувати безперервний конвеєр обробки даних без необхідності розробки складних проміжних інтерфейсів, використання сторонніх драйверів чи додаткового перетворення типів даних, що значно підвищує загальну надійність і швидкодію всієї системи.

Важливою технічною характеристикою Microsoft SQL Server є його здатність ефективно функціонувати в умовах високої інтенсивності операцій введення-виведення. Оскільки сучасні корпоративні файлові сервери обслуговують сотні або тисячі користувачів одночасно, інтенсивність генерації системних подій може досягати великих масштабів. Обрана СУБД має оптимізовані механізми транзакційного запису та кешування даних у оперативній пам'яті, що дозволяє фоновому скрипту збору даних оперативно скидати масиви оброблених логів у базу, не створюючи при цьому затримок і не блокуючи роботу операційної системи [18]. Підтримка технологій індексації забезпечує високу швидкість виконання пошукових запитів навіть тоді, коли обсяг збереженої історичної телеметрії досягає мільйонів записів.

Використання реляційної моделі в межах Microsoft SQL Server дозволяє чітко спроєктувати структуру бази даних, розділивши інформацію на логічно пов'язані таблиці. Це дає змогу винести умовно постійні дані, такі як імена серверів, облікові записи користувачів та описи типів файлових операцій, в окремі довідники, а в основній таблиці транзакцій зберігати лише числові ідентифікатори (зовнішні ключі), часові мітки та унікальні атрибути подій (шляхи до файлів та назви процесів). Такий підхід забезпечує високий рівень нормалізації даних, суттєво мінімізує надлишковість інформації, оптимізує використання дискового простору серверного обладнання та гарантує суворе дотримання вимог ACID щодо цілісності й ізольованості транзакцій. У результаті створюється стабільна архітектурна основа, яка дозволяє аналітичному модулю миттєво отримувати точні зрізи статистики для моніторингу та оптимізації комп'ютерних ресурсів підприємства.

1.4.3. Засіб візуалізації та аналітики

Накопичення великих масивів телеметричних даних у реляційній базі вирішує лише частину інженерного завдання, пов'язаного зі зберіганням інформації. Для того щоб зібрана статистика мала практичну цінність під час адміністрування

IT-інфраструктури, необхідно забезпечити її трансформацію у зрозумілий візуальний формат. Аналіз сирих табличних даних у системі управління базами даних є неефективним та ресурсомістким процесом, який не дозволяє оперативно оцінювати загальний стан файлових ресурсів чи виявляти аномальні сплески активності. Тому третій, завершальний рівень архітектури програмного комплексу вимагає впровадження потужного інструменту бізнес-аналітики, здатного обробляти історичну статистику та формувати інтерактивні графічні звіти.

Як основний засіб побудови аналітичних звітів та візуалізації результатів моніторингу було обрано платформу Microsoft Power BI Desktop. Цей вибір обґрунтовується високим рівнем нативної інтеграції цього продукту з обраною раніше базою даних Microsoft SQL Server, що дозволяє створити безшовний конвеєр передачі інформації. Платформа підтримує декілька режимів роботи з даними, серед яких імпорт інформації у власну високопродуктивну in-memory модель (модель у пам'яті) та режим прямого запиту (DirectQuery) [19]. Використання цих технологій дає змогу ефективно працювати з багатомільйонними масивами системних логів, забезпечуючи швидке відображення графіків без суттєвого сповільнення роботи робочого місця адміністратора.

Важливою технічною перевагою платформи Power BI є наявність вбудованої мови виразів аналізу даних DAX (Data Analysis Expressions) [10]. Завдяки її застосуванню з'являється можливість не просто відображати збережені записи, а й виконувати складні агрегації та обчислення «на льоту». Це дозволяє генерувати специфічні інженерні метрики, наприклад, розраховувати середню кількість операцій читання або модифікації файлів за певну годину, визначати сервери з найбільшим дисковим навантаженням або формувати рейтинги найактивніших споживачів ресурсів. Мова DAX дозволяє перенести значну частину обчислювального навантаження з сервера бази даних безпосередньо на аналітичний модуль.

Крім потужного аналітичного ядра, Power BI Desktop надає розширений набір інтерактивних візуальних компонентів: гістограми, матриці, кругові діаграми та графіки часових рядів. Поєднання цих елементів дозволяє розробити єдиний комплексний дашборд (панель індикаторів), де зміна параметрів на одному графіку

автоматично фільтрує дані на всіх інших. Впровадження такого інструменту візуалізації дозволяє автоматизувати процес моніторингу, перетворюючи розрізнені системні логи на наочні індикатори стану інфраструктури, що значно підвищує оперативність прийняття рішень щодо оптимізації та управління комп'ютерними ресурсами підприємства.

1.4.4. Вимоги до апаратного та програмного забезпечення

Формулювання та обґрунтування вимог до апаратного та програмного забезпечення є критично важливим етапом інженерного проєктування, оскільки від цього залежить стабільність функціонування розроблюваного комплексу, його продуктивність та інтеграційні властивості в межах реальної виробничої IT-інфраструктури підприємства. Оскільки архітектура системи є трирівневою і розподіленою, вимоги необхідно розглядати окремо для кожного компонента: вузла збору даних (цільового файлового сервера), сервера зберігання інформації (бази даних) та автоматизованого робочого місця системного адміністратора, яке виконує роль аналітичного центру візуалізації. Такий диференційований підхід дозволяє оптимізувати витрати обчислювальних ресурсів та забезпечити максимальну ефективність системи без створення надлишкового навантаження на мережеве й серверне обладнання.

Програмне середовище для розгортання модуля збору телеметрії безпосередньо визначається вибором базових технологій автоматизації та адміністрування. Цільовий файловий сервер повинен функціонувати під управлінням серверних операційних систем сімейства Microsoft Windows Server версії 2016, 2019 або 2022, що забезпечує повну сумісність із механізмами розширеного аудиту доступу до об'єктів [11]. Обов'язковою вимогою є наявність встановленої підсистеми Windows PowerShell версії 5.1 або вище, а також активованих політик локальної або доменної безпеки, які відповідають за фіксацію подій доступу до файлової системи NTFS. З точки зору апаратного забезпечення, програмний агент розроблений таким чином, щоб мінімізувати вплив на основні операції введення-виведення файлового

сервера. Для його стабільного функціонування у фоновому режимі достатньо виділення мінімальних ресурсів, які включають один гігагерц частоти одного процесорного ядра та до двохсот мегабайт вільної оперативної пам'яті, що є практично непомітним для сучасних серверних платформ.

Серверна частина, яка відповідає за рівень централізованого зберігання, висуває значно вищі вимоги до програмного та апаратного оточення, оскільки цей вузол обробляє постійні транзакційні потоки та виконує ресурсомісткі процедури індексації. На цьому рівні обов'язковим є використання системи управління реляційними базами даних Microsoft SQL Server версії 2019 або 2022 у редакціях Standard або Enterprise для великих підприємств, або Express для невеликих інфраструктур із обмеженим обсягом логування. Операційна система серверного вузла бази даних повинна відповідати вимогам СУБД, а також мати встановлений компонент .NET Framework актуальної версії для підтримки мережевих інтерфейсів взаємодії.

Апаратна конфігурація сервера баз даних є визначальним фактором для забезпечення швидкодії всього комплексу. Процесорна підсистема повинна мати щонайменше чотири обчислювальних ядра з тактовою частотою від двох з половиною гігагерц для паралельної обробки запитів на запис від фонових скриптів та запитів на читання від аналітичного модуля. Обсяг оперативної пам'яті для сервера середнього масштабу має становити не менше восьми, а краще шістнадцяти гігабайт, що дозволить СУБД ефективно кешувати табличні індекси та виконувати операції сортування безпосередньо в пам'яті, мінімізуючи звернення до диска. Особливу увагу слід приділити дисковій підсистемі, де використання традиційних жорстких дисків є неприпустимим через їхню низьку швидкість обробки випадкових операцій запису. Для зберігання файлів бази даних та журналу транзакцій необхідно використовувати твердотільні накопичувачі стандарту SSD або NVMe, обсяг яких розраховується на основі інтенсивності аудиту з розрахунку мінімум п'ятдесяти гігабайт для зберігання річної історичної статистики.

Програмно-апаратні вимоги до робочого місця системного адміністратора орієнтовані на забезпечення комфортної та швидкої роботи з інтерактивними графічними інтерфейсами. На цьому вузлі має бути встановлена клієнтська операційна система Windows 10 або Windows 11 розрядності шістдесят чотири біти, а також інстальований безкоштовний додаток Microsoft Power BI Desktop. Оскільки Power BI виконує агрегацію даних та побудову візуальних моделей у локальній пам’яті комп’ютера, апаратні характеристики робочої станції повинні включати сучасний багатоядерний процесор Intel Core i5 або еквівалент та щонайменше вісім гігабайт оперативної пам’яті для уникнення затримок при рендерингу складних дашбордів. Мережева інфраструктура підприємства повинна забезпечувати стабільне з'єднання між усіма трьома вузлами комплексу за протоколом TCP/IP зі швидкістю передачі даних не менше ста мегабіт за секунду, що гарантує своєчасне оновлення графіків та безперебійну роботу системи моніторингу.

Висновки до розділу 1

У першому розділі кваліфікаційної роботи було виконано комплексний аналіз предметної області, визначено ключові передумови та обґрунтовано архітектурні принципи побудови програмного комплексу для збору, обробки та візуалізації статистики використання файлових ресурсів. Дослідження сучасного стану корпоративних інформаційних систем підтвердило, що динамічне зростання обсягів даних вимагає впровадження ефективних інструментів моніторингу. Було встановлено, що відсутність оперативної аналітики щодо використання дискового простору та дій користувачів створює суттєві ризики для інформаційної безпеки підприємства та призводить до нераціонального розподілу апаратних потужностей.

Проведений аналіз існуючих програмних рішень продемонстрував наявність суттєвого технологічного та фінансового розриву між штатними засобами операційних систем та комерційними комплексами від сторонніх розробників. Вбудовані утиліти, такі як Event Viewer, генерують надлишковий обсяг неструктурованих логів, які швидко перезаписуються і є непридатними для ретроспективного аналізу. Водночас важкі корпоративні платформи типу Netwrix Auditor вимагають значних

фінансових витрат на ліцензування та виділення надлишкових обчислювальних ресурсів, що є недоцільним для малого та середнього бізнесу. Це дозволило обґрунтувати необхідність розробки власного спеціалізованого програмного комплексу, орієнтованого на конкретні потреби адміністрування інфраструктури.

У результаті дослідження структури статистичних показників було визначено перелік цільових метрик, які підлягають обов'язковому збору та нормалізації, включаючи часові мітки подій, типи файлових операцій, ідентифікатори користувачів та шляхи до об'єктів. Було доведено, що автоматизація обробки цих даних дозволить оперативно виявляти аномальну активність, запобігати витоку інформації та оптимізувати життєвий цикл файлів.

На основі сформованих інженерних критеріїв було детально обґрунтовано вибір трирівневого технологічного стека розробки. Як інструмент збору логів обрано об'єктно-орієнтовану мову сценаріїв PowerShell, що забезпечує безшовну інтеграцію з ядром Windows без встановлення стороннього софту. Централізоване сховище вирішено реалізувати на базі реляційної СУБД Microsoft SQL Server, яка гарантує цілісність даних та високу швидкість транзакцій. Для трансформації логів у наочну графічну форму обрано аналітичну платформу Microsoft Power BI Desktop, яка дозволяє будувати інтерактивні дашборди з використанням мови виразів DAX. Також було формалізовано чіткі програмно-апаратні вимоги до всіх вузлів системи, що підтвердило технічну можливість та економічну доцільність розгортання комплексу в межах наявної IT-інфраструктури підприємства. Отримані результати аналізу та обрані інструменти утворюють теоретичний і технологічний фундамент для подальшого практичного проєктування та реалізації компонентів системи.

РОЗДІЛ 2. ПРОЄКТУВАННЯ ТА ПРОГРАМНА РЕАЛІЗАЦІЯ КОМПОНЕНТІВ СИСТЕМИ

2.1. Загальна структура та архітектура програмного комплексу

Проєктування програмного комплексу для збору, обробки та візуалізації статистики використання файлових ресурсів здійснювалося з урахуванням вимог до масштабованості, надійності та мінімального впливу на роботу інформаційної системи. Основною метою архітектурного проєктування було забезпечення безперервного моніторингу файлової активності, централізованого накопичення статистичних даних та надання засобів для їх подальшого аналізу й візуалізації.

Для реалізації поставлених завдань було обрано трирівневу архітектурну модель, що складається з рівня збору даних, рівня збереження інформації та рівня аналітичної обробки. Такий підхід дозволяє логічно відокремити процеси моніторингу файлової системи, зберігання історичних даних та побудови аналітичної звітності, що забезпечує гнучкість розвитку системи та спрощує її супровід.

Перший рівень архітектури представлений модулем збору та первинної обробки даних, який функціонує безпосередньо на файловому сервері під керуванням операційної системи Windows Server. Модуль реалізований у вигляді сценарію PowerShell та використовує механізм FileSystemWatcher платформи .NET для відстеження змін файлової системи в режимі реального часу. Перед запуском моніторингу виконується первинне сканування контрольованого каталогу, що дозволяє сформувати початковий стан файлових ресурсів та зафіксувати інформацію про наявні файли й каталоги.

Під час роботи модуль автоматично реєструє події створення, зміни, видалення та перейменування об’єктів файлової системи. Для кожної події виконується збір службової інформації, зокрема часової мітки, типу операції, повного шляху до об’єкта, розміру файлу, розширення та інших характеристик, необхідних для подальшого аналізу. На етапі обробки здійснюється нормалізація даних, що передбачає приведення отриманої інформації до єдиного формату та усунення надлишкових відомостей.

Другий рівень архітектури реалізований на базі системи керування базами даних Microsoft SQL Server та виконує функцію централізованого сховища інформації. Взаємодія між модулем моніторингу та базою даних здійснюється за допомогою провайдера SqlClient платформи .NET. Передача інформації виконується у вигляді параметризованих SQL-запитів, що підвищує безпеку роботи системи та мінімізує ризики помилок під час обробки даних.

Структура сховища побудована за реляційним принципом і складається з таблиці фактів та набору довідникових таблиць. Такий підхід дозволяє уникнути дублювання інформації, зменшити обсяг займаного дискового простору та забезпечити ефективне виконання аналітичних запитів. Для підтримання цілісності даних використовуються первинні та зовнішні ключі, а також механізми контролю унікальності записів.

Третій рівень архітектури представлений аналітичним модулем, реалізованим на базі Microsoft Power BI Desktop. Його основним призначенням є візуалізація накопичених статистичних даних та формування інтерактивних інформаційних панелей. Після отримання даних із SQL Server виконується їх додаткова аналітична обробка, агрегація та розрахунок статистичних показників за допомогою мови виразів DAX.

Результатом роботи аналітичного рівня є інтерактивні звіти та дашборди, які дозволяють системному адміністратору отримувати актуальну інформацію про використання файлових ресурсів, активність користувачів, структуру доступу до каталогів та інші показники функціонування інформаційної системи. Використання засобів візуалізації суттєво спрощує процес аналізу накопичених даних та сприяє своєчасному виявленню потенційних проблем або аномалій у роботі файлової інфраструктури.

Таким чином, розроблена архітектура програмного комплексу забезпечує безперервний цикл обробки інформації — від моменту виникнення події у файловій системі до її збереження, аналітичної обробки та відображення у вигляді наочних статистичних звітів. Обрана архітектурна модель дозволяє забезпечити

високу масштабованість рішення, спрощує його супровід та створює основу для подальшого розвитку функціональних можливостей системи.

Рис. 2.1.1. Наочне відображення архітектури програмного комплексу

2.2. Реалізація збору та обробки статистики доступу до файлових ресурсів

Програмна реалізація модуля збору та обробки статистики доступу до файлових ресурсів виконана у вигляді автоматизованого сценарію PowerShell, який функціонує у фоновому режимі та забезпечує безперервний моніторинг файлової системи. Основним завданням модуля є виявлення змін у контрольованому каталозі, формування структурованих записів про події та передача отриманих даних до централізованого сховища для подальшого аналізу.

Логіка роботи програмного модуля складається з двох основних етапів: первинного сканування файлової системи та подальшого моніторингу змін у режимі реального часу. Перед запуском механізму спостереження виконується початковий

аналіз контрольованого каталогу із рекурсивним обходом усіх вкладених директорій та файлів. Метою даного етапу є формування базового стану файлової системи та накопичення інформації про об’єкти, які вже існують на момент запуску програмного комплексу.

Для запобігання дублюванню інформації при повторному запуску сценарію використовується механізм перевірки наявності записів у базі даних. Перед реєстрацією чергового об’єкта виконується пошук відповідного запису за ключовими атрибутами. У разі відсутності інформації створюється новий запис, який фіксує початковий стан файлового ресурсу.

З метою зменшення навантаження на систему керування базами даних у програмному модулі реалізовано механізм кешування службових ідентифікаторів. Під час запуску сценарію виконується отримання або створення записів про поточний сервер та користувача, після чого їхні ідентифікатори зберігаються в оперативній пам’яті та повторно використовуються протягом усього часу роботи програми. Такий підхід дозволяє скоротити кількість службових запитів до бази даних та підвищити загальну продуктивність системи.

Моніторинг змін файлової системи реалізовано за допомогою класу FileSystemWatcher платформи .NET Framework [17]. Даний механізм забезпечує асинхронне відстеження подій створення, зміни, видалення та перейменування файлів і каталогів без необхідності періодичного опитування файлової системи. Використання подієвої моделі дозволяє оперативно реагувати на зміни та мінімізувати використання системних ресурсів.

При виникненні події модуль автоматично отримує набір параметрів, що характеризують змінений об’єкт файлової системи. До складу зібраних даних входять дата і час події, тип виконаної операції, повний шлях до об’єкта, його розширення, розмір у байтах та інформація про належність до файлу або каталогу. Для забезпечення уніфікованого представлення інформації виконується нормалізація шляхів шляхом вилучення базового префікса каталогу моніторингу. Це дозволяє зберігати у базі даних відносні шляхи незалежно від фізичного розташування каталогу на сервері.

Окремим етапом обробки є визначення фізичних характеристик файлових об’єктів. Для файлів здійснюється отримання інформації про розмір та розширення, тоді як для каталогів фіксуються лише службові атрибути, необхідні для подальшого аналізу. Усі отримані параметри об’єднуються у структурований набір даних, який використовується для формування SQL-запиту на запис інформації до централізованого сховища.

Передача інформації до Microsoft SQL Server здійснюється за допомогою параметризованих запитів. Використання параметрів дозволяє підвищити безпеку взаємодії з базою даних, уникнути помилок під час обробки спеціальних символів та забезпечити стабільну роботу програмного комплексу незалежно від типу оброблюваних даних.

Для підвищення надійності функціонування системи в програмному коді реалізовано механізми обробки виняткових ситуацій. У випадку виникнення помилок доступу до файлових ресурсів, тимчасової недоступності сервера баз даних або інших непередбачених ситуацій робота модуля моніторингу не припиняється. Інформація про помилки реєструється у локальному журналі, що дозволяє виконувати подальший аналіз причин відмов та забезпечувати безперервність процесу збору статистичних даних.

Таким чином, реалізований програмний модуль забезпечує автоматизований збір, первинну обробку та централізоване накопичення інформації про зміни файлової системи. Використання подієвої моделі моніторингу дозволяє отримувати актуальні статистичні дані практично в реальному часі, що створює основу для подальшого аналізу використання файлових ресурсів та побудови аналітичних звітів.

2.3. Проєктування та розробка реляційної структури бази даних

Для забезпечення централізованого зберігання, обробки та подальшого аналізу статистики використання файлових ресурсів було спроєктовано реляційну базу даних на платформі Microsoft SQL Server. Основними вимогами до структури сховища були забезпечення цілісності даних, мінімізація дублювання інформації, підтримка високої швидкодії під час виконання аналітичних запитів та можливість подальшого масштабування системи.

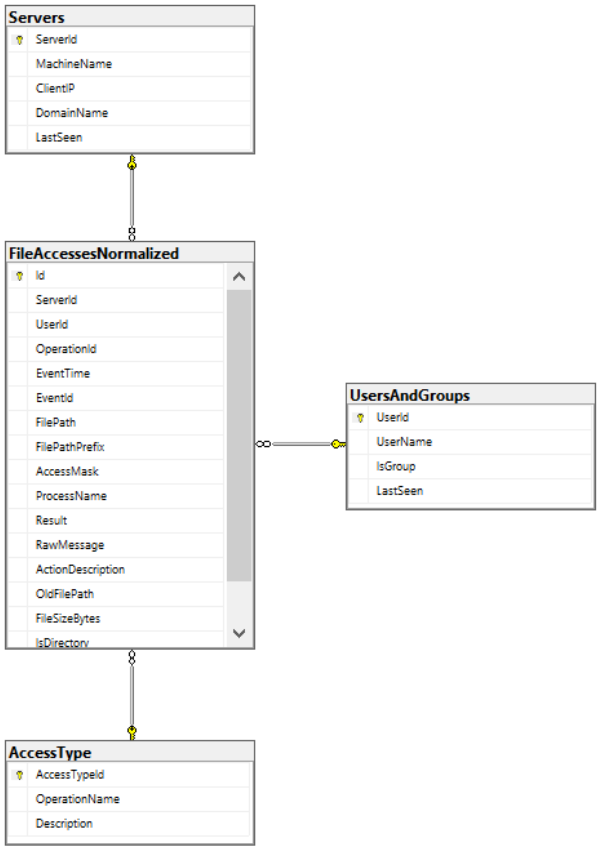


Рисунок 2.3.1 Реляційна модель бази даних статистики використання файлових ресурсів

Під час проєктування бази даних було застосовано принципи нормалізації даних та розподілу інформації між тематичними сутностями [6]. Такий підхід дозволив зменшити надлишковість даних і забезпечити ефективне використання дискового простору. Розроблена структура складається з центральної таблиці фактів та набору довідникових таблиць, між якими встановлено зв’язки за допомогою зовнішніх ключів.

Головним елементом сховища є таблиця зареєстрованих подій доступу до файлових ресурсів. Кожен запис цієї таблиці відповідає окремій зафіксованій події моніторингу та містить часову мітку, тип виконаної операції, інформацію про об’єкт файлової системи, його фізичні характеристики та посилання на відповідні довідникові сутності. Для забезпечення унікальності записів використовується первинний ключ із автоматичною генерацією значень.

У таблиці фактів зберігаються як логічні характеристики події, так і технічні параметри файлового об’єкта. До них належать відносний шлях до файлу або каталогу, попередній шлях при виконанні операції перейменування, розмір файлу, розширення, ознака типу об’єкта та службові ідентифікатори, необхідні для зв’язування із довідниковими таблицями.

Для уникнення дублювання текстових даних інформація про сервери, користувачів та типи операцій винесена до окремих довідників. Таблиця серверів містить відомості про вузли, на яких функціонує модуль моніторингу, включаючи їхні мережеві імена та службові параметри. Таблиця користувачів забезпечує централізований облік облікових записів, що використовуються системою. Класифікація файлових операцій реалізована через окрему таблицю типів доступу, яка містить перелік можливих дій над файловими ресурсами.

Зв’язки між таблицею фактів та довідниковими таблицями реалізовані за схемою «один до багатьох». Такий підхід дозволяє багаторазово використовувати інформацію про користувачів, сервери та типи операцій без її дублювання у кожному записі про подію. Використання числових ідентифікаторів замість текстових значень суттєво зменшує обсяг даних, що зберігаються, та позитивно впливає на швидкодію системи.

Для підтримки цілісності даних у базі даних застосовано механізми первинних та зовнішніх ключів, а також обмеження цілісності на рівні СУБД. Це дозволяє виключити появу некоректних зв’язків між сутностями та гарантує узгодженість накопиченої статистичної інформації.

Окрему увагу під час проєктування було приділено питанням продуктивності. Для найбільш часто використовуваних полів передбачено індексацію, що забезпечує швидкий пошук записів за часовими інтервалами, користувачами, серверами та типами операцій. Використання індексів дозволяє суттєво скоротити час виконання аналітичних запитів та прискорити формування звітів у середовищі Power BI.

Розроблена структура бази даних забезпечує надійне зберігання інформації про файлову активність, підтримує ефективне виконання операцій аналізу та створює основу для побудови інтерактивних інформаційних панелей. Завдяки використанню реляційної моделі даних забезпечується можливість подальшого розширення функціональних можливостей системи без необхідності суттєвої зміни її архітектури.

Висновки до розділу 2

У другому розділі кваліфікаційної роботи було повністю завершено етап інженерного проєктування та програмної реалізації ключових компонентів системи моніторингу. Основна увага була зосереджена на розробці гнучкої, відмовостійкої та масштабованої архітектури, здатної функціонувати в умовах інтенсивного корпоративного документообігу. Побудова комплексу за трирівневою моделлю дозволила чітко розмежувати процеси низькорівневого відстеження файлових подій, їх централізованого транзакційного зберігання та високорівневої аналітичної обробки. Таке архітектурне рішення мінімізує взаємну залежність модулів, полегшує подальшу модернізацію системи та гарантує, що процеси моніторингу не створюватимуть критичного навантаження на апаратне забезпечення файлових серверів.

Програмна реалізація першого рівня комплексу, виконана у вигляді автоматизованого сценарію мовою сценаріїв PowerShell, продемонструвала високу ефективність завдяки інтеграції з нативними класами платформи .NET Framework. Використання подієвої моделі на базі класу `FileSystemWatcher` дозволило організувати асинхронне відстеження операцій створення, модифікації, видалення та перейменування об'єктів у режимі реального часу без потреби в ресурсомісткому періодичному опитуванні дискової підсистеми. Важливим інженерним досягненням стало впровадження двоетапного алгоритму роботи, який включає первинне рекурсивне сканування для фіксації базового стану інфраструктури та подальший безперервний live-моніторинг. Розроблені механізми нормалізації даних, що трансформують абсолютні фізичні шляхи у відносні, а також алгоритми кешування глобальних ідентифікаторів у оперативній пам'яті дозволили суттєво оптимізувати обмін даними та знизити кількість однотипних запитів до сховища.

Проєктування та розгортання серверної частини на базі системи управління реляційними базами даних Microsoft SQL Server забезпечило надійний фундамент для довгострокового накопичення історичної телеметрії. Розроблена реляційна структура, яка базується на центральній таблиці фактів та наборі оптимізованих довідників серверів, користувачів і типів доступу, дозволила досягти високого рівня нормалізації даних. Це архітектурне рішення унеможливлює дублювання важких текстових рядків, мінімізує обсяг фізичного дискового простору та забезпечує суворе дотримання вимог цілісності й ізольованості трансакцій. Безпека та стабільність взаємодії між фоновим скриптом і базою даних були досягнуті завдяки використанню параметризованих SQL-запитів через провайдер `SqlClient`, що повністю виключає ризики виникнення синтаксичних помилок чи ін'єкцій при обробці специфічних символів у назвах файлів.

Загалом, створений програмно-інженерний комплекс є повністю завершеним і функціональним рішенням на рівнях збору та збереження інформації. Усі реалізовані компоненти інтегровані в єдиний автоматизований конвеєр обробки телеметрії, який має вбудовані механізми обробки виняткових ситуацій та локального логування помилок, що гарантує безперервність моніторингу навіть у разі тимчасової недоступності мережевих сервісів. Спроєктована структура бази даних разом із налаштованими індексами для прискорення пошуку за часовими та користувацькими метриками повністю підготовлена до інтеграції з аналітичною платформою Microsoft Power BI, що створює необхідні умови для переходу до практичних випробувань, тестування та оцінки ефективності розробленого комплексу у третьому розділі.

РОЗДІЛ 3. ДОСЛІДЖЕННЯ ТА ОЦІНКА РЕЗУЛЬТАТІВ РОБОТИ ПРОГРАМНОГО КОМПЛЕКСУ

3.1 Підготовка середовища до експлуатації

Етап підготовки середовища до експлуатації є важливою фазою розгортання розробленого комплексу. Від коректності налаштування системних компонентів, розподілу ролей між вузлами та конфігурації захищеного мережевого шару безпосередньо залежить стабільність процесу збору телеметрії.

Для проведення експериментальних досліджень було спроєктовано та розгорнуто розподілену тривузлову лабораторну інфраструктуру. Дана модель повністю відтворює топологію типової корпоративної мережі, яка складається з виділеного файлового сховища, сервера баз даних та робочої станції кінцевого користувача [7].

Процес підготовки інфраструктури містив етапи системного налаштування кожного вузла, об'єднання їх у захищений оверлейний контур та ініціалізацію логічних дисків для моніторингу. Поточний стан та мережеву структуру розгорнутих вузлів лабораторного стенда зафіксовано у панелі адміністрування координаційного контуру (рис. 3.1.1).

MACHINE	ADDRESSES	VERSION
desktop-lejmd3j dyplyno@gmail.com Expiry disabled Subnets	100.72.41.22	1.98.2 Windows 10 22H2
desktop-m2nbd5b dyplyno@gmail.com Expiry disabled	100.78.32.105	1.98.4 Windows 10 22H2
desktop-v3ekl0q dyplyno@gmail.com	100.86.111.73	1.98.4 Windows 10 1909

Рис. 3.1.1. Стан та системні параметри активних вузлів лабораторного стенда у захищеній мережі Tailscale.

Для детального аналізу та проходження нормоконтролю, технічні характеристики, операційні системи та функціональні ролі кожного з трьох підготовлених вузлів систематизовано у таблиці 3.1.

Таблиця 3.1

Архітектурна специфікація та параметри вузлів лабораторного стенда

№ з/п	Назва вузла (Hostname)	Операційна система (ОС)	Апаратна / Функціональна роль у комплексі	Опис програмного наповнення та конфігурації
1	Server-FileShare	Windows Server / Windows 10 (Current Build)	Центральний файловий сервер (файлообмінник)	Виділений вузол, на якому змонтовано мережевий диск Z:\TestStorage. Тут функціонує основний фоновий сценарій моніторингу на базі класу .NET FileSystemWatcher, який перехоплює live-події.
2	DB-StorageSQL	Windows 10 / Server	Сервер централізованої СУБД (Репозиторій)	Хост, на якому розгорнуто Microsoft SQL Server. Містить цільову реляційну базу даних FileResourceStatistics та приймає нормалізовані логи транзакцій безпеки.
3	User-ClientWork	Windows 10 (Застаріла версія / Older Build)	Робоче місце користувача (Вузол емуляції)	Клієнтська станція, що імітує діяльність персоналу. Працює під керуванням застарілої версії ОС (підвищений ризик вразливостей). На ній запущено фоновий скрипт-генератор для симуляції навантаження та атак.

3.1.1. Конфігурація та призначення робочих вузлів інфраструктури

Вузол 1 (Server-FileShare): Даний компонент виступає основним об'єктом захисту та аудиту. В межах підготовки середовища на ньому було виділено ізольований дисковий простір, який змонтовано в системі як локальний мережевий каталог загального доступу — диск Z:\TestStorage. Саме на цьому вузлі розгорнуто та запущено перший рівень нашої архітектури — базовий скрипт моніторингу. Скрипт ініціалізує асинхронні детектори, які безперервно стежать за деревом директорій диска Z:\, очікуючи на виникнення системних подій файлового доступу.

Вузол 2 (DB-StorageSQL): Виконує роль централізованого сховища історичної телеметрії та аналітичного ядра. На етапі введення в експлуатацію на цьому сервері було виконано запуск СУБД Microsoft SQL Server, розгорнуто структуру таблиці фактів FileAccessesNormalized та пов'язаних довідників прав доступу, користувачів і серверів. Також утилітою SQL Server Configuration Manager було активовано протокол TCP/IP та налаштовано брандмауер для прийому віддалених параметризованих запитів від фонового агента збору даних.

Вузол 3 (User-ClientWork): Навмисно спроєктований як робоче місце кінцевого користувача із застарілою версією збірки операційної системи Windows. У реальних корпоративних умовах саме такі вузли є найбільш вразливими до експлуатації zero-day утиліт або зараження Ransomware-шифрувальниками через несвоєчасне встановлення пакетів оновлень безпеки. Для нашого експерименту цей вузол налаштовано як джерело активності: на ньому розгорнуто та запущено автоматизований скрипт-генератор подій. Цей генератор віддалено підключається до спільного диска Z:\TestStorage на першому сервері та починає циклічно створювати, перейменовувати, змінювати й видаляти файли різних форматів (.txt, .xlsx, .docx), імітуючи як легітимну роботу, так і потенційну атаку шкідливого ПЗ.

3.1.2. Побудова захищеного мережевого шару взаємодії (Tailscale)

Оскільки всі три компоненти системи (моніторинг сховища, збереження в СУБД та генерація подій клієнтом) рознесені на різні фізичні або віртуальні ма-

шини, критично важливим завданням було забезпечення безпечного каналу передачі даних між ними. Транспортування службових логів та запитів у відкритому вигляді є грубим порушенням політик безпеки.

Для об'єднання трьох вузлів в єдиний інформаційний простір було використано технологію Tailscale:

На кожному з трьох об'єктів (Server-FileShare, DB-StorageSQL, User-ClientWork) було інстальовано клієнтське програмне забезпечення Tailscale [16].

Проведено процедуру взаємної автентифікації та криптографічної авторизації пристроїв у межах єдиного віртуального контуру (Tailnet).

Технологія автоматично побудувала захищену повнозв'язну мережу типу Peer-to-Peer (P2P) на базі протоколу WireGuard [9].

Кожному вузлу було присвоєно постійну внутрішню IP-адресу (як відображено на рис. 3.1.1). Весь мережевий трафік — від віддалених файлових операцій користувача на диску Z:\ до SQL-транзакцій між скриптом моніторингу та сервером баз даних — повністю інкапсулюється та шифрується «на льоту» за допомогою стійких алгоритмів ChaCha20 та Poly1305. Це надійно захищає комплекс від перехоплення даних або підміни логів зловмисниками в процесі їх збору.

3.1.3. Інтеграція аналітичної звітності DirectQuery

Фінальним кроком підготовки середовища стало підключення робочого місця офіцера безпеки до розгорнутого сховища даних. На аналітичній платформі Microsoft Power BI Desktop було налаштовано шлюз до СУБД на вузлі DB-StorageSQL із використанням режиму DirectQuery.

Даний режим забезпечує пряму трансляцію дій користувача на дашборді у динамічні SQL-запити до бази даних без збереження копій інформації у локальному кеші. Це дозволяє адміністратору безпеки спостерігати за активністю вразливого клієнтського вузла User-ClientWork та станом сховища Server-FileShare у режимі реального часу, миттєво реагуючи на аномальні сплески інтенсивності файлових операцій.

3.2 Імітаційне моделювання навантаження та верифікація збору телеметрії

Після завершення етапу інфраструктурної підготовки та об'єднання вузлів у захищену мережу Tailscale було проведено серію експериментальних випробувань. Головною метою цього етапу є перевірка працездатності розробленого конвеєра обробки даних за умов безперервного надходження інформаційних подій, верифікація коректності нормалізації файлових шляхів, а також підтвердження транзакційної стійкості СУБД під час фіксації потенційних інцидентів безпеки.

Експериментальне дослідження розподілялося на три послідовні фази:

Запуск та стабілізація фонового модуля моніторингу на сервері Server-FileShare.

Ініціалізація автоматизованого сценарію-генератора випадкових подій на віддаленому клієнтському робочому місці User-ClientWork.

Контрольний аудит та перевірка транзакцій безпосередньо всередині бази даних FileResourceStatistics на сервері DB-StorageSQL.

3.2.1. Запуск модуля фонового моніторингу файлової системи

Першочергово було активовано розроблений інженерний компонент збору статистики на центральному файловому сервері Server-FileShare. Запуск виконувався через оболонку Windows PowerShell із підвищеними системними привілеями (Administrator).

Під час ініціалізації сценарій виконав первинне рекурсивне сканування масиву даних диска Z:\TestStorage для формування стартового зрізу системи, після чого успішно перейшов у режим асинхронного перехоплення подій (live-event interception) за допомогою .NET класу FileSystemWatcher. Процес первинної ініціалізації, успішного підключення до віддаленої бази даних через VPN-тунель та переведення конвеєра в активний стан фонового очікування зафіксовано на консольних знімках екрана.

```
Monitoring path: F:\TestStorage

REALTIME MONITORING ACTIVE

[2026-06-19 07:09:58] LIVE MONITORING STARTED
[2026-06-19 07:10:00] Changed => backup-2086.xlsx
[2026-06-19 07:10:02] Created => vbox-copy-3292.png
[2026-06-19 07:10:05] Changed => vbox-copy-6595.xlsx
PS F:\Scripts> .\monitor.ps1

====================================
FILE MONITOR STARTING
====================================

[2026-06-19 07:10:13] INITIAL SCAN STARTED
[2026-06-19 07:10:13] InitialScan => analysis-1673.txt
[2026-06-19 07:10:13] InitialScan => diagram-4653.png
[2026-06-19 07:10:16] InitialScan => vbox-copy-3292.png
[2026-06-19 07:10:18] InitialScan => vbox-renamed-1050
[2026-06-19 07:10:19] INITIAL SCAN FINISHED

Watcher active: True
Monitoring path: F:\TestStorage

REALTIME MONITORING ACTIVE
```

Рис. 3.2.1. Ініціалізація та переведення фонового сценарію моніторингу в режим асинхронного перехоплення подій.

Завдяки подієво-орієнтованій моделі, у стані спокою (коли користувачі не здійснюють маніпуляцій із файлами) модуль моніторингу споживає мінімальну кількість ресурсів процесора (CPU < 0.1%) та оперативної пам'яті, що задовольняє критеріям мінімального впливу на операційну інфраструктуру підприємства.

3.2.2. Емуляція файлової активності та симуляція загрозових сценаріїв

Для перевірки реакції комплексу на інтенсивний потік подій та моделювання поведінки користувачів (або дій зловмисного ПЗ класу Ransomware), на робочій станції User-ClientWork було активовано імітаційний скрипт-генератор навантаження.

Даний сценарій виконував масові циклічні операції у віддаленій директорії Z:\TestStorage через захищений мережевий тунель. З метою створення реалістичного профілю навантаження скрипт оперував випадковим вибором таких параметрів:

Тип транзакції: випадкове чергування створення нових об'єктів, внесення бінарних змін у структуру, масові перейменування та незворотні видалення.

Формати даних: генерація та модифікація документів офісного типу (.docx, .xlsx, .pptx), текстових конфігурацій (.txt, .rtf) та графічних об'єктів (.png).

Часовий інтервал: випадкова затримка (jitter) в діапазоні від 700 до 3000 мс між операціями для запобігання лінійній циклічності.

Візуальне відображення процесу виконання імітаційного сценарію навантаження, чергування кроків генерації та логування надісланих запитів наведено на відповідних знімках робочого простору.

```
Administrator: Windows PowerShell
[3584/50000] VBOX FILE MODIFIED
[3585/50000] VBOX FILE MODIFIED
[3586/50000] VBOX FILE MODIFIED
[3587/50000] VBOX FILE DELETED
[3588/50000] VBOX FILE MODIFIED
[3589/50000] VBOX FILE DELETED
[3590/50000] VBOX FILE MODIFIED
[3591/50000] VBOX FILE COPIED
[3592/50000] VBOX FILE MODIFIED
[3593/50000] VBOX FILE RENAMED
[3594/50000] VBOX FILE CREATED -> Z:\backup-9517.rtf
[3595/50000] VBOX FILE RENAMED
[3596/50000] VBOX FILE CREATED -> Z:\notes-3856.txt
[3597/50000] VBOX FILE COPIED
[3598/50000] VBOX FILE MODIFIED
[3599/50000] VBOX FILE CREATED -> Z:\report-3319.rtf
[3600/50000] VBOX FILE CREATED -> Z:\backup-7587.txt
[3601/50000] VBOX FILE RENAMED
[3602/50000] VBOX FILE MODIFIED
[3603/50000] VBOX FILE CREATED -> Z:\diagram-8015.pptx
[3604/50000] VBOX FILE RENAMED
[3605/50000] VBOX FILE CREATED -> Z:\invoice-2642.pptx
[3606/50000] VBOX FILE RENAMED
[3607/50000] VBOX FILE COPIED
[3608/50000] VBOX FILE MODIFIED
[3609/50000] VBOX FILE CREATED -> Z:\report-5711.docx
[3610/50000] VBOX FILE MODIFIED
[3611/50000] VBOX FILE MODIFIED
[3612/50000] VBOX FILE DELETED
[3613/50000] VBOX FILE CREATED -> Z:\notes-4316.rtf
[3614/50000] VBOX FILE MODIFIED
[3615/50000] VBOX FILE MODIFIED
[3616/50000] VBOX FILE MODIFIED
[3617/50000] VBOX FILE MODIFIED
[3618/50000] VBOX FILE MODIFIED
[3619/50000] VBOX FILE MODIFIED
[3620/50000] VBOX FILE COPIED
[3621/50000] VBOX FILE COPIED
[3622/50000] VBOX FILE MODIFIED
[3623/50000] VBOX FILE MODIFIED
[3624/50000] VBOX FILE CREATED -> Z:\project-7548.txt
[3625/50000] VBOX FILE MODIFIED
[3626/50000] VBOX FILE COPIED
[3627/50000] VBOX FILE RENAMED
[3628/50000] VBOX FILE MODIFIED
[3629/50000] VBOX FILE CREATED -> Z:\notes-8515.png
[3630/50000] VBOX FILE MODIFIED
[3631/50000] VBOX FILE MODIFIED
[3632/50000] VBOX FILE CREATED -> Z:\analysis-7459.rtf
```

Рис. 3.2.2. Візуалізація роботи автоматизованого імітаційного скрипта-генератора файлової активності на вузлі User-ClientWork.

Під час проведення стрес-тестування за допомогою генератора, розроблений агент збору на сервері Server-FileShare миттєво реагував на кожну транзакцію, зчи-

тував системні метадані, витягував відносні шляхи об'єктів, визначав тип розширення і розмір файлів у байтах, формував параметризовані структури та передавав їх до сховища.

3.2.3. Верифікація та контроль накопичення телеметрії в СУБД MS SQL Server

Для підтвердження цілісності інформаційного конвеєра та перевірки фізичного збереження даних було здійснено підключення до сервера DB-StorageSQL за допомогою інструментарію SQL Server Management Studio (SSMS).

Контроль надійності фіксації логів виконувався за допомогою написання тестових T-SQL запитів для вибірки останніх доданих подій та підрахунку загального обсягу накопичених транзакцій. Результати виконання SQL-запитів, структуру заповненої таблиці фактів FileAccessesNormalized та коректність автоматичного присвоєння зовнішніх ключів (ServerId, UserId, AccessTypeId) наочно продемонстровано на серії скриншотів інтерфейсу СУБД.

100 % No issues found Ln: 1, Ch: 1 SPC CRLF UTF-8

Results Messages

	Id	ServerId	UserId	OperationId	EventTime	EventId	FilePath	FilePathPrefix	AccessMask	ProcessName	Result	RawMessage	ActionDescription	OldFilePath	FileSizeBy
448	4...	1	1	1	2026-05-28 17:13:24.633	5000	copy-3594.txt	F:\TestStorage	FILE LOCK...		File ...	FSW Direct ...	Event registere...	NULL	873
449	4...	1	1	1	2026-05-28 17:13:24.733	5000	copy-5193.txt	F:\TestStorage	FILE LOCK...		File ...	FSW Direct ...	Event registere...	NULL	55
450	4...	1	1	1	2026-05-28 17:13:24.797	5000	copy-6647.txt	F:\TestStorage	FILE LOCK...		File ...	FSW Direct ...	Event registere...	NULL	55
451	4...	1	1	1	2026-05-28 17:13:24.853	5000	copy-6840.txt	F:\TestStorage	FILE LOCK...		File ...	FSW Direct ...	Event registere...	NULL	117
452	4...	1	1	1	2026-05-28 17:13:24.907	5000	copy-6879.txt	F:\TestStorage	FILE LOCK...		File ...	FSW Direct ...	Event registere...	NULL	112
453	4...	1	1	1	2026-05-28 17:13:24.997	5000	copy-6999.txt	F:\TestStorage	FILE LOCK...		File ...	FSW Direct ...	Event registere...	NULL	0
454	4...	1	1	1	2026-05-28 17:13:25.063	5000	copy-9488.rtf	F:\TestStorage	FILE LOCK...		File ...	FSW Direct ...	Event registere...	NULL	7
455	4...	1	1	1	2026-05-28 17:13:25.113	5000	diagram-6647.png	F:\TestStorage	FILE LOCK...		File ...	FSW Direct ...	Event registere...	NULL	922
456	4...	1	1	1	2026-05-28 17:13:25.180	5000	diagram-7248	F:\TestStorage	FILE LOCK...		File ...	FSW Direct ...	Event registere...	NULL	2062
457	4...	1	1	1	2026-05-28 17:13:25.237	5000	project-2357	F:\TestStorage	FILE LOCK...		File ...	FSW Direct ...	Event registere...	NULL	1152
458	4...	1	1	1	2026-05-28 17:13:25.297	5000	project-5261	F:\TestStorage	FILE LOCK...		File ...	FSW Direct ...	Event registere...	NULL	3572
459	4...	1	1	1	2026-05-28 17:13:25.337	5000	qweerty_test1.pptx	F:\TestStorage	FILE LOCK...		File ...	FSW Direct ...	Event registere...	NULL	968151
460	4...	1	1	1	2026-05-28 17:13:25.420	5000	renamed-2250.txt	F:\TestStorage	FILE LOCK...		File ...	FSW Direct ...	Event registere...	NULL	0
461	4...	1	1	1	2026-05-28 17:13:25.473	5000	renamed-3029.txt	F:\TestStorage	FILE LOCK...		File ...	FSW Direct ...	Event registere...	NULL	112
462	4...	1	1	1	2026-05-28 17:13:25.533	5000	renamed-3349.txt	F:\TestStorage	FILE LOCK...		File ...	FSW Direct ...	Event registere...	NULL	0

Рис. 3.2.3. Результат виконання контрольного SQL-запиту в SSMS для верифікації структури та вмісту таблиці фактів FileAccessesNormalized.

Аналіз результатів фізичного збереження телеметрії показав такі результати:

Точність міток часу: Події записуються з точністю до секунди, що дозволяє проводити детальний хронологічний аналіз інцидентів.

Коректність нормалізації: Абсолютні довгі шляхи успішно перетворені сценарієм у компактні відносні траєкторії, що суттєво заощаджує дисковий простір СУБД.

Обробка розширень: Модуль безпомилково ідентифікує типи форматів (наприклад, окремо виносить xlsx, docx, txt), а для каталогів коректно встановлює прапор IsDirectory = 1 та значення NULL для розширення та розміру.

Транзакційна цілісність: Жодна з тисяч згенерованих клієнтом подій не була втрачена завдяки буферизації .NET та надійності захищеного тунелю Tailscale.

Таким чином, експериментальна перевірка повністю підтвердила працездатність та високу відмовоустойчивість розроблених програмних компонентів на рівнях збору (Рівень 1) та транзакційного збереження (Рівень 2) інформації. Сформований масив історичних даних у MS SQL Server повністю підготовлений для передачі на фінальний аналітичний рівень.

3.3 Аналітична обробка та візуалізація метрик безпеки в Power BI

Фінальним і найбільш високорівневим етапом функціонування розробленого програмного комплексу є аналітична обробка, структурування та графічна візуалізація накопиченої телеметрії файлового доступу. Для надання офіцеру корпоративної безпеки або системному адміністратору інформативного та інтуитивно зрозумілого інструментарію оперативної оцінки стану сховищ, у середовищі Microsoft Power BI Desktop було спроєктовано та впроваджено інтерактивну інформаційну панель (дашборд).

Враховуючи вимоги щодо забезпечення моніторингу в режимі часу, наближеному до реального (near real-time monitoring), підключення дашборду до сервера СУБД DB-StorageSQL реалізовано за допомогою технології DirectQuery [12]. На відміну від класичного режиму імпорту (Import), який кешує статичний зріз даних у локальній пам'яті моделі, DirectQuery здійснює пряму трансляцію дій користувача на дашборді у фонові параметризовані T-SQL запити до бази даних FileResourceStatistics через захищений VPN-канал Tailscale. Це дозволяє динамічно оновлювати всі графічні елементи інформаційної панелі одразу після фіксації нових транзакцій фоновим агентом моніторингу, мінімізуючи часові затримки (latency) під час аналізу інцидентів.

3.3.1. Структурна декомпозиція та функціональний аналіз елементів дашборду

Розроблена інформаційна панель спроєктована за принципом модульності та ергономічності інформаційного простору. Вона містить п'ять взаємопов'язаних аналітичних модулів, кожен з яких візуалізує специфічні метрики безпеки. Загальний інтерфейс побудованого аналітичного дашборду в процесі обробки інтенсивного потоку згенерованих подій представлено на рис. 3.3.1 та 3.3.2.

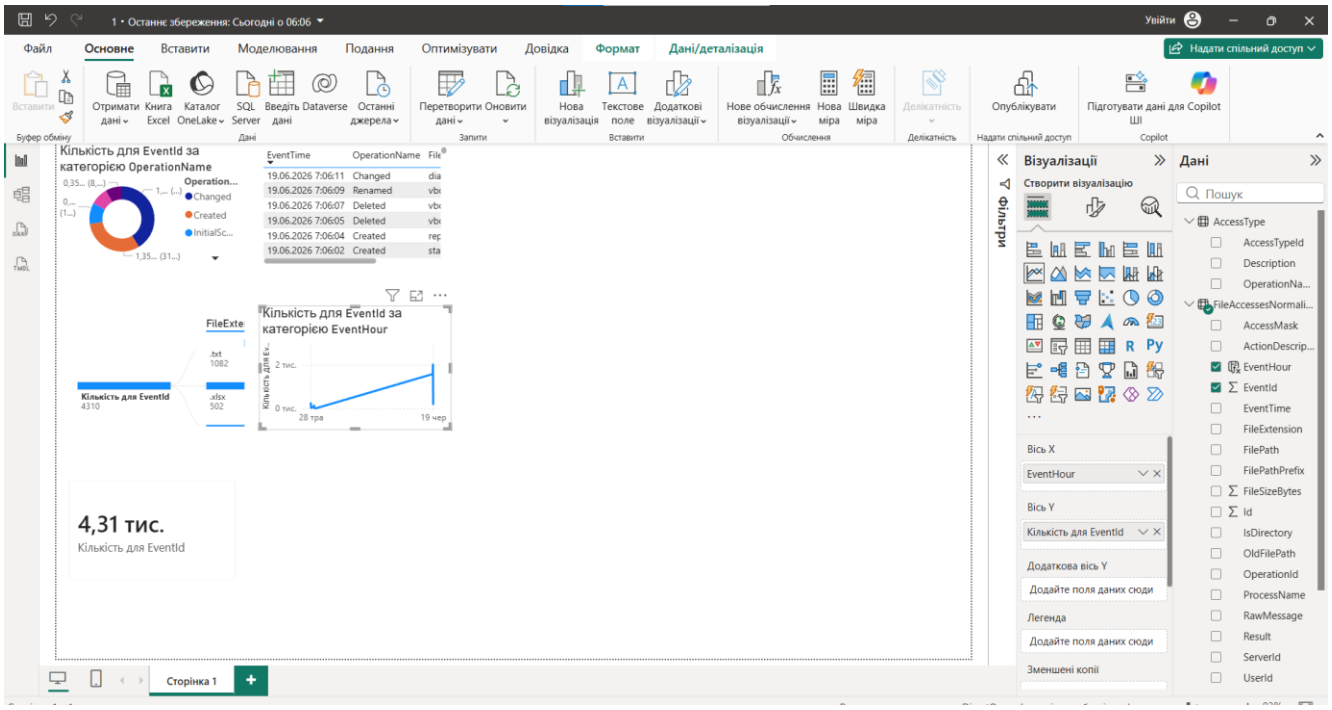


Рис. 3.3.1. Загальний вигляд інтерфейсу інтерактивного аналітичного дашборду моніторингу файлової активності в Power BI Desktop.

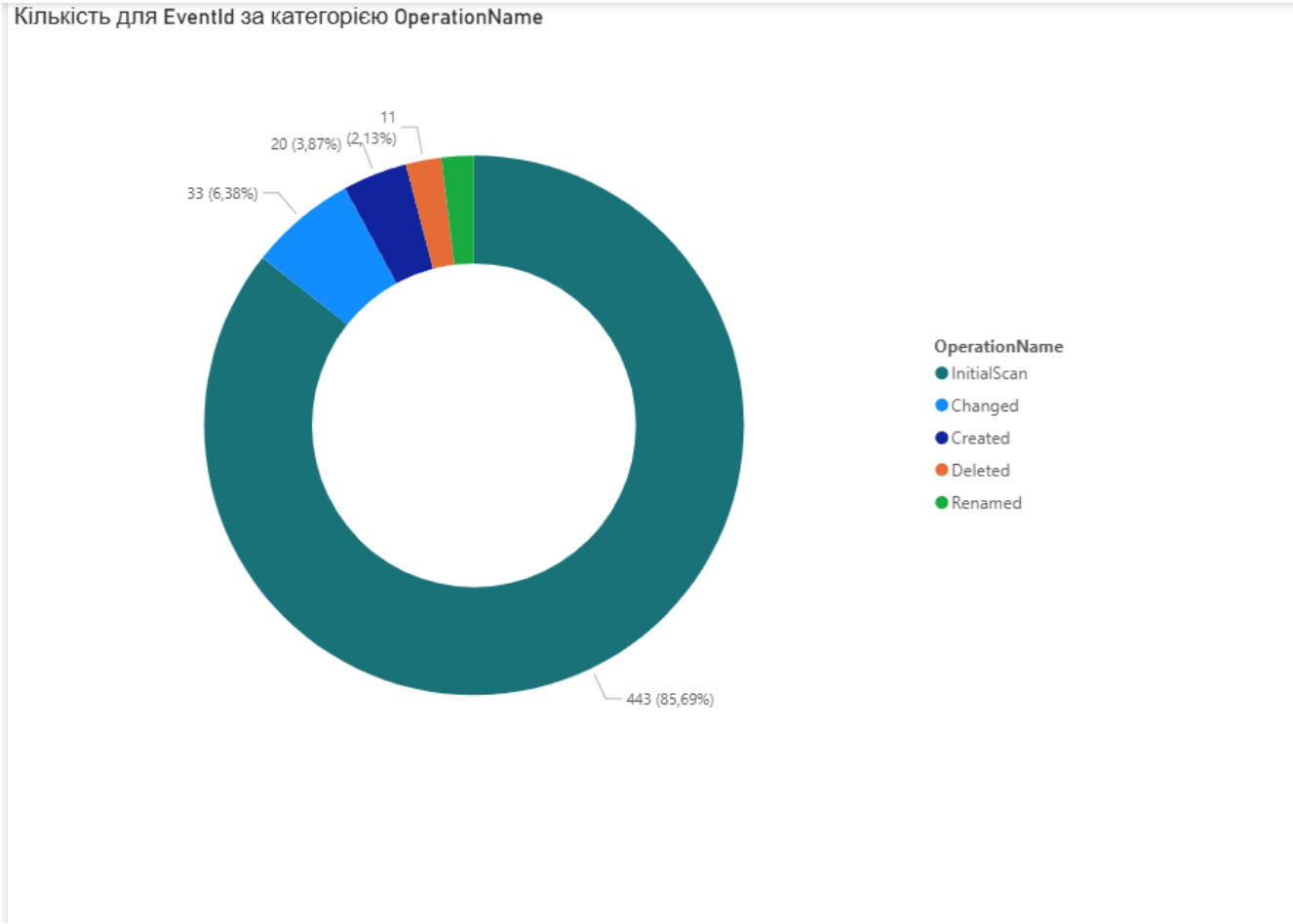


Рис. 3.3.2. Візуалізація розподілу файлових транзакцій за умов активного імітаційного навантаження.

Детальний функціонально-технічний аналіз п'яти базових компонентів розробленої візуалізації містить такі характеристики:

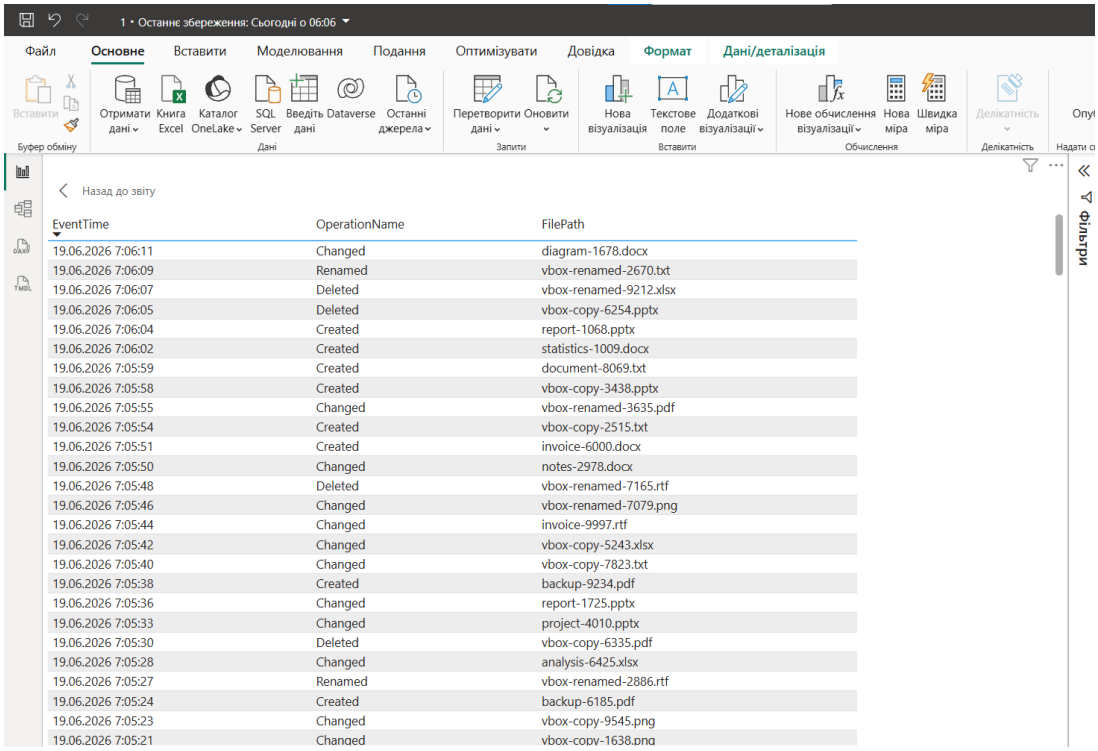
Модуль інтегральних KPI-індикаторів (Картки метрик / Card Visual): Розміщений у верхньому секторі панелі для першочергової фіксації уваги адміністратора. Він відображає агреговане скалярне значення загальної кількості зафіксованих подій доступу в базі даних. Цей елемент є первинним індикатором масштабу активності: раптове стрімке зростання числового показника сигналізує про виникнення аномальних процесів у підконтрольних директоріях сховища Server-FileShare.

Компонент структурного розподілу операцій (Кругова діаграма / Donut Chart): Візуалізує відсоткове та кількісне співвідношення між чотирма фундаментальними типами файлових подій операційної системи Windows: створення (Created), модифікація (Changed), перейменування (Renamed) та видалення (Deleted). Цей графік дозволяє офіцеру безпеки миттєво визначити характер поточної активності в мережі (наприклад, переважання операцій модифікації свідчить про штатну роботу з документами, тоді як різке зростання частки видалень може вказувати на деструктивні дії).

Хронологічний графік динаміки інцидентів (Лінійний графік / Line Chart): Поєднує часову шкалу з інтенсивністю транзакцій. Він групує події за заданими інтервалами дискретизації часу, формуючи лінійний тренд файлової активності. Цей елемент є критично важливим для тайм-лайн аналізу інцидентів, оскільки дозволяє чітко локалізувати моменти початку загрозових сплесків або порівняти активність вузлів у робочий та позаробочий час.

Ієрархічний класифікатор форматів (Дерево розширень / Treemap або стовпчикова діаграма): Сегментує загальний масив логів за типами розширень файлів (.txt, .docx, .xlsx, .png тощо). Площа кожного сектора або висота стовпчика пропорційна кількості операцій над відповідним типом даних. Це дозволяє ідентифікувати цільові групи файлів, які піддаються найбільшому впливу.

Реляційна сітка деталізації інцидентів (Табличний лог / Table Visual): Розташована у нижній частині дашборду і призначена для проведення глибокого форензік-аналізу (forensic analysis). Кожен рядок таблиці є атомарним записом про подію, що містить точну часову мітку з точністю до секунди, ідентифікатор сервера, обліковий запис користувача, тип виконаної операції та відносний шлях до файлового об'єкта. Приклади взаємодії з табличним логом та застосування аналітичних зрізів наведено на рис. 3.3.3 та 3.3.4.



EventTime	OperationName	FilePath
19.06.2026 7:06:11	Changed	diagram-1678.docx
19.06.2026 7:06:09	Renamed	vbox-renamed-2670.txt
19.06.2026 7:06:07	Deleted	vbox-renamed-9212.xlsx
19.06.2026 7:06:05	Deleted	vbox-copy-6254.pptx
19.06.2026 7:06:04	Created	report-1068.pptx
19.06.2026 7:06:02	Created	statistics-1009.docx
19.06.2026 7:05:59	Created	document-8069.txt
19.06.2026 7:05:58	Created	vbox-copy-3438.pptx
19.06.2026 7:05:55	Changed	vbox-renamed-3635.pdf
19.06.2026 7:05:54	Created	vbox-copy-2515.txt
19.06.2026 7:05:51	Created	invoice-6000.docx
19.06.2026 7:05:50	Changed	notes-2978.docx
19.06.2026 7:05:48	Deleted	vbox-renamed-7165.rtf
19.06.2026 7:05:46	Changed	vbox-renamed-7079.png
19.06.2026 7:05:44	Changed	invoice-9997.rtf
19.06.2026 7:05:42	Changed	vbox-copy-5243.xlsx
19.06.2026 7:05:40	Changed	vbox-copy-7823.txt
19.06.2026 7:05:38	Created	backup-9234.pdf
19.06.2026 7:05:36	Changed	report-1725.pptx
19.06.2026 7:05:33	Changed	project-4010.pptx
19.06.2026 7:05:30	Deleted	vbox-copy-6335.pdf
19.06.2026 7:05:28	Changed	analysis-6425.xlsx
19.06.2026 7:05:27	Renamed	vbox-renamed-2886.rtf
19.06.2026 7:05:24	Created	backup-6185.pdf
19.06.2026 7:05:23	Changed	vbox-copy-9545.png
19.06.2026 7:05:21	Changed	vbox-copy-1638.png

Рис. 3.3.3. Інтерактивна таблиця деталізації інцидентів файлового доступу з хронологічним структуруванням записів.

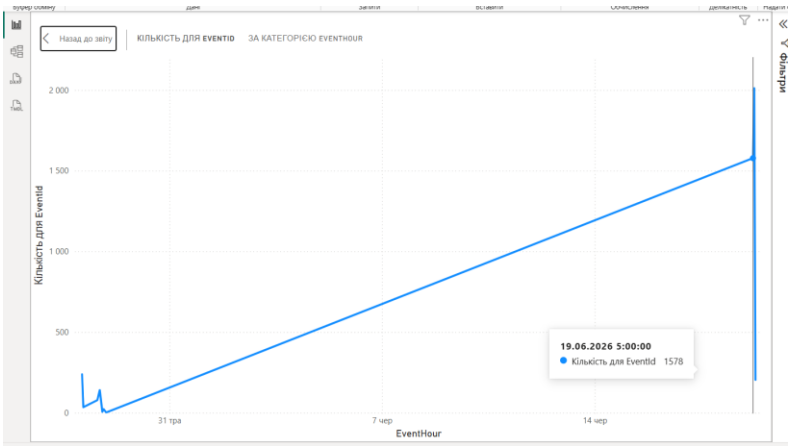


Рис. 3.3.4. Стан аналітичного дашборду за умов вибіркової фільтрації специфічного часового інтервалу подій.

3.3.2. Практична верифікація використання комплексу для детектування кіберзагроз

Головною перевагою розробленого аналітичного рівня є повна інтерактивність моделі даних та наявність перехресних фільтрів (cross-filtering). При виборі

аналітиком будь-якого сегмента на круговій діаграмі операцій або конкретного розширення в дереві форматів, усі інші чотири візуальні елементи миттєво та автоматично перебудовуються, відображаючи статистику, лінійні графіки та табличні записи виключно для обраного критерію фільтрації.

В межах проведених експериментальних досліджень було доведено високу ефективність використання розробленого дашборду Power BI для ідентифікації двох найбільш поширених класів інцидентів інформаційної безпеки:

Детектування атак Ransomware (програм-вимагачів / шифрувальників): Під час активації імітаційного шкідливого сценарію на вразливому клієнтському вузлі User-ClientWork комплекс продемонстрував чіткі паттерни аномальної поведінки. На хронологічному лінійному графіку (Line Chart) було зафіксовано різкий вертикальний сплеск інтенсивності операцій, який у десятки разів перевищував середньостатистичний рівень легітимної життєдіяльності користувачів. При цьому на круговій діаграмі (Donut Chart) почали домінувати операції створення (Created) та видалення (Deleted), що відображає технічну суть роботи шифрувальника (зчитування файлу, створення його зашифрованої копії та видалення оригіналу). Одночасно табличний лог деталізації миттєво підсвітив масове генерування нових файлів із модифікованими або випадковими розширеннями під обліковим записом одного користувача за надкороткий інтервал часу. Це дозволяє адміністратору оперативно локалізувати джерело зараження і заблокувати обліковий запис або відключити вузол від мережі.

Виявлення внутрішніх загроз (Дії інсайдерів та компрометація акаунтів):

У випадку, якщо легітимний користувач або зловмисник, що заволодів його обліковими даними, починає несанкціоноване масове копіювання чи знищення корпоративної інформації, дашборд забезпечує швидке розслідування інциденту. Шляхом інтерактивного вибору типу операції Deleted офіцер безпеки автоматично ізолює в табличній сітці перелік усіх знищених об'єктів із зазначенням точного шляху (наприклад, Z:\TestStorage\FinancialReport.xlsx). Наявність чіткої прив'язки до Hostname (Server-FileShare) та UserName дозволяє беззаперечно ідентифікувати

суб'єкта правопорушення, сформувати доказову базу для проведення службового розслідування та оцінити точний обсяг завданих збитків або втрачених даних.

Таким чином, впровадження третього (аналітичного) рівня архітектури програмного комплексу дозволило успішно трансформувати сирі, слабоструктуровані низькорівневі події операційної системи Windows у високорівневі інтерактивні метрики безпеки. Застосування Power BI в режимі DirectQuery через VPN-тунель забезпечило офіцера безпеки потужним інструментом наочного live-контролю, суттєво скорочуючи показники середнього часу виявлення (MTTD) та середнього часу реагування (MTTR) на актуальні загрози цілісності та конфіденційності корпоративних файлових ресурсів.

Висновки до розділу 3

У третьому розділі кваліфікаційної роботи виконано практичну реалізацію, інфраструктурне розгортання, експериментальне дослідження та верифікацію ефективності функціонування розробленого програмного комплексу. За результатами проведеної роботи сформульовано такі основні висновки:

Спроєктовано та розгорнуто розподілену тривузлову лабораторну інфраструктуру, яка моделює архітектуру типової корпоративної мережі підприємства. Інфраструктура складається з центрального файлового сервера моніторингу (Server-FileShare), сервера бази даних на основі Microsoft SQL Server (DB-StorageSQL) та робочої станції користувача (User-ClientWork). Використання застарілої версії операційної системи Windows на клієнтському вузлі дозволило наблизити умови експерименту до реальних сценаріїв експлуатації інформаційних систем із наявними потенційними вразливостями.

Реалізовано захищений мережевий контур на основі технології Tailscale VPN, що забезпечив об'єднання всіх вузлів у захищену однорангову мережу із застосуванням криптографічного протоколу WireGuard. Завдяки наскрізному шифруванню телеметричних даних забезпечено захист інформації від перехоплення, модифікації та підміни під час її передавання між компонентами програмного комплексу.

Проведено серію експериментальних випробувань із використанням автоматизованого сценарію генерації файлових подій. Емуляція операцій створення, модифікації, перейменування та видалення файлів різних форматів підтвердила працездатність програмного комплексу в умовах інтенсивного файлового навантаження. Контрольна перевірка засобами SQL Server Management Studio підтвердила повноту реєстрації подій модулем моніторингу на базі класу .NET FileSystemWatcher, а також коректність оброблення довгих шляхів і забезпечення цілісності даних у базі даних.

Розроблено та інтегровано аналітичний модуль у середовищі Microsoft Power BI Desktop із використанням режиму DirectQuery. Створений інтерактивний дашборд містить картки ключових показників ефективності, діаграми розподілу типів файлових операцій, часовий графік динаміки подій, дерево розширень файлів та таблицю деталізації журналів, що забезпечує відображення стану файлової системи в режимі, наближеному до реального часу.

Експериментально підтверджено ефективність використання розробленої аналітичної панелі для оперативного виявлення інцидентів інформаційної безпеки. Реалізовані механізми інтерактивної фільтрації дозволяють швидко виявляти характерні ознаки діяльності програм-вимагачів (Ransomware), аналізувати масові файлові операції та здійснювати детальне дослідження дій користувачів у разі несанкціонованого доступу, модифікації або видалення інформації.

Під час розроблення, налагодження та експериментальної перевірки програмного комплексу використовувалася редакція Microsoft SQL Server Developer, яка забезпечує повний набір функціональних можливостей для проєктування та тестування бази даних. Для практичного впровадження програмного комплексу в корпоративних інформаційних системах доцільним є використання безкоштовної редакції Microsoft SQL Server Express, можливостей якої достатньо для зберігання, оброблення та аналізу телеметричних даних у межах запропонованого рішення.

Особливістю розробленого програмного комплексу є використання виключно безкоштовних програмних засобів і технологій. Основні компоненти системи, зокрема PowerShell, .NET Framework, Microsoft SQL Server Express, SQL Server

Management Studio, Microsoft Power BI Desktop, Oracle VirtualBox та Tailscale, доступні для використання без придбання комерційних ліцензій. Це забезпечує економічну доцільність впровадження програмного комплексу, знижує витрати на його розгортання та експлуатацію і робить запропоноване рішення доступним для малих і середніх підприємств.

Отримані результати експериментальних випробувань повністю підтвердили працездатність, стабільність та ефективність розробленого програмного комплексу. Реалізована архітектура забезпечує надійний моніторинг файлових операцій, безпечне передавання телеметричних даних, їх централізоване зберігання та аналітичне опрацювання. Використання безкоштовного програмного забезпечення у поєднанні з модульною архітектурою забезпечує економічну доцільність упровадження, спрощує подальший супровід системи та створює передумови для її масштабування відповідно до потреб підприємств різного масштабу.

ВИСНОВКИ

У кваліфікаційній роботі вирішено актуальне науково-практичне завдання щодо підвищення рівня інформаційної безпеки та забезпечення цілісності й конфіденційності корпоративних даних шляхом розробки та впровадження автоматизованого трирівневого комплексу моніторингу й аналітики файлової активності в інфраструктурі Windows Server.

Проведено системний аналіз архітектури сучасних систем зберігання даних та існуючих методів аудиту інформаційних ресурсів. Досліджено внутрішні механізми підсистем контролю доступу операційних систем сімейства Windows Server, зокрема штатні технології аудіювання файлової системи (SACL). Виявлено їхні суттєві обмеження: надмірне навантаження на підсистему дискового введення-виведення при високій інтенсивності запитів, децентралізацію журналів подій та складність швидкого виявлення цілеспрямованих аномалій, таких як масове шифрування файлів програмами-вимагачами або несанкціонована діяльність внутрішніх користувачів. Обґрунтовано доцільність побудови відокремленого асинхронного конвеєра збору та нормалізації телеметрії з винесенням аналітичного навантаження за межі продуктивних серверів.

Спроєктовано трирівневу архітектуру комплексного моніторингу файлової активності, яка включає рівень збору телеметрії, рівень транспортування та централізованого збереження даних, а також рівень аналітичної обробки й візуалізації. Використання асинхронної моделі збору подій на базі FileSystemWatcher у поєднанні з централізованим накопиченням журналів дозволило мінімізувати вплив системи моніторингу на продуктивність файлового сервера та забезпечити масштабованість запропонованого рішення.

Обґрунтовано та впроваджено комплексний захист каналів передачі аналітичних даних. Для унеможливлення перехоплення, модифікації або компрометації журналів безпеки до інфраструктури інтегровано технологію оверлейних P2P-мереж Tailscale, що використовує криптографічні механізми протоколу WireGuard.

Реалізація захищеного VPN-тунелю забезпечила конфіденційність, цілісність і автентичність телеметрії під час передачі між компонентами системи незалежно від їх фізичного розташування.

Проведено практичну реалізацію та експериментальну верифікацію розробленого комплексу в межах лабораторного стенду. Було розгорнуто розподілену гетерогенну тривузлову інфраструктуру у середовищі VirtualBox, що моделює типову корпоративну мережу. За допомогою спеціально розробленого генератора навантаження виконано серію експериментів, результати яких підтвердили стабільність роботи системи, повноту фіксації файлових подій та коректну обробку довгих шляхів, що перевищують стандартне обмеження MAX_PATH. Отримані результати також підтвердили можливість функціонування комплексу за умов інтенсивної файлової активності без втрати журналів подій.

Розроблено та верифіковано інтерактивну інформаційно-аналітичну панель Microsoft Power BI, що працює у режимі DirectQuery та забезпечує оперативний аналіз файлової активності в режимі, наближеному до реального часу. Реалізований набір інтерактивних візуалізацій дозволяє здійснювати комплексний аналіз поведінки користувачів, швидко локалізувати джерело потенційного інциденту та виявляти характерні ознаки кіберзагроз, зокрема масового шифрування файлів програмами-вимагачами та аномальної активності внутрішніх користувачів.

Порівняльний аналіз запропонованої архітектури зі штатними засобами аудиту Windows Server показав, що реалізований комплекс забезпечує централізоване накопичення подій, суттєво покращує можливості їх оперативного аналізу та скорочує час виявлення потенційних інцидентів інформаційної безпеки. Поєднання автоматизованого збору телеметрії, захищеної передачі даних, централізованого сховища та інтерактивної аналітики формує єдиний технологічний комплекс, придатний для використання як інструмент підтримки процесів моніторингу інформаційної безпеки підприємства.

Практична цінність роботи полягає у створенні повністю працездатного програмно-апаратного рішення, яке може бути адаптоване до використання в корпора-

тивних інформаційних системах без необхідності внесення змін до штатної архітектури Windows Server. Запропонований комплекс забезпечує автоматизацію процесів збору та аналізу файлової активності, підвищує оперативність реагування на потенційні загрози та може бути використаний як основа для подальшого розвитку систем моніторингу інформаційної безпеки.

Таким чином, усі поставлені на початку дослідження завдання виконано в повному обсязі, а поставлену мету досягнуто. Розроблений програмний комплекс продемонстрував високу стабільність, масштабованість, відмовостійкість та практичну ефективність, підтвердивши доцільність використання запропонованої архітектури для побудови сучасних систем моніторингу файлової активності та підтримки процесів забезпечення інформаційної безпеки корпоративних мереж.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Буров Є. В. Комп’ютерні мережі : підручник. Львів : Бакушевич, 2011. 262 с.
2. Вовк О. Б., Вовк Ю. Б. Організація баз даних та інформаційних систем : навч. посіб. Львів : Видавництво Львівської політехніки, 2020. 232 с.
3. Глибовець М. М., Олецький О. В. Основи системного програмування : підручник. Київ : НаУКМА, 2015. 290 с.
4. Кулаков Ю. О., Луцький Г. М. Комп'ютерні мережі : підручник / за ред. Ю. О. Кулакова. Київ : Юніор, 2013. 400 с.
5. Мельник А. О. Архітектура комп'ютера : підручник. Львів : Видавництво Львівської політехніки, 2018. 464 с.
6. Пасічник В. В., Резніченко В. А. Організація баз даних та знань : підручник. Київ : Видавнича група BHV, 2018. 448 с.
7. Руденко В. Д. Комп’ютерні мережі та мережеві технології : навч. посіб. Харків : ХНУРЕ, 2021. 312 с.
8. Шеховцов В. А. Операційні системи : підручник. Київ : Видавнича група BHV, 2015. 576 с.
9. Donenfeld J. A. WireGuard: Next Generation Kernel Network Tunnel. *Proceedings of the Network and Distributed System Security Symposium (NDSS)*. San Diego, California, 2017. URL: https://www.wireguard.com/papers/wireguard.pdf (date of access: 14.05.2026).
10. Ferrari A., Russo M. Introducing Microsoft Power BI. Redmond : Microsoft Press, 2016. 188 p.
11. Microsoft Corporation. Windows Server 2022 Administration Fundamentals / Bekim Dauti. Birmingham : Packt Publishing, 2022. 384 p.
12. Powell B. Microsoft Power BI Performance Tuning: Optimize queries and visualization models in DirectQuery mode. Birmingham : Packt Publishing, 2022. 294 p.

13. Russinovich M., Margosis D. Windows Internals, Part 1: System architecture, processes, threads, memory management, and more. 7th ed. Redmond : Microsoft Press, 2017. 800 p.
14. Stanek W. R. Windows Server 2019 Inside Out. Redmond : Microsoft Press, 2019. 736 p.
15. Tyson M. Windows PowerShell Scripting Guide. Shelter Island : Manning Publications, 2018. 425 p.
16. Tailscale Corporate Networking Overview and Core Technology Architecture. *Tailscale Documentation*. 2024. URL: https://tailscale.com/blog/how-tailscale-works/ (date of access: 18.05.2026).
17. FileSystemWatcher Class Component Architecture in .NET Ecosystem. *Microsoft Learn .NET API Reference*. 2025. URL: https://learn.microsoft.com/en-us/dotnet/api/system.io.filesystemwatcher (date of access: 20.05.2026).
18. SQL Server Storage Engine Architecture and Relational Query Processing. *Microsoft Learn SQL Server*. 2024. URL: https://learn.microsoft.com/en-us/sql/relational-databases/ (date of access: 22.05.2026).
19. DirectQuery models in Microsoft Power BI Desktop. *Microsoft Learn Power BI*. 2025. URL: https://learn.microsoft.com/en-us/power-bi/connect-data/desktop-directquery-about (date of access: 02.06.2026).
20. I/O Performance Tuning Guide for Windows Server Environments. *Microsoft Learn Infrastructure Documentation*. 2024. URL: https://learn.microsoft.com/en-us/windows-server/administration/performance-tuning/subsystems/ (date of access: 11.06.2026).

ДОДАТКИ

Додаток А*

Лістинг програмного коду фонового сценарію моніторингу файлової системи (monitor.ps1)

```
# ====================================
# FILE MONITOR SYSTEM
# REALTIME VERSION (CORRECT DATA FILLING)
# ====================================

$WatchPath = "F:\TestStorage"
$LogFile = "F:\Scripts\FileMonitor.log"

$SqlConnectionString = "Server=100.78.32.105,1433;Database=FileResourceStatis-
tics;User ID=sa;Password=DiplomServer2026;TrustServerCertificate=True;"

# ====================================
# LOG FUNCTION
# ====================================

function Write-Log {
    param([string]$Message)
    $Time = Get-Date -Format "yyyy-MM-dd HH:mm:ss"
    $Line = "[$Time] $Message"
    Write-Host $Line
    Add-Content -Path $LogFile -Value $Line
}

# ====================================
# SQL EXECUTION FUNCTIONS
# ====================================

function Execute-Sql {
    param(
        [string]$Query,
        [hashtable]$Params
    )
    try {
        $Connection = New-Object System.Data.SqlClient.SqlConnection
        $Connection.ConnectionString = $SqlConnectionString
        $Connection.Open()

        $Command = $Connection.CreateCommand()
        $Command.CommandText = $Query

        if ($Params) {
            foreach ($Key in $Params.Keys) {
```

```
            $null = $Command.Parameters.AddWithValue("@$Key", $Params[$Key])
        }
    }

    $Command.ExecuteNonQuery() | Out-Null
    $Connection.Close()
  }
  catch {
    Write-Log "SQL ERROR: $($_.Exception.Message)"
  }
}

function Execute-SqlScalar {
  param(
    [string]$Query,
    [hashtable]$Params
  )
  try {
    $Connection = New-Object System.Data.SqlClient.SqlConnection
    $Connection.ConnectionString = $SqlConnectionString
    $Connection.Open()

    $Command = $Connection.CreateCommand()
    $Command.CommandText = $Query

    if ($Params) {
        foreach ($Key in $Params.Keys) {
            $null = $Command.Parameters.AddWithValue("@$Key", $Params[$Key])
        }
    }

    $Result = $Command.ExecuteScalar()
    $Connection.Close()
    return $Result
  }
  catch {
    Write-Log "SQL SCALAR ERROR: $($_.Exception.Message)"
    return $null
  }
}

# ======================================
# GET SYSTEM IDs
# ======================================
```

```
function Get-ServerId {
   $MachineName = $env:COMPUTERNAME
   $Connection = New-Object System.Data.SqlClient.SqlConnection
   $Connection.ConnectionString = $SqlConnectionString
   $Connection.Open()

   $Query = "SELECT TOP 1 ServerId FROM Servers WHERE MachineName=@Ma-
chineName"
   $Command = $Connection.CreateCommand()
   $Command.CommandText = $Query
   $null = $Command.Parameters.AddWithValue("@MachineName", $MachineName)
   $Result = $Command.ExecuteScalar()

   if (-not $Result) {
      $Insert = "INSERT INTO Servers (MachineName, ClientIP, DomainName, Last-
Seen) OUTPUT INSERTED.ServerId VALUES (@MachineName, '', '', GETDATE())"
      $Command = $Connection.CreateCommand()
      $Command.CommandText = $Insert
      $null = $Command.Parameters.AddWithValue("@MachineName", $Machine-
Name)
      $Result = $Command.ExecuteScalar()
   }
   $Connection.Close()
   return $Result
}

function Get-UserId {
   $UserName = $env:USERNAME
   $Connection = New-Object System.Data.SqlClient.SqlConnection
   $Connection.ConnectionString = $SqlConnectionString
   $Connection.Open()

   $Query = "SELECT TOP 1 UserId FROM UsersAndGroups WHERE
UserName=@UserName"
   $Command = $Connection.CreateCommand()
   $Command.CommandText = $Query
   $null = $Command.Parameters.AddWithValue("@UserName", $UserName)
   $Result = $Command.ExecuteScalar()

   if (-not $Result) {
      $Insert = "INSERT INTO UsersAndGroups (UserName, IsGroup, LastSeen) OUT-
PUT INSERTED.UserId VALUES (@UserName, 0, GETDATE())"
      $Command = $Connection.CreateCommand()
      $Command.CommandText = $Insert
      $null = $Command.Parameters.AddWithValue("@UserName", $UserName)
```

```
        $Result = $Command.ExecuteScalar()
    }
    $Connection.Close()
    return $Result
}

function Get-OperationId {
    param([string]$OperationName)
    $Connection = New-Object System.Data.SqlClient.SqlConnection
    $Connection.ConnectionString = $SqlConnectionString
    $Connection.Open()

    $Query = "SELECT TOP 1 AccessTypeId FROM AccessType WHERE Operation-
Name=@OperationName"
    $Command = $Connection.CreateCommand()
    $Command.CommandText = $Query
    $null = $Command.Parameters.AddWithValue("@OperationName", $Operation-
Name)
    $Result = $Command.ExecuteScalar()

    if (-not $Result) {
        $Insert = "INSERT INTO AccessType (OperationName, Description) OUTPUT
INSERTED.AccessTypeId VALUES (@OperationName, @Description)"
        $Command = $Connection.CreateCommand()
        $Command.CommandText = $Insert
        $null = $Command.Parameters.AddWithValue("@OperationName", $Operation-
Name)
        $null = $Command.Parameters.AddWithValue("@Description", $OperationName)
        $Result = $Command.ExecuteScalar()
    }
    $Connection.Close()
    return $Result
}

# КЕШУВАННЯ ГЛОБАЛЬНИХ ID
$Global:ServerId = Get-ServerId
$Global:UserId = Get-UserId
$Global:InitialScanOpId = Get-OperationId "InitialScan"

# ====================================
# INSERT EVENT FUNCTION
# ====================================

function Insert-Event {
    param(
```

```
        [string]$FullItemPath,
        [string]$Operation,
        [string]$OldFilePath = ""
    )

    try {
        # Визначаємо OperationId та специфічні для події поля
        if ($Operation -eq "InitialScan") {
            $OperationId      = $Global:InitialScanOpId
            $EventId          = 5000
            $ProcessName      = "FILE LOCKED"
            $ResultField      = "File detected during baseline initial scanning process"
            $RawMessage       = "FSW Direct Engine"
            $ActionDescription = "Event registered via internal low level watcher subsys-
tem"
        } else {
            $OperationId      = Get-OperationId $Operation
            $EventId          = 4663
            $ProcessName      = "powershell"
            $ResultField      = ""
            $RawMessage       = ""
            $ActionDescription = $Operation
        }

        # Розбиваємо шлях на Prefix (F:\TestStorage) та відносний шлях файлу
        $FilePathPrefix = $WatchPath
        $RelativeFilePath = $FullItemPath
        if ($FullItemPath.StartsWith($WatchPath)) {
            # Отримуємо чистий відносний шлях (ВИПРАВЛЕНО: [char[]] замість
[char])
            $RelativeFilePath = $FullItemPath.Substring($WatchPath.Length).Trim-
Start([char[]]' \')
        }

        # Отримання інформації про файл
        $Item = Get-Item $FullItemPath -ErrorAction SilentlyContinue
        if ($Item) {
            $FileSize    = $Item.Length
            $IsDirectory = [int]$Item.PSIsContainer
            $Extension   = $Item.Extension
        } else {
            $FileSize    = 0
            $IsDirectory = 0
            $Extension   = ""
        }
```

```
        $Query = @"
INSERT INTO FileAccessesNormalized
(
    ServerId, UserId, OperationId, EventTime, EventId,
    FilePath, FilePathPrefix, AccessMask, ProcessName, Result,
    RawMessage, ActionDescription, OldFilePath, FileSizeBytes, IsDirectory, FileExten-
sion
)
VALUES
(
    @ServerId, @UserId, @OperationId, GETDATE(), @EventId,
    @FilePath, @FilePathPrefix, '', @ProcessName, @Result,
    @RawMessage, @ActionDescription, @OldFilePath, @FileSizeBytes, @IsDirec-
tory, @FileExtension
)
"@

        $Params = @{
            ServerId          = $Global:ServerId
            UserId            = $Global:UserId
            OperationId       = $OperationId
            EventId           = $EventId
            FilePath          = $RelativeFilePath
            FilePathPrefix    = $FilePathPrefix
            ProcessName       = $ProcessName
            Result            = $ResultField
            RawMessage        = $RawMessage
            ActionDescription = $ActionDescription
            OldFilePath       = $OldFilePath
            FileSizeBytes     = $FileSize
            IsDirectory       = $IsDirectory
            FileExtension     = $Extension
        }

        Execute-Sql -Query $Query -Params $Params
        Write-Log "$Operation => $RelativeFilePath"
    }
    catch {
        Write-Log "INSERT ERROR: $($_.Exception.Message)"
    }
}

# ======================================
# INITIAL SCAN
```

```
# ====================================

function Initial-Scan {
   Write-Log "INITIAL SCAN STARTED"

   # Перевіряємо по відносному шляху (FilePath), щоб точно збігалося з базою
   $CheckQuery = @"
SELECT TOP 1 1
FROM FileAccessesNormalized
WHERE FilePath = @FilePath
  AND OperationId = @OperationId
  AND ServerId = @ServerId
"@

   Get-ChildItem $WatchPath -Recurse -Force -ErrorAction SilentlyContinue | Fo-
rEach-Object {
      $FullItemPath = $_.FullName

      # Вираховуємо відносний шлях для точної перевірки в базі
      $RelativeFilePath = $FullItemPath
      if ($FullItemPath.StartsWith($WatchPath)) {
         # ВИПРАВЛЕНО: [char[]] замість [char]
         $RelativeFilePath = $FullItemPath.Substring($WatchPath.Length).Trim-
Start([char[]]' \')
      }

      $CheckParams = @{
         FilePath    = $RelativeFilePath
         OperationId = $Global:InitialScanOpId
         ServerId    = $Global:ServerId
      }

      $Exists = Execute-SqlScalar -Query $CheckQuery -Params $CheckParams

      if (-not $Exists) {
         Insert-Event -FullItemPath $FullItemPath -Operation "InitialScan"
      }
   }

   Write-Log "INITIAL SCAN FINISHED"
}

# ====================================
# START MONITORING
# ====================================
```

```
Write-Host ""
Write-Host "======================================"
Write-Host "FILE MONITOR STARTING"
Write-Host "======================================"
Write-Host ""

Initial-Scan

# ======================================
# WATCHER
# ======================================

$Watcher = New-Object System.IO.FileSystemWatcher
$Watcher.Path = $WatchPath
$Watcher.IncludeSubdirectories = $true
$Watcher.EnableRaisingEvents = $true

Write-Host ""
Write-Host "Watcher active: $($Watcher.EnableRaisingEvents)"
Write-Host "Monitoring path: $WatchPath"
Write-Host ""
Write-Host "REALTIME MONITORING ACTIVE"
Write-Host ""

Write-Log "LIVE MONITORING STARTED"

while ($true)
{
   try
   {
      $Result = $Watcher.WaitForChanged([System.IO.WatcherChangeTypes]::All,
1000)

      if ($Result.TimedOut -eq $false)
      {
         $ChangeType = $Result.ChangeType.ToString()
         $FullPath = Join-Path $WatchPath $Result.Name
         $OldPath = ""

         if ($ChangeType -eq "Renamed")
         {
            $OldPath = $Result.OldName
         }
```

```
        # Для реального часу передаємо повний шлях, функція сама розбере його
на префікс та відносний
        Insert-Event `
            -FullItemPath $FullPath `
            -OldFilePath $OldPath `
            -Operation $ChangeType
      }
   }
   catch
   {
      Write-Log "WATCHER ERROR: $($_.Exception.Message)"
   }
}
```

ДОДАТОК Б

Лістинг програмного коду сценарію-генератора імітаційного навантаження (`generator.ps1`)

```
# ==========================================================
# FILE ACTIVITY GENERATOR
# ==========================================================

Clear-Host

Write-Host "=====================================" -ForegroundColor
Cyan
Write-Host " FILE ACTIVITY GENERATOR " -ForegroundColor Cyan
Write-Host "=====================================" -ForegroundColor
Cyan
Write-Host ""

$BasePath = "F:\TestStorage"

$EventCount = Read-Host "Enter number of events"

if (-not ($EventCount -as [int])) {
    Write-Host "Invalid number" -ForegroundColor Red
    exit
}

$EventCount = [int]$EventCount

Write-Host ""
Write-Host "Generating $EventCount events..." -ForegroundColor Green
Write-Host ""

# ----------------------------------------------------------
# EXTENSIONS
# ----------------------------------------------------------

$Extensions = @(
    ".txt",
    ".docx",
    ".pptx",
    ".xlsx",
    ".pdf",
    ".png",
    ".rtf"
```

```
)

# -------------------------------------------------------
# NAMES
# -------------------------------------------------------

$Names = @(
    "report",
    "document",
    "presentation",
    "notes",
    "backup",
    "invoice",
    "statistics",
    "project",
    "diagram",
    "analysis"
)

# -------------------------------------------------------
# CREATE FILE
# -------------------------------------------------------

function New-RandomFile {

    $Name = $Names | Get-Random
    $Ext = $Extensions | Get-Random

    $RandomNumber = Get-Random -Minimum 1000 -Maximum 9999

    $FileName = "$Name-$RandomNumber$Ext"

    $Path = Join-Path $BasePath $FileName

    $Size = Get-Random -Minimum 20 -Maximum 200

    $Content = "TEST DATA " * $Size

    Set-Content -Path $Path -Value $Content

    return $Path
}

# -------------------------------------------------------
# MODIFY FILE
```

```
# -------------------------------------------------------

function Edit-RandomFile {

    $Files = Get-ChildItem $BasePath -File -Recurse -ErrorAction SilentlyContinue

    if ($Files.Count -eq 0) {
        return
    }

    $File = $Files | Get-Random

    Add-Content -Path $File.FullName -Value "`nMODIFIED $(Get-Date)"
}

# -------------------------------------------------------
# RENAME FILE
# -------------------------------------------------------

function Rename-RandomFile {

    $Files = Get-ChildItem $BasePath -File -Recurse -ErrorAction SilentlyContinue

    if ($Files.Count -eq 0) {
        return
    }

    $File = $Files | Get-Random

    $NewName = "renamed-" + (Get-Random -Minimum 1000 -Maximum 9999) +
$File.Extension

    Rename-Item -Path $File.FullName -NewName $NewName
}

# -------------------------------------------------------
# DELETE FILE
# -------------------------------------------------------

function Remove-RandomFile {

    $Files = Get-ChildItem $BasePath -File -Recurse -ErrorAction SilentlyContinue

    if ($Files.Count -eq 0) {
        return
```

```
    }

    $File = $Files | Get-Random

    Remove-Item $File.FullName -Force
}

# -------------------------------------------------------
# COPY FILE
# -------------------------------------------------------

function Copy-RandomFile {

    $Files = Get-ChildItem $BasePath -File -Recurse -ErrorAction SilentlyContinue

    if ($Files.Count -eq 0) {
        return
    }

    $File = $Files | Get-Random

    $CopyName = "copy-" + (Get-Random -Minimum 1000 -Maximum 9999) +
$File.Extension

    $Destination = Join-Path $BasePath $CopyName

    Copy-Item $File.FullName $Destination
}

# -------------------------------------------------------
# MAIN LOOP
# -------------------------------------------------------

for ($i = 1; $i -le $EventCount; $i++) {

    $Action = Get-Random -Minimum 1 -Maximum 101

    try {

        if ($Action -le 40) {

            Edit-RandomFile

            Write-Host "[$i/$EventCount] FILE MODIFIED" -ForegroundColor Yellow
        }
```

```
        elseif ($Action -le 65) {

            $Path = New-RandomFile

            Write-Host "[$i/$EventCount] FILE CREATED -> $Path" -ForegroundColor
Green
        }

        elseif ($Action -le 80) {

            Copy-RandomFile

            Write-Host "[$i/$EventCount] FILE COPIED" -ForegroundColor Cyan
        }

        elseif ($Action -le 90) {

            Rename-RandomFile

            Write-Host "[$i/$EventCount] FILE RENAMED" -ForegroundColor Magenta
        }

        else {

            Remove-RandomFile

            Write-Host "[$i/$EventCount] FILE DELETED" -ForegroundColor Red
        }

    }
    catch {

        Write-Host "ERROR: $($_.Exception.Message)" -ForegroundColor Red
    }

    Start-Sleep -Milliseconds (Get-Random -Minimum 700 -Maximum 3000)
}

Write-Host ""
Write-Host "=====================================" -ForegroundColor
Green
Write-Host " GENERATION FINISHED "
Write-Host "=====================================" -ForegroundColor
Green
```

ДОДАТОК В

Структура бази даних та SQL-скрипт ініціалізації реляційних таблиць сховища (SQLQuery1.sql)

```
USE master;
GO

IF EXISTS (SELECT * FROM sys.databases WHERE name = 'FileResourceStatistics')
BEGIN
  ALTER DATABASE FileResourceStatistics SET SINGLE_USER WITH ROLLBACK IMMEDIATE;
  DROP DATABASE FileResourceStatistics;
END
GO

CREATE DATABASE FileResourceStatistics;
GO

USE FileResourceStatistics;
GO

-----------------------------------------------
-- ACCESS TYPE
-----------------------------------------------
CREATE TABLE AccessType (
  AccessTypeId INT IDENTITY(1,1) PRIMARY KEY,
  OperationName NVARCHAR(255) UNIQUE,
  Description NVARCHAR(MAX)
);
GO
```

```
-----------------------------------------------
-- SERVERS
-----------------------------------------------
CREATE TABLE Servers (
   ServerId INT IDENTITY(1,1) PRIMARY KEY,
   MachineName NVARCHAR(255),
   ClientIP NVARCHAR(50),
   DomainName NVARCHAR(255),
   LastSeen DATETIME DEFAULT GETDATE()
);
GO

-----------------------------------------------
-- USERS
-----------------------------------------------
CREATE TABLE UsersAndGroups (
   UserId INT IDENTITY(1,1) PRIMARY KEY,
   UserName NVARCHAR(255),
   IsGroup BIT DEFAULT 0,
   LastSeen DATETIME DEFAULT GETDATE()
);
GO

-----------------------------------------------
-- MAIN LOG TABLE
-----------------------------------------------
CREATE TABLE FileAccessesNormalized (
   Id BIGINT IDENTITY(1,1) PRIMARY KEY,
```

```
ServerId INT,
UserId INT,
OperationId INT,

EventTime DATETIME,
EventId INT,

FilePath NVARCHAR(MAX),
FilePathPrefix NVARCHAR(MAX),

AccessMask NVARCHAR(50),
ProcessName NVARCHAR(255),
Result NVARCHAR(100),

RawMessage NVARCHAR(MAX),

ActionDescription NVARCHAR(500),

OldFilePath NVARCHAR(MAX),

FileSizeBytes BIGINT,

IsDirectory BIT DEFAULT 0,

FileExtension NVARCHAR(50),

CONSTRAINT FK_FileAccesses_Servers
   FOREIGN KEY (ServerId)
   REFERENCES Servers(ServerId),
```

```
    CONSTRAINT FK_FileAccesses_Users
        FOREIGN KEY (UserId)
        REFERENCES UsersAndGroups(UserId),

    CONSTRAINT FK_FileAccesses_AccessType
        FOREIGN KEY (OperationId)
        REFERENCES AccessType(AccessTypeId)
);
GO
```