Хмельницький національний університет
Факультет інформаційних технологій
Кафедра кібербезпеки

КВАЛІФІКАЦІЙНА РОБОТА

Кушніра Давида Володимировича

на здобуття ступеня вищої освіти Бакалавра

Система протидії витокам даних на підприємстві з використанням DLP

Галузь знань 12 – Інформаційні технології
Спеціальність 125 – Кібербезпека
Освітня програма Кібербезпека

Шифр КРБКБ.2102155.21.02.33 ПЗ

Виконав студент 4 курсу група КБ-21-2 Давид КУШНІР
Керівник канд. техн. наук, доцент Ігор МУЛЯР
Нормоконтролер старший викладач Сергій МОСТОВИЙ

До захисту допускаю:
Завідувач кафедри кібербезпеки Юрій КЛЬОЦ

09 06 2025 р.

Хмельницький 2025

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет Інформаційних технологій

Кафедра Кібербезпеки

Рівень вищої освіти Бакалавр

Галузь знань 12 – Інформаційні технології

Спеціальність 125 – Кібербезпека

Освітня програма Кібербезпека

ЗАТВЕРДЖУЮ

Завідувач кафедри кібербезпеки

Юрій КЛЬОЦ

15 лютого 2025 р.

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Кушніру Давиду Володимировичу

1 Тема роботи Система протидії витокам даних на підприємстві з використанням DLP мережі.

Керівник роботи к.т.н. доцент Ігор Муляр

Затверджено наказом ректора університету від 7 лютого 2025 № 23

2 Строк подання студентом кваліфікаційної роботи на кафедру 04.06.2025

3 Вихідні дані до роботи Проаналізувати предметну область та існуючі рішення в галузі захисту чутлтвої інформації. Сформулювати постановку задачі та визначити функціональні вимоги до системи. Розробити архітектуру та загальну структуру системи. Обґрунтувати вибір інструментів та технологій для реалізації. Реалізувати модулі для сканування хостів і портів. Розробити серверну частину додатку з використанням обраного стеку технологій. Реалізувати клієнтську частину. Провести тестування функціональності системи.

4 Зміст пояснювальної записки (перелік питань, які потрібно розробити)

Вступ. Аналіз предметної області. Огляд існуючих рішень. Постановка задачі. Функціональні вимоги до системи. Архітектура та загальна структура системи Вибір засобів та технологій для реалізації. Програмна реалізація системи та її. тестування

5 Перелік графічного матеріалу (із зазначенням обов'язкових креслень)

Архітектура системи. Алгоритм роботи модуля лексичного аналізу Алгоритм роботи серверної частини

6 Консультанти розділів кваліфікаційної роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7 Дата видачі завдання 16 лютого 2025 р.

КАЛЕНДАРНИЙ ПЛАН

Назва етапів (розділів) кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
Вибір і затвердження теми кваліфікаційної роботи	Січень-Лютий	
Ознайомлення з предметною областю	Лютий	
Дослідження існуючих рішень	Лютий	
Постановка задачі	Березень	
Визначення загальних принципів рішення задачі	Березень	
Деталізація принципів рішення задачі	Квітень	
Розробка проєктних рішень	Квітень	
Апробація проєктних рішень	Травень	
Оформлення пояснювальної записки згідно вимог	Травень	
Оформлення графічної частини	Червень	
Захист КР	Червень	

Студент _____ Давид КУШНІР

Керівник кваліфікаційної роботи _____ Ігор МУЛЯР

АНОТАЦІЯ

Тема кваліфікаційної роботи: Система протидії витокам даних на підприємстві з використанням DLP.

Автор роботи: Кушнір Давид Вікторович.

Керівник роботи: Муляр Ігор Володимирович.

Пояснювальна записка: 69 с., 2 додатки, 25 рисунків, 44 джерела.

Графічна частина: 3 плакати.

ЗАХИСТ ІНФОРМАЦІЇ, КОФІДЕНЦІЙНА ІФОРМАЦІЯ, СИСТЕМА ЗАПОБІГАННЯ ВИТОКІВ ІНФОРМАЦІЇ

Кваліфікаційна робота бакалавра присвячена розробці система DLP для підприємств.

Мета роботи полягає у дослідженні та систематизації основних загроз інформаційній безпеці, спричинених несанкціонованим розголошенням конфіденційних відомостей, а також у створенні комплексної системи запобігання витокам даних, здатної здійснювати моніторинг чутливої інформації в корпоративному інформаційному середовищі та виявляти порушення з боку внутрішніх користувачів системи.

В межах даної кваліфікаційної роботи було спроектовано та реалізовано спеціалізовану систему протидії витокам конфіденційних даних, яка забезпечує комплексний захист корпоративної інформації від внутрішніх загроз.

07.06.2025 _____

ABSTRACT

Subject of qualification work: Enterprise data leakage prevention system using DLP

Author: Kushnir Davyd Volodymyrovych.

Head of work: Mulyar Ihor Volodymyrovych.

Explanatory note: 69 p., 2 appendices, 25 figures, 44 sources

Graphic part: 3 posters

INFORMATION PROTECTION, CONFIDENTIAL INFORMATION, INFORMATION LEAKAGE PREVENTION SYSTEM

The bachelor's thesis is devoted to the development of a DLP system for enterprises.

The purpose of the work is to study and systematize the main threats to information security caused by unauthorized disclosure of confidential information, as well as to create a comprehensive data leakage prevention system capable of monitoring sensitive information in the corporate information environment and detecting violations by internal users of the system.

Within the framework of this qualification work, a specialized system for counteracting confidential data leaks was designed and implemented, which provides comprehensive protection of corporate information from internal threats.

07.06.2025

ЗМІСТ

ВСТУП

У сучасному світі організації дедалі частіше застосовують технології для обробки та зберігання значних обсягів даних. У зв’язку з цим питання безпеки інформації набуває критичного значення. Кіберзагрози постійно еволюціонують, охоплюючи хакерські атаки, внутрішні порушення, технічні збої та людські помилки. Витік конфіденційних даних може спричинити значні наслідки: втрату клієнтської довіри, фінансові збитки та юридичні труднощі.

Особливого значення захист даних набуває через потребу відповідати законодавчим вимогам у сфері конфіденційності. Організації зобов’язані суворо дотримуватися правил захисту персональних даних клієнтів, що допомагає уникати правових ризиків і забезпечує відповідність стандартам.

Зі зростанням технологічних можливостей і збільшенням числа підключених пристроїв у бізнес-середовищі посилюється потреба в інтегрованих стратегіях кіберзахисту. Такі системи повинні не лише своєчасно виявляти загрози, але й оперативно реагувати на них. Цілісний підхід до захисту стає ключовим для підтримання репутації компанії, її стабільності на ринку та довіри з боку клієнтів.

Збереження конфіденційності інформації має виняткове значення для захисту бізнесу. Наприклад, керівники компаній прагнуть уникнути ситуацій, коли їхні конкуренти отримують доступ до даних про нові розробки або стратегічні плани розвитку.

Порушення конфіденційності часто відбувається через витік інформації – ситуацію, коли дані стають доступними для осіб, які не мають на це права чи повноважень. Значною мірою причиною таких витоків можуть бути співробітники, які мають доступ до внутрішньої мережі організації. Це може бути працівник, якого мотивували конкуренти, або особа, яка відчуває образу через конфлікт чи несправедливість.

Дії внутрішніх зловмисників нерідко завдають організації більше шкоди, ніж атаки зовнішніх загроз. Їхні дії можуть не тільки спричинити значні фінансові втрати, але й підривати репутацію компанії, що має довготривалі наслідки. Зважаючи на це, ефективний захист від внутрішніх ризиків є не менш важливим, ніж протидія зовнішнім атакам.

Інформація про кредитну картку та ідентифікаційний номер клієнтів є прикладами конфіденційної інформації, яка є дуже поширеною у корпоративному світі, це інформація, яка має бути обмеженою і не повинна пропускатись або поширюватися. Ситуація, коли працівник намагається надіслати таку інформацію конкуруючим компаніям, є прикладом витоку, який може мати руйнівні наслідки для компанії, наприклад, втрата клієнтів, призведення до судових справ та ін. Щоб запобігти подібним витокам, системи, призначені для виявлення та захисту конфіденційних даних, були запроваджені в 2006 році і стали називатись Data Leakage (Loss) Prevention (DLP) – запобігання витоків інформації [1]. Їх мета полягає в тому, щоб закупорювати існуючі точки витоку і блокувати несанкціоновані дії з конфіденційними даними.

Сьогодні існує багато вендорів з різними системами DLP для різних операційних систем і мобільних платформ. Але обмеження доступу працівників до повсякденної роботи іноді може викликати роздратування і фрустрацію. Це, у свою чергу, може призвести до того, що працівники намагатимуться знайти способи обійти систему.

Головна мета кваліфікаційної роботи полягає у дослідженні та аналізі основних викликів, пов’язаних із захистом інформації, зокрема проблем, що виникають через витік конфіденційних даних. Також передбачено створення ефективної системи, здатної забезпечувати контроль за переміщенням чутливої інформації всередині корпоративного інформаційного простору. Така система повинна своєчасно виявляти несанкціоновані дії користувачів, пов’язані з обробкою даних, і запобігати можливим загрозам.

1 ДОСЛІДЖЕННЯ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Аналіз загроз інформаційної безпеки

У сфері інформаційних технологій дані являють собою ключовий ресурс, який можна записувати, зберігати, обробляти та передавати за допомогою цифрових систем. Вони можуть існувати у різних форматах: текстових, числових, графічних, аудіовізуальних чи структурованих у базах даних.

Рівень чутливості та значущості даних залежить від їхнього характеру. Наприклад, персональні дані, такі як імена, адреси або фінансові реквізити, потребують високого рівня захисту через їхню конфіденційність. Водночас дані про технічний стан системи чи результати операцій можуть мати нижчий ступінь ризику та потребувати менш суворих заходів захисту.

Аналіз і обробка інформації стали основою багатьох сучасних технологічних напрямів, зокрема машинного навчання, штучного інтелекту та аналітики великих обсягів даних. Важливо забезпечувати надійний захист інформації від несанкціонованого доступу, втрати чи змін, адже це критично для збереження її цінності.

Три базові принципи безпеки даних – цілісність, доступність і конфіденційність – слугують фундаментом захисту інформації від загроз. Ця концепція є основою для формування політик і рішень, спрямованих на мінімізацію ризиків у цифровому середовищі [2].

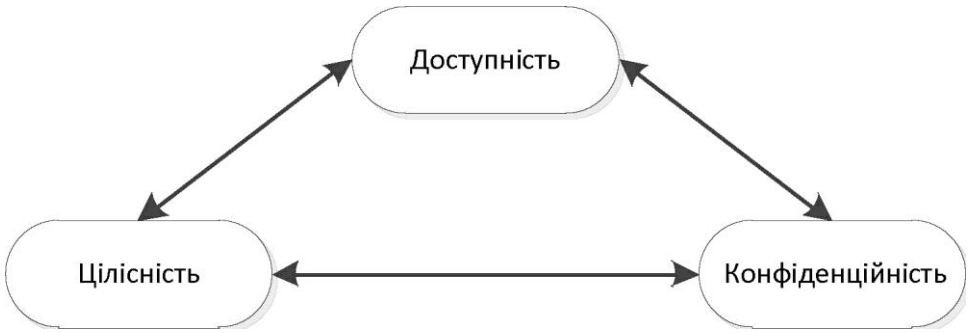


Рисунок 1.1 – Тріада інформаційної безпеки

Конфіденційність означає стан та принцип збереження інформації, коли доступ до неї надається виключно визначеним особам або групам, які отримали такі повноваження. Ці особи повинні використовувати інформацію відповідно до чітко встановлених правил, процедур і політик, що регламентують її захист [3, 4].

Для забезпечення конфіденційності інформації організації впроваджують різноманітні механізми контролю доступу, використання шифрування та надання персоналізованих прав доступу. Наприклад, фінансові установи використовують системи багатофакторної аутентифікації, включаючи двоетапну перевірку, щоб захистити дані від несанкціонованого проникнення.

Банки та подібні організації дотримуються суворих політик конфіденційності, які визначають, хто має право на доступ до клієнтської інформації та в яких межах ці дані можуть бути застосовані. Такі підходи спрямовані на мінімізацію ризиків витоку даних та збереження довіри клієнтів.

За останні роки проблема витоків конфіденційної інформації стала настільки поширеною, що фактично перетворилася на щоденну реальність для багатьох комерційних і некомерційних організацій. Наприклад, у березні 2019 року компанія Tesla подала два судових позови проти п’ятьох колишніх співробітників, звинувачуючи їх у передачі секретної інформації конкурентам, як повідомляє портал TechCrunch [5]. У штаб-квартирі Європолу в Гаазі зникли секретні документи, що містили особисту інформацію вищих посадових осіб агентства. Деякі з цих файлів були знайдені в громадському місці та повернуті до місцевої поліції. Цей інцидент викликав серйозну кризу в поліцейському управлінні ЄС [6]. У німецькому Бундесвері стався витік інформації про таємні онлайн-наради, проведені через систему відеоконференцій Webex. Дані про приблизно 6000 конференцій, включаючи розклад, тривалість, теми та імена організаторів, були доступні у відкритому доступі в інтернеті. Деякі наради були позначені як секретні, що підкреслює серйозність цього інциденту [7].

Клас інформації безпосередньо впливає як на масштаби потенційної шкоди у випадку витоку, так і на вимоги до системи захисту, яку повинна забезпечити

організація, що працює з такими даними [8]. Чим вищий рівень чутливості інформації, тим суворішими повинні бути заходи для її захисту (рис.1.2).

Рисунок 1.2 – Класифікація видів інформації [9]

За результатами останніх досліджень встановлено, що значна частина інцидентів, пов’язаних із витоками конфіденційної інформації, здійснюється зсередини організацій. Ці злочини нерідко вчиняють співробітники компаній, які мають доступ до внутрішніх систем і даних. Такі дії можуть бути обумовлені особистими мотивами, наприклад, незадоволеністю умовами праці, або ж підкупом з боку конкурентів.

Це підкреслює необхідність не лише технічних, але й організаційних заходів захисту, зокрема регулярного аудиту доступу, моніторингу активностей користувачів та впровадження політик управління ризиками.

Такі інциденти доводять критичність загрози витоку даних і несанкціонованого доступу до інформації, що входять до числа найважливіших проблем сучасності.

Системи протидії витоку даних є важливим інструментом для організацій, що прагнуть зберегти конфіденційну та чутливу інформацію в межах корпоративної мережі. Це особливо актуально для захисту даних від внутрішніх загроз, коли працівники, маючи авторизований доступ до інформації, можуть сприяти її витоку. Для компаній збереження таких даних є не лише правом, але й необхідністю, оскільки вони часто відіграють ключову роль у підтримці операційної діяльності. Водночас втрата критичних даних може стати катастрофічною, поставивши під загрозу існування компанії. Проте впровадження технологічних рішень не завжди є достатнім. Організації мають також аналізувати фактори, які провокують внутрішніх співробітників на порушення безпеки.

Серед основних причин подібних інцидентів – бажання швидкого збагачення, яке часто стає мотивом для працівників, переманених конкурентами. Вони можуть передавати інформацію про дослідження, технології чи розробки, які становлять стратегічну цінність. Такі дії спрямовані на отримання конкурентної переваги, яка є вагомим чинником у боротьбі не лише між компаніями, а й у глобальному протистоянні між державами.

Іншим поширеним мотивом є негативне ставлення працівника до компанії. Це може бути спричинено низьким рівнем заробітної плати, тиском з боку керівництва, шантажем чи конфліктами на політичному ґрунті. Така поведінка іноді класифікується як кібертероризм, адже наслідки можуть мати масштабний вплив на діяльність компанії.

Неправомірне копіювання даних, що містять захищену політикою безпеки інформацію, на зовнішній носій та їхнє виведення за межі корпоративної мережі організації є однією з найбільш поширених форм викрадення. Такі витоки часто трапляються через необережність користувачів або в стресових ситуаціях, коли, наприклад, під час перевантаження роботою конфіденційні дані можуть бути надіслані не тому адресату або потрапити до третіх осіб. Хоча ці дії не завжди мають злочинний умисел, наслідки таких витоків часто непередбачувані. Наприклад, може статися випадок, коли документ з чутливою інформацією прикріплюється до листа і відправляється на помилкову адресу (рис.1.3).

Рисунок 1.3 – Ненавмисний витік даних [10]

Навмисне витікання відбувається, коли користувач, знаючи про політику безпеки організації, все одно вирішує надіслати конфіденційний документ. Це може статися, коли він обминає політики безпеки чи технічні заходи, не шукаючи особистої вигоди. До прикладу, подібне може мати місце, коли працівник змінює назву документа, його тип або копіює вміст без дозволу.

Зловмисний витік, по суті, є різновидом навмисного витоку, проте відрізняється тим, що його метою є отримання конкретної вигоди, часто у вигляді грошової компенсації (рис.1.4).

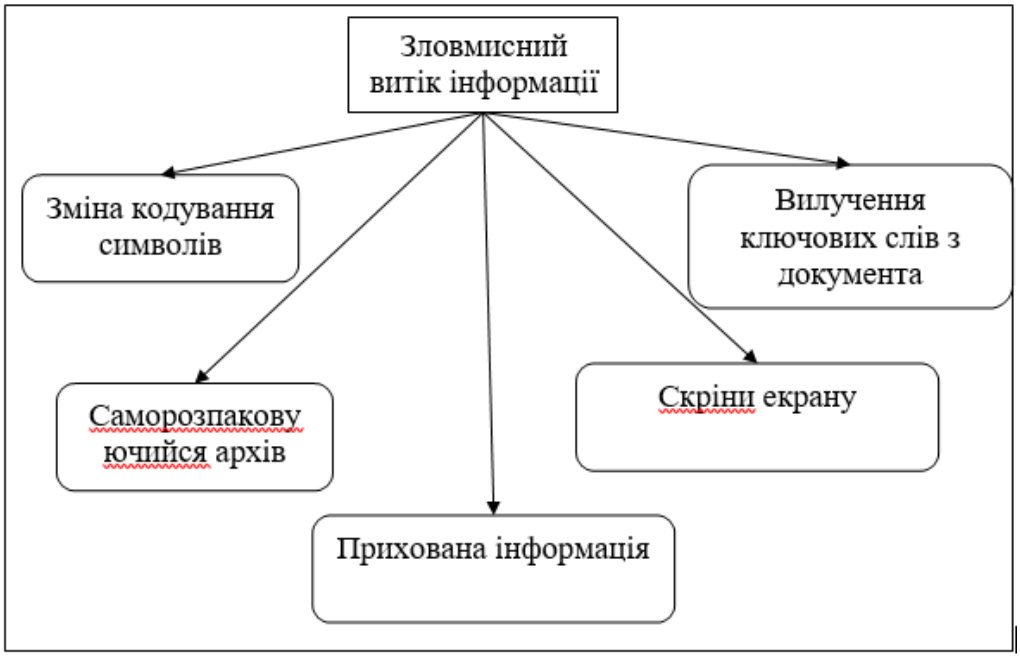


Рисунок 1.4 – Зловмисний витік

До прикладу, подібне може мати місце, коли працівник змінює назву документа, його тип або копіює вміст без дозволу.

1.2 Технології захисту даних

Операційний центр безпеки (SOC) є структурою, що відповідає за управління інцидентами інформаційної безпеки та забезпечення необхідного технічного обладнання для цього процесу. Це місце, де відбувається контроль, оцінка та захист корпоративних інформаційних систем, включаючи веб-сайти, додатки, бази даних, центри обробки даних, сервери, мережеві пристрої та кінцеві точки, як-от комп'ютери.

SOC працює як централізований підрозділ, що займається питаннями кібербезпеки та інформаційної безпеки на різних рівнях організації. Останнім часом такі центри набувають популярності в обох секторах – державному та приватному – завдяки ряду переваг, серед яких: оперативність у реагуванні на інциденти, зменшення ризиків зупинок у роботі мереж, удосконалення заходів захисту, підвищення ефективності діяльності, зменшення витрат і відповідність нормативним вимогам компенсації (рис.1.5).

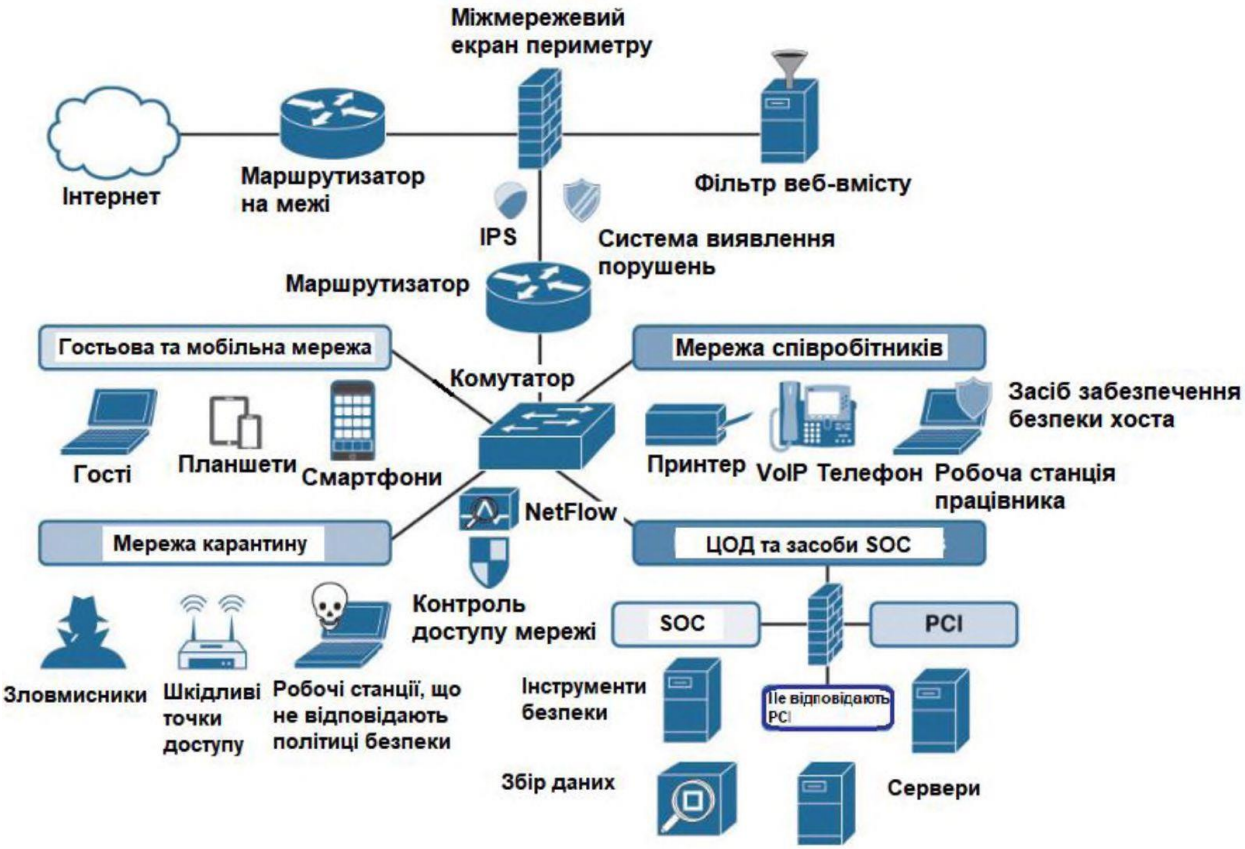


Рисунок 1.5 – Структура SOC [11]

Створення і впровадження SOC, а також його ефективне функціонування, часто стикаються з труднощами через великий обсяг технологій і складність управління численними процесами, які необхідно враховувати для досягнення бажаних результатів. SOC є важливим інструментом для комплексного підходу до моніторингу та реагування на інциденти відповідно до нормативних стандартів (ISO/IEC 27035, ISO/IEC 27001 тощо) [11].

SOC функціонує як центральний пункт, приймаючи дані з різних джерел: телеметрії, мережевих пакетів, системних журналів та інших елементів IT-інфраструктури організації. Він аналізує і корелює події для кожного інциденту, визначаючи, як з ними працювати і які дії необхідно вжити.

SOC зазвичай організовані на основі моделі "концентратор-кінцеві точки", де система управління інформацією про безпеку та подіями безпеки (SIEM) збирає та аналізує дані з різних безпекових каналів [12]. Кінцеві точки в цій схемі можуть охоплювати різні системи, зокрема інструменти для оцінки вразливостей, управління ризиками та дотримання стандартів безпеки, сканери додатків і баз даних, системи для запобігання вторгнень, аналітичні інструменти для вивчення поведінки користувачів і організацій, рішення для виявлення і виправлення проблем з кінцевими точками (EDR), а також платформи для аналізу загроз [13].

Для збору та аналізу даних часто використовуються SIEM-системи, які агрегують велику кількість інформації та представляють її в зручному вигляді для подальшого перегляду, пошуку та аналізу. Джерелами такої інформації можуть бути різноманітні системи. Одним із основних джерел є DLP-системи, які фіксують спроби витоків даних зсередини організації, а також порушення прав доступу.

Застосування DLP-систем дозволяє ефективно вирішувати ряд важливих завдань:

- запобігання несанкціонованому витоку та передачі конфіденційної інформації;
- зменшення фінансових та репутаційних ризиків;
- покращення дисципліни серед користувачів;
- проведення розслідувань інцидентів та аналіз їхніх наслідків;
- усунення загроз безпеці персональних даних та забезпечення відповідності стандартам захисту цієї інформації.

У контексті DLP дані можна розглядати в трьох основних станах (рис.1.6).

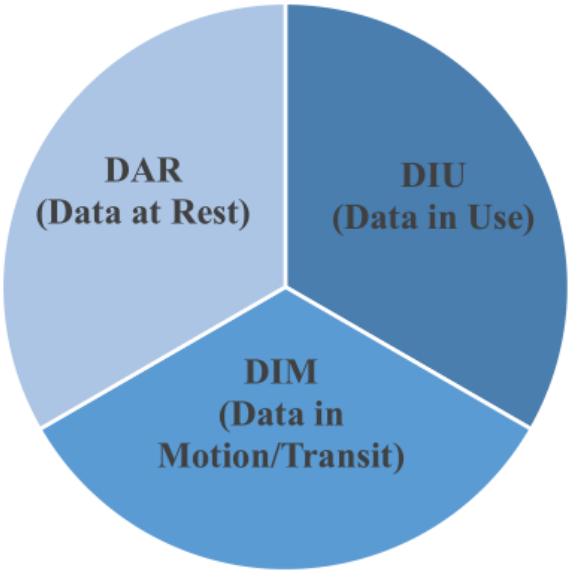


Рисунок 1.6 – Стани даних [14]

DAR (Data-at-Rest) – це дані, що знаходяться в спокої. Вони зберігаються на носіях інформації, таких як жорсткі диски, архіви баз даних, USB-накопичувачі, хмарні сховища та інші інфраструктурні рішення. У цьому стані не відбувається жодних маніпуляцій з даними, вони не передаються між мережами чи пристроями.

DIU (Data-in-Us) – це дані, які активно використовуються. Вони взаємодіють з користувачем або іншими процесами. Наприклад, вони можуть бути оновлені, оброблені, змінені або видалені.

DIM (Data-in-Motion) – дані, що знаходяться в процесі переміщення. Це означає, що дані переміщуються з одного місця в інше, наприклад, між комп'ютерами, мережами, хмарами, через інтернет, електронну пошту або месенджери. Коли дані досягають кінцевої точки призначення, вони повертаються у стан спокою [14, 15].

Сучасна DLP-система являє собою комплексний програмно-апаратний розподілений набір, який складається з кількох основних компонентів, що працюють на спеціалізованих серверах, пристроях користувачів та на рівні внутрішньої служби безпеки:

– модулі бази даних, аналізу та обробки інформації, що стосується інцидентів, виявлених системою, а також даних, що потребують контролю та моніторингу;

– модулі для пасивного та активного спостереження за діями співробітників організації, визначаючи заздалегідь обмежений список дій, що підлягають контролю. Це може включати моніторинг входу та виходу з системи, передачу даних через бездротові канали, підключення зовнішніх носіїв, друк документів тощо;

– модулі управління та налаштування, які забезпечують моніторинг роботи системи, аналітичну обробку та підтримку функцій безпеки (рис.1.7).

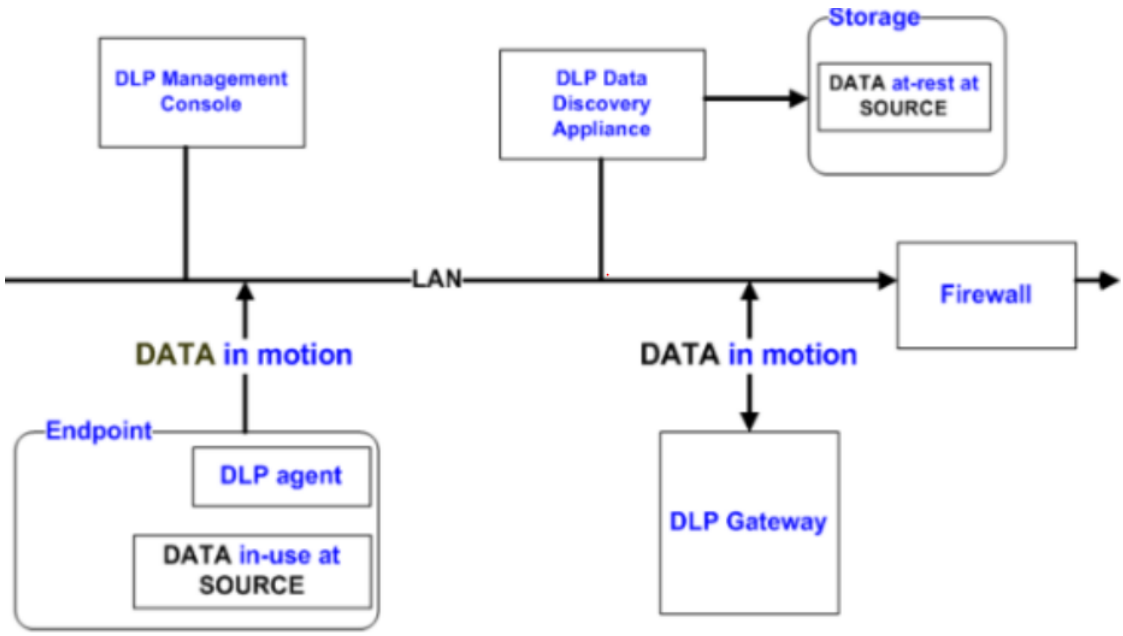


Рисунок 1.7 – Базова архітектура DLP-системи [16]

Існує три основних типи DLP [17]:

– мережевий DLP займається відстеженням та захистом всіх даних, що використовуються, перебувають у спокої або знаходяться в мережі організації, включаючи хмарні сервіси;

– DLP кінцевих точок фокусується на моніторингу та захисті кінцевих точок, таких як сервери, комп'ютери, ноутбуки, мобільні телефони та інші пристрої, де відбувається використання, передача або зберігання даних;

– хмарний DLP є підмножиною мережевого DLP, яка спеціалізується на захисті даних, збережених у хмарних сховищах, забезпечуючи безпеку для організацій, що активно використовують хмарні ресурси для зберігання інформації.

В загальному на підпрємстві інформація переміщається по каналах, наведених на рисунку 1.8.

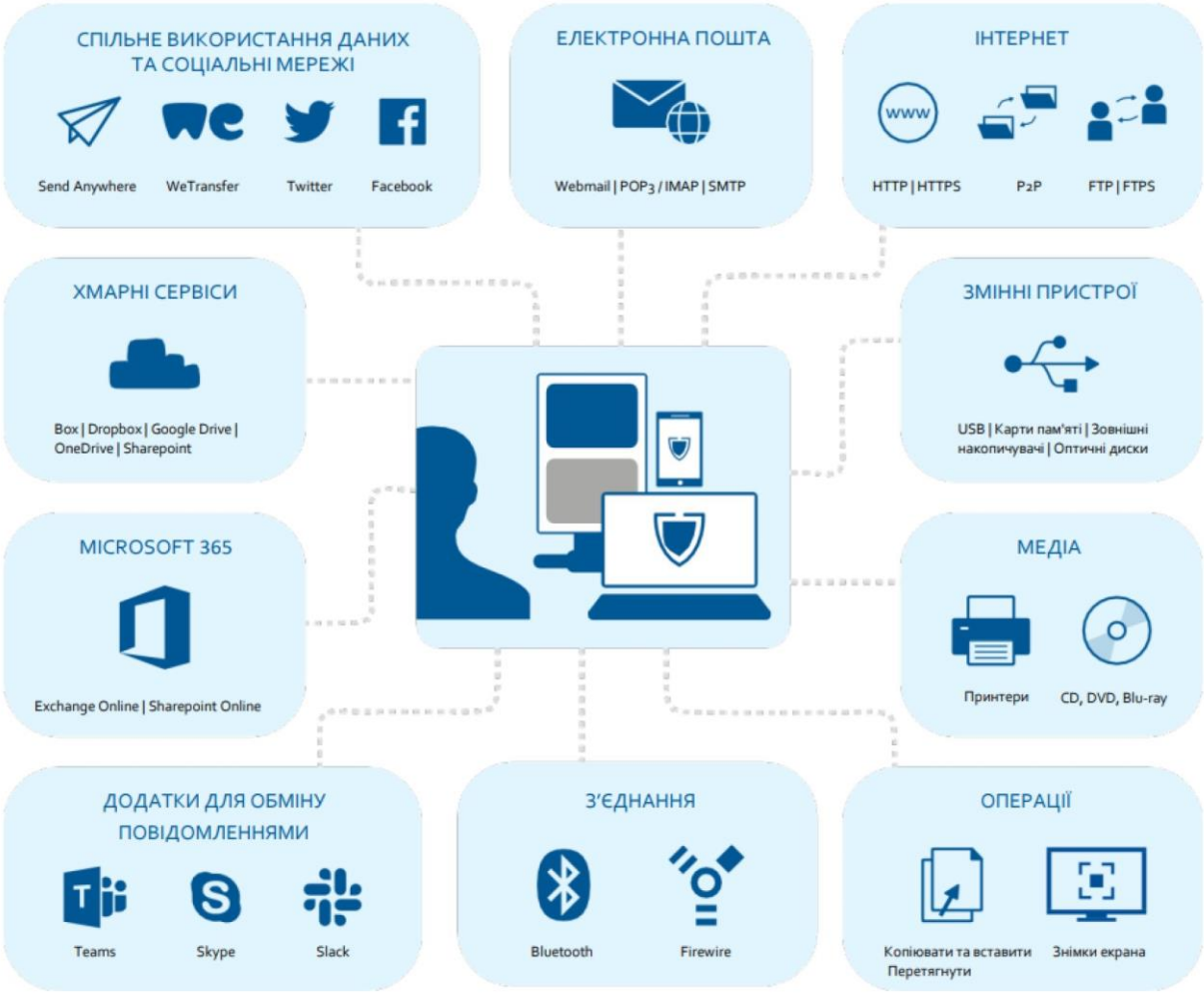


Рисунок 1.8 – Канали переміщення інформації [18]

Основними канали витоку інформації в корпоративній мережі підприємства відбуваються по протоколах HTTP, SMTP та FTP. На кінцевих пристроях основні канали витоку зображені на рисунку 1.9.

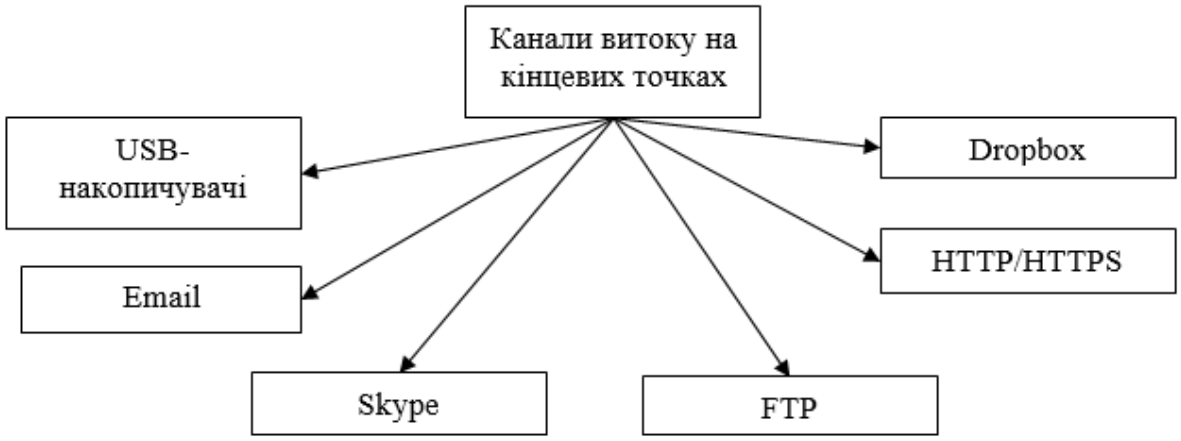


Рисунок 1.9 – Основні канали витоку інформації

Для запобігання витоку конфіденційної інформації важливо спочатку виявити, коли та де можуть виникнути загрози. Існує кілька підходів до виявлення таких витоків. Один із них полягає в застосуванні аналізу вмісту, що дозволяє виявити чутливу інформацію, яка може бути передана або збережена без належного контролю. Цей метод включає використання шаблонів, що ідентифікують конкретні види даних, такі як фінансові реквізити або особисті дані.

Іншим підходом є моніторинг поведінки користувачів, що включає аналіз звичних дій співробітників і виявлення аномалій у їхній діяльності. Якщо співробітник намагається доступити або передати дані, які не відповідають його звичайним обов'язкам, система може виявити таке порушення і попередити про можливий витік.

Також широко використовуються методи на основі аналізу метаданих, які дозволяють визначити, як і коли інформація переміщається в межах організації. Цей підхід дозволяє відстежувати передачу файлів між різними пристроями або мережами, що є важливим для виявлення потенційних витоків даних. Всі ці

методи забезпечують ефективний контроль за конфіденційністю інформації і допомагають мінімізувати ризики витоків.

Перевірка на основі контексту зосереджена на моніторингу та аналізі різних систем безпеки, таких як брандмауери, проксі-сервери, системи виявлення та запобігання вторгнень, а також фільтрація спаму. Основна мета цієї моделі – відслідковувати контекстну інформацію про передані дані, включаючи відправника, отримувача, розмір, мету, заголовки, метадані і типи файлів. Прикладом цього підходу є фільтри пакетів, що використовуються в брандмауерах, які вирішують, чи може конкретний пакет даних продовжити рух по мережі. Для фільтрації пакетів можуть використовуватися такі атрибути, як IP-адреси відправника та отримувача, порти і інші характеристики пакетів. Цей метод дозволяє детально аналізувати передачу інформації і забезпечувати її захист на основі контексту.

Аналіз з використанням словника ключових слів полягає в перевірці даних на наявність певних термінів або шаблонів, які вказують на конфіденційну інформацію. Наприклад, ключові слова, як «пароль», «пароль», «конфіденційно», можуть бути виявлені під час аналізу. Сфера діяльності організації впливає на набір таких слів, оскільки вони повинні відповідати специфіці діяльності та внутрішнім процесам. До того ж, можуть використовуватися шаблони, такі як 16-значний номер кредитної картки, що дозволяє автоматично виявляти важливі дані. Багато систем захисту також включають стандартизовані словники, що враховують вимоги законодавчих актів, наприклад PCI-DSS для захисту платіжних карток або HIPAA, що регулює захист медичних даних. Цей метод є швидким і простим у впровадженні, адже не потребує складних алгоритмів або інфраструктурних змін [19]:

Метод зняття цифрових відбитків полягає у створенні унікальних цифрових «відбитків пальців» для конфіденційних файлів та записів баз даних, що дозволяє виявляти витоки даних через порівняння цих відбитків з іншими. Ці відбитки є хеш-значеннями, які точно відповідають конкретному набору даних і не

змінюються при подальшому використанні цих самих даних. Хеші конфіденційних даних зберігаються локально на захищених пристроях або в спеціалізованих базах даних для подальшого порівняння. При передачі або обробці даних система перевіряє їх хеш-значення та порівнює з існуючими записами, щоб виявити будь-які несанкціоновані зміни або передачу інформації. Таким чином, метод допомагає запобігти несанкціонованому доступу і контролювати витоки даних, адже будь-яка спроба використання цих даних в інших системах буде зафіксована завдяки зміщенню їхніх хешів [20, 21].

Метод навчання полягає в застосуванні технологій машинного навчання для автоматичного аналізу і класифікації даних на основі їхнього рівня конфіденційності [22 - 25]. Один з поширених підходів у цьому методі – використання моделей, таких як Vector Space Model (VSM), для оцінки контексту і важливості даних в межах інформаційної системи. В основі такого методу лежить аналіз взаємозв'язків між різними елементами даних і визначення, наскільки ці дані є чутливими або важливими для організації. За допомогою алгоритмів машинного навчання система може навчатися на основі історичних даних, створюючи моделі, що дозволяють автоматично ідентифікувати та класифікувати конфіденційну інформацію, а також прогнозувати й запобігати можливим витокам. Цей підхід не лише ефективно аналізує дані, але й здатен адаптуватися до змін у поведінці користувачів і умовах інформаційного середовища, що забезпечує гнучкість і точність виявлення потенційних загроз.

Для виявлення витоків даних з використанням машинного навчання доцільно застосовувати різні моделі, в залежності від характеру завдання та типу даних. Лінійні моделі, такі як логістична регресія або метод опорних векторів, підходять для класифікації даних, наприклад, для визначення, чи є інформація конфіденційною чи ні. Вони ефективні при чітких межах між класами, що дозволяє швидко оцінити ризики. Моделі, що базуються на рішеннях дерев, такі як Random Forest, дозволяють аналізувати складні взаємозв’язки між даними, а

також адаптуватися до різних типів інформації, що важливо при виявленні витоків.

Глибинне навчання, зокрема нейронні мережі, здатні обробляти великі обсяги неструктурованих даних, таких як текст або зображення. Вони допомагають розпізнавати складні патерни в даних, що дозволяє виявляти аномалії, які можуть свідчити про витоки. Наївний баєсівський класифікатор, який використовує ймовірності, може бути корисним для текстової класифікації та оцінки ризиків витоку на основі попередніх спроб витоків або порушень.

Методи кластеризації, такі як K-means, допомагають ідентифікувати аномалії, групуючи подібні дані, що дозволяє виявити незвичні або підозрілі шаблони поведінки. Генеративні моделі, зокрема генеративні змагальні мережі (GANs), можна використовувати для створення фальшивих даних або моделювання атак, що допомагає системам безпеки підготуватися до нових методів витоку.

Методи підкріпленого навчання, як-от Reinforcement Learning, можуть бути використані для покращення реакції на інциденти та оптимізації процесу виявлення витоків. Крім того, застосування крос-валідації і ансамблевих методів дозволяє підвищити точність моделей і зробити систему більш стійкою до помилок.

Маркування вмісту є підходом, що дозволяє позначати конфіденційну інформацію у файлах або документах, забезпечуючи її подальший контроль та захист. Ідея цього методу полягає в тому, щоб додавати спеціальні мітки або теги до файлів, що містять чутливі дані. Такі маркування зберігаються навіть після обробки файлів іншими програмами, що може включати їх шифрування, перекодування або стиснення в архіви. Це дає змогу забезпечити додаткову безпеку даних, оскільки маркування допомагає системам захисту ідентифікувати та обробляти ці файли як конфіденційні, незалежно від того, які маніпуляції були з ними здійснені. Цей метод гарантує, що важлива інформація залишається захищеною навіть при передаванні через небезпечні канали [26].

1.3 Постановка завдання

Сучасні DLP-системи значно розширили спектр своїх можливостей і функцій, що адаптуються до нових викликів. Для розробки ефективної системи захисту даних у локальній мережі необхідно провести детальний аналіз інформаційних потоків, які циркулюють у межах цієї мережі.

Окрім того, важливо враховувати економічну доцільність проєкту, намагаючись мінімізувати витрати на його впровадження. Ефективний підхід до боротьби з цими загрозами передбачає не лише впровадження інноваційних систем захисту, але й роботу з персоналом. Важливо створювати умови, що знижують ризик внутрішнього саботажу, такі як покращення умов праці, етичне ставлення до співробітників і моніторинг їхньої активності для виявлення потенційних загроз.

Основною метою роботи є аналіз типових проблем у сфері інформаційної безпеки, які стосуються витоків даних, а також розробка ефективної системи захисту від таких витоків. Ця система повинна забезпечувати відслідковування та моніторинг чутливих даних в межах інформаційного простору компанії та здатність виявляти неправомірні дії користувачів, які можуть призвести до витоку цих даних. Для досягнення цієї мети необхідно вирішити кілька ключових завдань. По-перше, потрібно провести всебічний аналіз найбільш поширених проблем інформаційної безпеки, що виникають у разі витоку даних. По-друге, слід розробити модель функціонування системи захисту від витоку даних, яка буде ефективно відслідковувати та контролювати потоки конфіденційної інформації. По-третє, необхідно створити алгоритм для оцінки чутливої інформації, що дозволить визначати її рівень та важливість для організації. І нарешті, останнім завданням є розробка відповідного програмного забезпечення, яке реалізує запропоновані моделі та алгоритми, забезпечуючи надійний захист та моніторинг інформаційної безпеки в організації.

У межах реалізації проєкту необхідно розробити сервер, який виконуватиме низку ключових функцій. Серед основних завдань серверу – забезпечення можливості додавання документів із конфіденційною інформацією для подальшого моніторингу, водночас гарантуючи, що такі дані не зберігатимуться безпосередньо на сервері. Для підвищення доступності та зручності взаємодії сервер може бути реалізований у формі вебсайту.

Для забезпечення ефективного моніторингу та контролю інформації, що циркулює на кінцевих точках, таких як робочі станції користувачів, необхідно створити клієнтський додаток із широким функціоналом. Однією з ключових вимог є можливість автоматичного запуску додатка разом із завантаженням операційної системи. Це забезпечить постійний контроль і безперервну роботу додатка. Крім того, важливо, щоб розроблений додаток підтримував роботу на різних платформах і операційних системах, що дозволить застосовувати його на різноманітних пристроях.

2 ПРОЄКТУВАННЯ СИСТЕМИ БЕЗПЕКИ

2.1 Обґрунтування обраного підходу

Головним завданням розробленої системи є впровадження превентивних заходів для запобігання несанкціонованому витоку інформації в інформаційно-комунікаційному середовищі відділення протидії кіберзлочинам. Система має забезпечити ефективний контроль та моніторинг обробки чутливих даних, вчасно виявляючи та блокуючи спроби витоку інформації. Це допоможе значно підвищити рівень безпеки в організації, зменшуючи ризики несанкціонованого доступу до даних та їх витоку.

Вона має забезпечувати захист чутливих даних на всіх етапах їх обробки, зберігання та передачі. Це охоплює три ключові фази: дані в стані спокою, дані в процесі обробки, та дані, що переміщуються між системами або по мережах. Кожна з цих фаз має свої унікальні вимоги та підходи до захисту, що вимагають різних технічних рішень та стратегій.

В основному використовуються паперові носії інформації, хоча в останні роки активно здійснюються спроби інформатизації робочого процесу, зокрема шляхом впровадження електронного документообігу. Це дозволяє значно полегшити доступ до інформації, пришвидшити обробку документів та зменшити фізичне навантаження на співробітників. Однак використання паперових носіїв інформації має свої переваги: вони можуть бути менш вразливими для атак зловмисників, оскільки фізичні документи важче піддаються несанкціонованому копіюванню чи видаленню без залишкових слідів. У той же час, впровадження електронних систем, хоча і спрощує доступ і пошук даних, все ж вимагає належного рівня захисту, оскільки відсутність належних програмно-апаратних засобів для захисту електронних носіїв може збільшити ймовірність витоку або компрометації даних. Окрім того, перехід до електронних форм документів потребує додаткової уваги до питання кібербезпеки, створення ефективних

систем резервного копіювання та засобів захисту від несанкціонованого доступу, щоб уникнути можливих загроз для конфіденційності даних.

Захист конфіденційної інформації забезпечується завдяки використанню функціоналу та технологій, спрямованих на захист даних від витоків. Одним із основних інструментів є DLP-система (система запобігання витокам даних), яка складається з двох основних типів модулів: хост-модулю та мережного модуля.

Хост-модулі встановлюються безпосередньо на робочі станції користувачів і контролюють їх дії, спрямовані на взаємодію з конфіденційною інформацією. Вони відстежують, які операції здійснюються з даними, де саме і в який спосіб ці дані використовуються, що дозволяє вчасно виявити спроби несанкціонованого доступу або витоку інформації [27].

Мережний модуль фокусується на моніторингу та аналізі всього трафіку, який передається по локальній мережі організації. Цей модуль перевіряє, чи містить переданий трафік конфіденційну інформацію, і якщо така інформація виявляється, мережний модуль негайно блокує або припиняє її передачу. Це дозволяє запобігти можливому витоку даних за межі організації через інтернет або інші канали комунікації [28].

Завдяки комбінації цих двох модулів система може забезпечити комплексний захист даних на рівні як окремих пристроїв, так і на рівні мережі, що робить її ефективним інструментом для запобігання витокам інформації в межах організації.

У фазі даних у стані спокою система контролює та захищає інформацію, що зберігається на жорстких дисках, базах даних, хмарних платформах або інших носіях інформації. Технології шифрування, контроль доступу та політики резервного копіювання є основними засобами захисту на цьому етапі.

У фазі даних в процесі обробки система здійснює моніторинг дій користувачів, програмних процесів та операцій з даними. Сюди включається контроль за доступом до документів, виявлення несанкціонованих змін або спроб витоку даних. Для цього використовуються такі інструменти, як хост-модулі

DLP-систем, що аналізують активність на рівні користувача та забезпечують захист під час виконання операцій з чутливою інформацією.

У фазі даних в русі система контролює передачу даних через мережу, перевіряючи, чи не містять вони конфіденційну інформацію, яка може бути витікана за межі організації. Мережні модулі DLP-систем, як правило, аналізують весь трафік, що проходить через корпоративну мережу, блокуючи або обмежуючи передачу чутливої інформації.

Загальна структура такої системи дозволяє створити багаторівневий захист, що покриває всі аспекти обробки та переміщення конфіденційних даних, що є важливою складовою частиною політики інформаційної безпеки будь-якої організації (рис. 3.1).

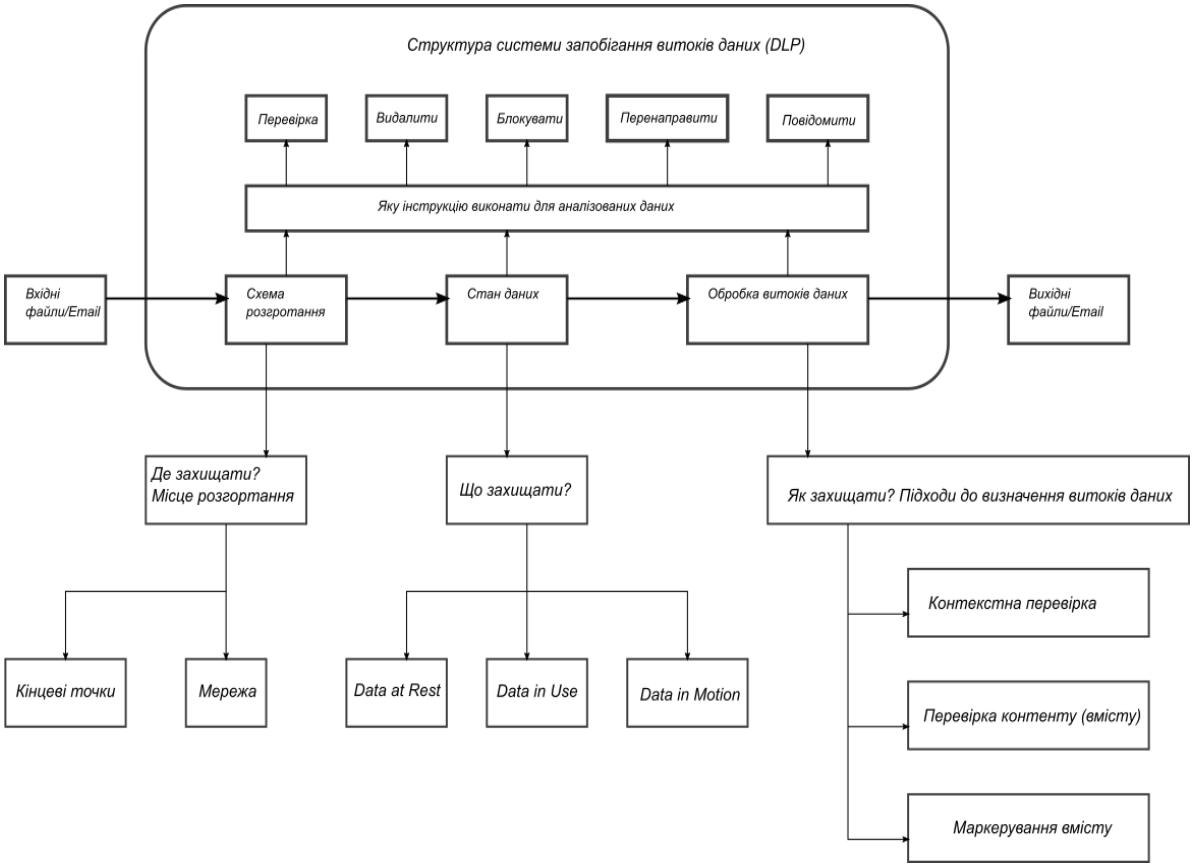


Рисунок 2.1 – Загальна структура DLP-системи [29]

Призначення цієї системи полягає у виявленні, сповіщенні та запобіганні несанкціонованому використанню або передачі конфіденційної інформації. Вона

здатна працювати в різних інформаційних середовищах, де захист інтелектуальної власності, персональних даних та іншої важливої інформації є критично важливим.

Розробка системи для запобігання витокам даних здійснюватиметься на основі клієнт-серверної архітектури, що забезпечить розподіл функцій між постачальниками послуг та їх користувачами. У цій архітектурі роль провайдера та клієнта виконуватиме програмне забезпечення. Зазвичай постачальника послуг називають сервером. Основний спосіб взаємодії між клієнтом та сервером полягає в обміні запитами через мережу (рис. 2.2).

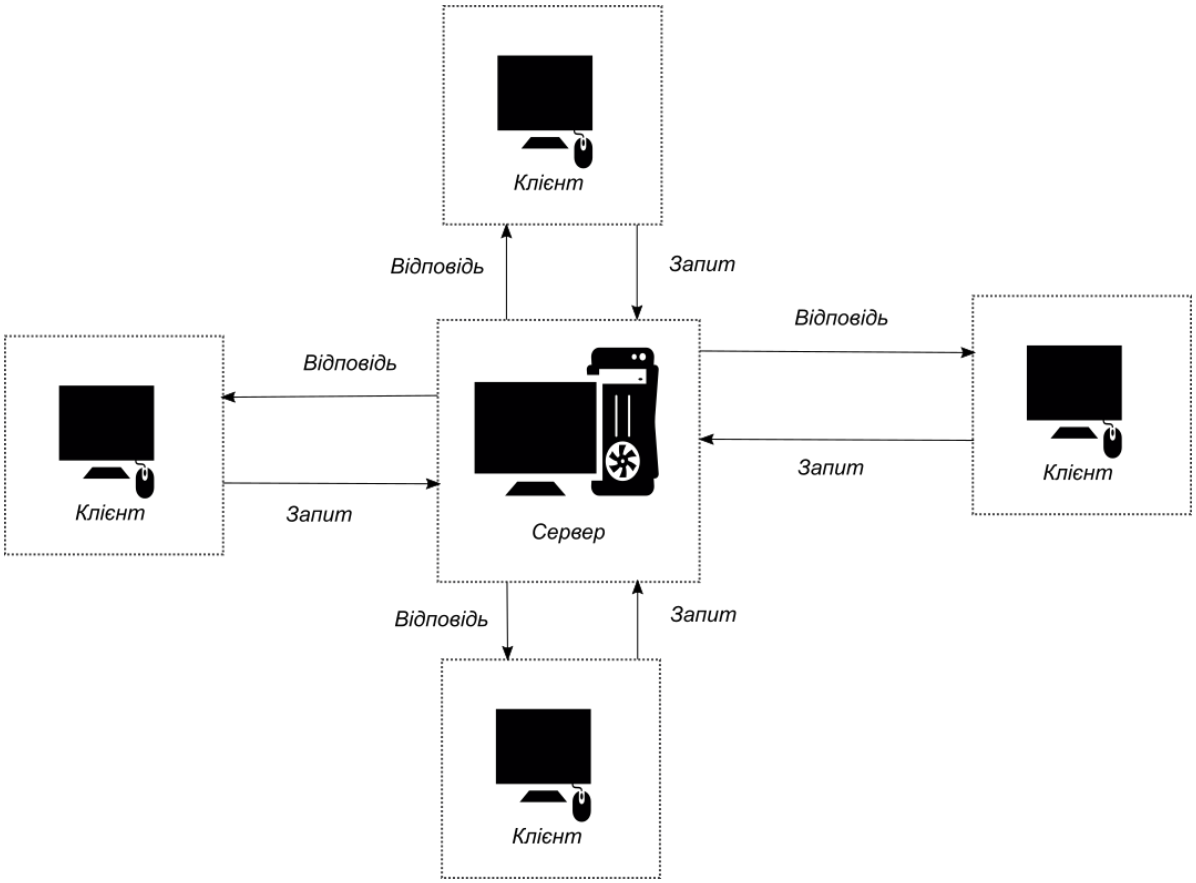


Рисунок 2.2 – Клієнт-серверна архітектура [30]

Клієнт являє собою програму на робочій станції користувача, яка формує запити до сервера і, отримавши відповіді, виконує певні операції. Сервер, у свою чергу, є програмою на спеціалізованому обладнанні, яке налаштоване на прийом

запитів від кількох клієнтів, їх обробку відповідно до внутрішньої логіки і надання відповідей на основі результатів цієї обробки.

Система застосовується для забезпечення захисту від інцидентів інформаційної безпеки, пов'язаних з несанкціонованим доступом до даних та їх витоками. Вона має бути здатною оперативно виявляти загрози і вжити заходів для їх нейтралізації, що дозволяє мінімізувати можливі збитки від таких інцидентів.

У рамках реалізації проєкту необхідно створити сервер, який матиме кілька основних функцій. Однією з головних можливостей буде додавання документів з конфіденційною інформацією для подальшого відслідковування. Водночас важливо, щоб конфіденційні дані не зберігалися безпосередньо на сервері. Для зручності користувачів сервер може бути реалізований у вигляді вебсайту, що забезпечить доступність та зручність взаємодії.

Сервер повинен мати можливість отримувати повідомлення від клієнтів, які містять дані, що передаються через кінцеві точки. Зібрані дані підлягатимуть лінгвістичному аналізу, за результатами якого буде проводитися оцінка, чи містять передані дані конфіденційну інформацію організації. Після цього сервер надсилатиме повідомлення-відповідь, що ґрунтується на проведеній оцінці.

Архітектура сервера має бути двохланковою, що забезпечить гнучкість і масштабованість системи [30]. Крім того, сервер повинен бути абстрагованим від структури бази даних, що містить необхідну для роботи систему інформацію. Важливо також реалізувати рольову політику, щоб забезпечити розмежування доступу до сервера, визначаючи, які функції доступні для кожного користувача в залежності від його ролі [31].

Також сервер повинен забезпечувати класифікацію інформації за рівнем критичності її витоку для організації, щоб у разі інциденту можна було оперативно вжити відповідних заходів. Крім того, необхідно реалізувати можливість надання ключів для операцій шифрування та дешифрування даних, причому доступ до цих ключів повинен регулюватися політикою розмежування доступу.

Система повинна бути незалежною від платформи, що дозволить її використовувати на різних операційних системах та в різних середовищах. Сервер також має містити функціонал для звітності для офіцерів інформаційної безпеки та забезпечити своєчасне повідомлення про інциденти кібербезпеки, що виникають у процесі роботи системи.

Для забезпечення ефективного контролю за циркулюючою інформацією на кінцевих точках, таких як робочі станції користувачів, необхідно розробити клієнтський додаток з низкою функцій. Одна з основних вимог – самозавантаження додатка в операційній системі, що дозволить йому автоматично запускатися при включенні пристрою, забезпечуючи безперервний моніторинг і контроль. Крім того, додаток повинен бути сумісним з різними платформами і операційними системами, що дозволить використовувати його на різних типах обладнання.

Клієнтський додаток має вміти відслідковувати підключення зовнішніх пристроїв, таких як USB-накопичувачі, зовнішні жорсткі диски, або інші периферійні пристрої, що можуть використовуватися для передачі даних. Також важливим аспектом є відстеження інформації, що перебуває в буфері обміну, оскільки це є важливим каналом для несанкціонованих передач даних.

Ще однією важливою функцією є можливість надсилати повідомлення з інформацією про операції, що виконуються на робочому місці користувача, до центрального сервера для подальшого аналізу. На основі отриманих даних сервер може здійснити оцінку та надати рекомендації або вжити заходів для запобігання витоку конфіденційної інформації.

Додаток також повинен включати функцію авторизації користувачів для забезпечення правильного доступу та ідентифікації осіб, які взаємодіють з конфіденційними даними. Для забезпечення високого рівня безпеки клієнтський додаток має підтримувати шифрування і дешифрування файлів, що дозволить захистити інформацію під час її зберігання або передавання.

Таким чином, клієнтський додаток буде ефективно взаємодіяти з серверною частиною системи для здійснення контролю за даними на кінцевих точках, а також забезпечуватиме належний рівень захисту від можливих витоків конфіденційної інформації.

Для здійснення адміністрування системи та додавання файлів на відслідковування, модуль контролю буде реалізований у вигляді вебдодатку, що є інтерфейсом для зручної взаємодії з системою. Ця панель складатиметься з двох основних частин: меню та робочої зони (рис. 2.3).

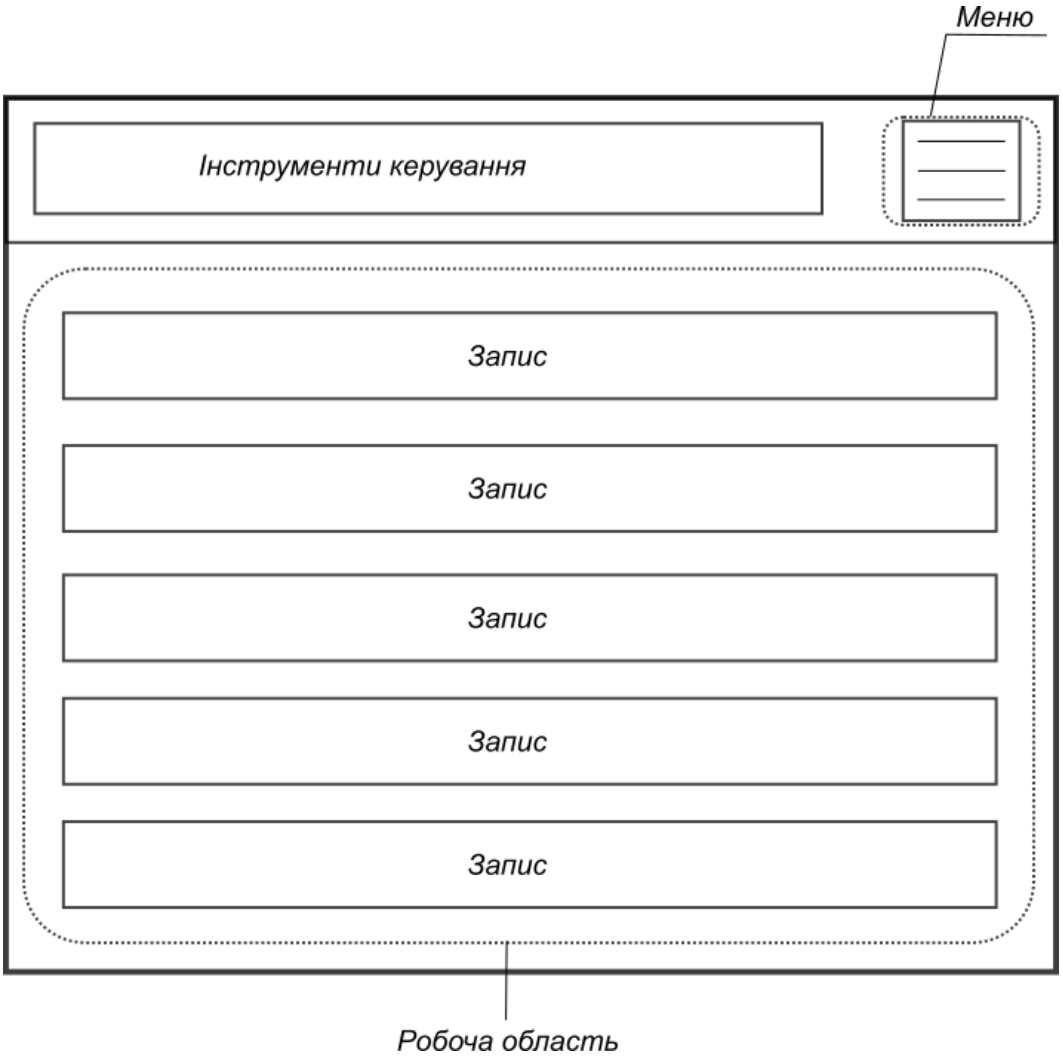


Рисунок 2.3 – Структура адміністративної панелі вебдодатку

Меню вебдодатку міститиме кілька основних пунктів, що дозволяють здійснювати необхідні операції з системою. Перший пункт меню, "Керування

інформацією, що відслідковується", дозволить адміністратору додавати та налаштовувати файли для подальшого контролю. Другий пункт меню, "Керування користувачами", дасть змогу адміністратору додавати нових користувачів, змінювати їхні права доступу та здійснювати управління їхніми акаунтами. Третій пункт меню, "Керування ключами криптографічного захисту", буде відповідати за налаштування та видачу криптографічних ключів для забезпечення безпеки переданих даних, а також за їх управління.

Робоча зона вебдодатку надаватиме адміністраторам зручний інтерфейс для виконання цих операцій, забезпечуючи інтерактивність, ефективність і зручність у використанні.

Робоча зона вебдодатку буде адаптуватися в залежності від вибору пункту меню. При виборі "Керування інформацією, що відслідковується", адміністратор отримає можливість завантажити файл або додати текст, який буде оброблятися системою. Після завантаження текст проходить кілька етапів: спочатку здійснюється нормалізація тексту, потім він розбивається на лексеми, і на кожну лексему проводиться токенізація. Результати цих процесів зберігаються в сховищі даних системи у вигляді цифрових відбитків (токенів). Важливо, що сама конфіденційна інформація не зберігається в системі. Додатково в робочій зоні буде доступна можливість видаляти застарілі або неактуальні для захисту дані, а також визначати пріоритети захисту для окремих файлів чи текстів.

При виборі пункту "Керування користувачами" адміністраторам буде надано інтерфейс для здійснення операцій додавання, редагування та видалення користувачів, а також для керування їхніми ролями в системі. Ця функція дасть змогу точно налаштувати права доступу та забезпечити необхідний рівень безпеки для кожного користувача.

В пункті "Керування криптографічними засобами захисту" відображатимуться ключі алгоритмів шифрування, що були застосовані до файлів із конфіденційною інформацією. Ключі можуть бути надані користувачам згідно з політикою безпеки та їхніми правами доступу, що дозволяє здійснювати

контроль за криптографічним захистом даних на різних етапах їх обробки та зберігання.

На рисунку 2.4 можна спостерігати різноманітні інформаційні потоки, що існують у межах підприємства, для прикладу розглянемо рекламне агенство.

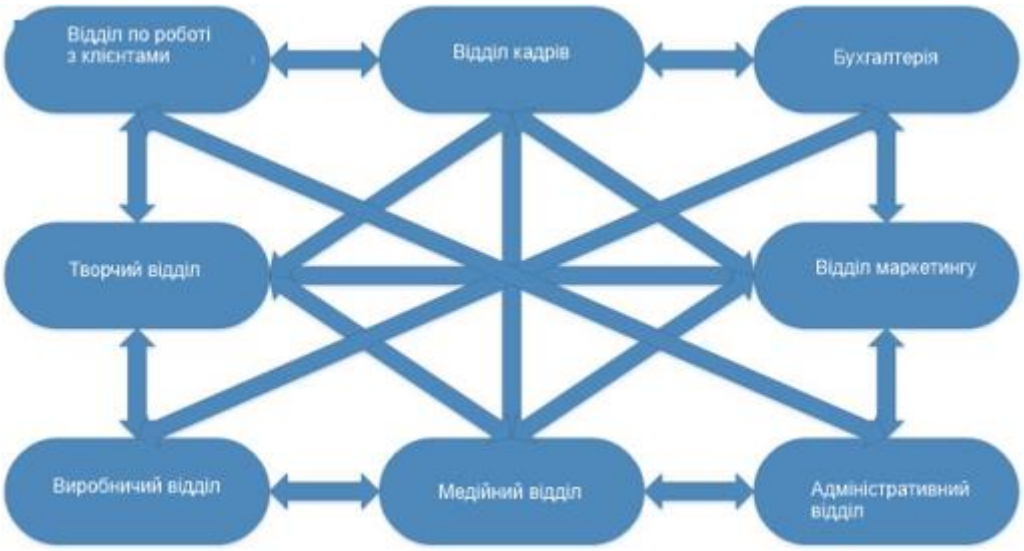


Рисунок 2.4 – Інформаційні потоки

Горизонтальні інформаційні потоки займають важливе місце в організаційній комунікації, оскільки передача інформації між людьми на одному рівні ієрархії забезпечує її найменшу втрату. Зазвичай, такі потоки мають неофіційний характер, але інколи вони можуть бути формалізовані, коли необхідно забезпечити швидкий обмін знаннями між співробітниками, які стикаються з подібними завданнями. Як правило, цей тип комунікації дозволяє зберегти до 85% переданих даних, оскільки співробітники часто мають спільний досвід роботи, працюють в одних відділах або надають підтримку один одному в процесі вирішення проблем.

Низхідні інформаційні потоки, що йдуть від керівництва до нижчих ланок, зазвичай є формальними, хоча існують і неформальні випадки. Зі збільшенням кількості ланок в організації, інформація поступово втрачає точність і достовірність. Спостерігається тенденція до спотворення даних, що є результатом комунікаційних бар'єрів або різниці у сприйнятті та інтерпретації інформації на

різних рівнях. Кожен етап передачі інформації може призвести до втрати до 40% її змісту, що обумовлено спотворенням або неповнотою її передачі. Такий процес у практиці часто можна порівняти з ефектом "зіпсованого телефону", коли інформація змінюється на кожному етапі.

Висхідні інформаційні потоки, які надходять від нижчестоящих працівників до керівництва, мають значно менше значення в плані неформальної комунікації. Однак ці потоки часто страждають від найбільших спотворень. Інформація, що надходить із низу, може бути сильно змінена або взагалі втрачена. Якщо компанія або організація не стимулює зворотний зв'язок від співробітників і не впроваджує механізмів збору ідей, це може суттєво обмежити її інноваційний потенціал і завадити розвитку.

2.2 Архітектура системи запобігання витоків даних

Розглянемо детально принципи роботи DLP системи. На вході система отримує дані різних типів: це можуть бути файли різних форматів, текстові дані або мережевий трафік. Кожен тип даних спочатку проходить через відповідний модуль попередньої обробки, який перетворює їх у формат, придатний для подальшого аналізу [32].

Модуль аналізу контенту є центральним компонентом системи. Він застосовує різні методи виявлення чутливої інформації. Перш за все, використовуються регулярні вирази для пошуку структурованих даних, таких як номери кредитних карток, паспортні дані чи телефонні номери. Паралельно відбувається пошук ключових слів, які можуть вказувати на конфіденційний характер інформації. Також система порівнює хеші файлів з базою відомих конфіденційних документів (рис. 2.5).

Після виявлення потенційно чутливої інформації система агрегує результати всіх перевірок. На цьому етапі застосовуються політики безпеки, які

визначають, як саме слід реагувати на різні типи виявлених даних. Якщо виявлено порушення політики безпеки, система генерує інцидент безпеки.

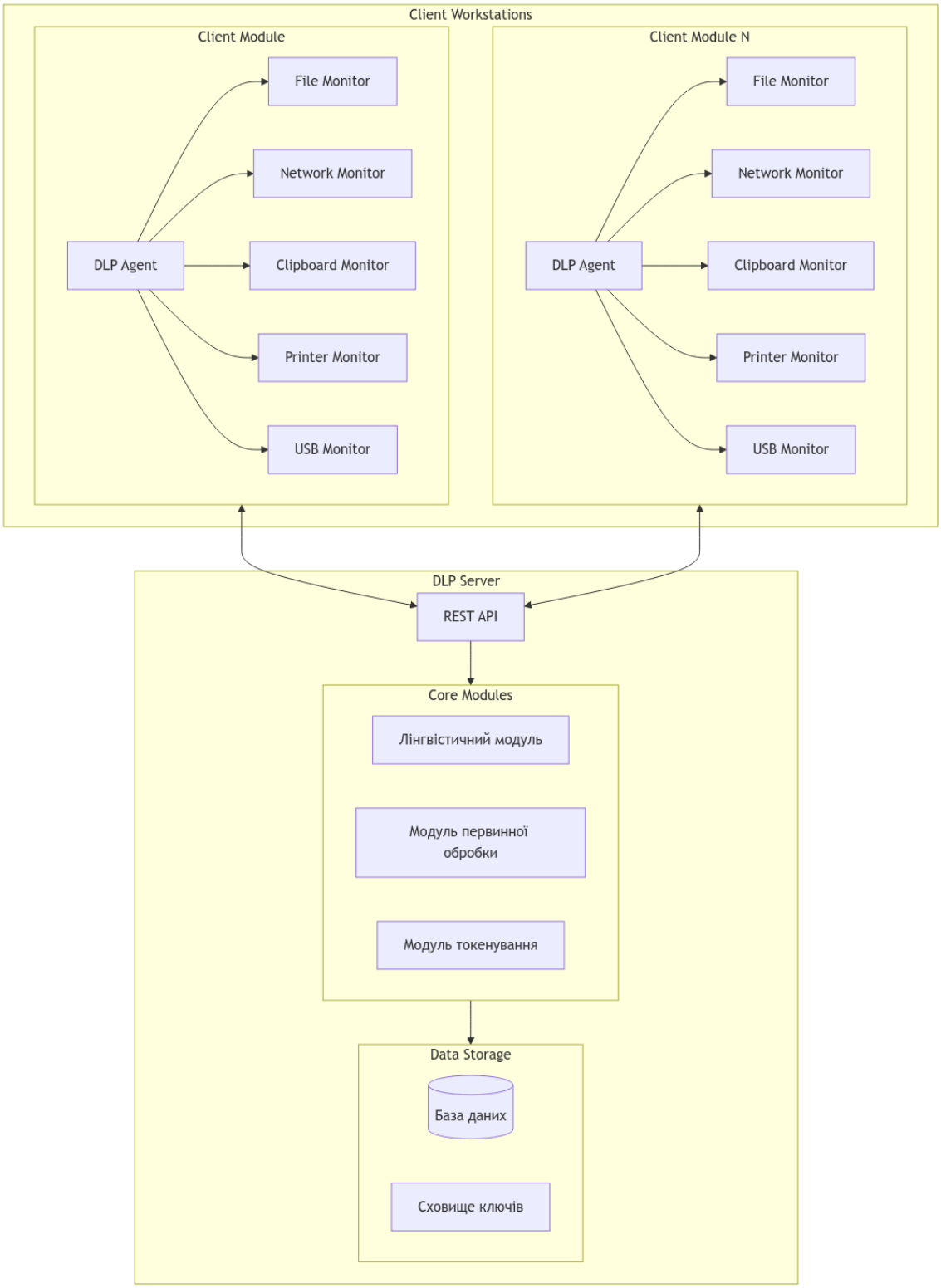


Рисунок 2.5 – Схема роботи системи запобігання витоку даних

Обробка інциденту включає кілька паралельних процесів. Система створює детальний запис у журналі подій, який містить всю релевантну інформацію про інцидент. Одночасно відправляються сповіщення відповідальним особам через налаштовані канали комунікації. У разі необхідності відбувається активне блокування передачі даних для запобігання витоку.

Якщо порушень не виявлено, дані отримують дозвіл на передачу і проходять далі по бізнес-процесу. Незалежно від результату перевірки, інформація про всі операції надходить до модуля звітності та аналітики.

Система постійно накопичує статистику та метрики, які використовуються для формування звітів та вдосконалення правил виявлення. Це дозволяє адаптувати систему до нових загроз та покращувати точність детектування чутливої інформації. Важливою особливістю є можливість тонкого налаштування правил та політик безпеки відповідно до специфіки організації.

Для реалізації системи захисту від витоків даних пропонується створити загальну архітектуру, яка забезпечить ефективну взаємодію всіх її компонентів. Центральним елементом системи стане вебсервер, що виконуватиме функцію ядра керування. Цей сервер забезпечуватиме обмін даними з іншими елементами через API, що дозволить інтегрувати всі складові системи в єдину інформаційно-комунікаційну структуру.

Система включатиме клієнтські додатки, встановлені на робочих станціях користувачів, які виконуватимуть безперервний моніторинг та контроль дій із конфіденційною інформацією. Вебсервер, в свою чергу, здійснюватиме збір, обробку та аналіз даних, що надходять від клієнтів, забезпечуючи централізоване управління та прийняття рішень.

Інтеграція системи в інформаційно-комунікаційне середовище організації дозволить виявляти та аналізувати інциденти інформаційної безпеки, пов’язані з несанкціонованою передачею конфіденційної інформації. Такий підхід не лише забезпечить оперативний контроль за інформаційними потоками, але й сприятиме

швидкому реагуванню на потенційні загрози, мінімізуючи ризики витоку даних та підвищуючи загальний рівень безпеки організації (рис. 2.6).

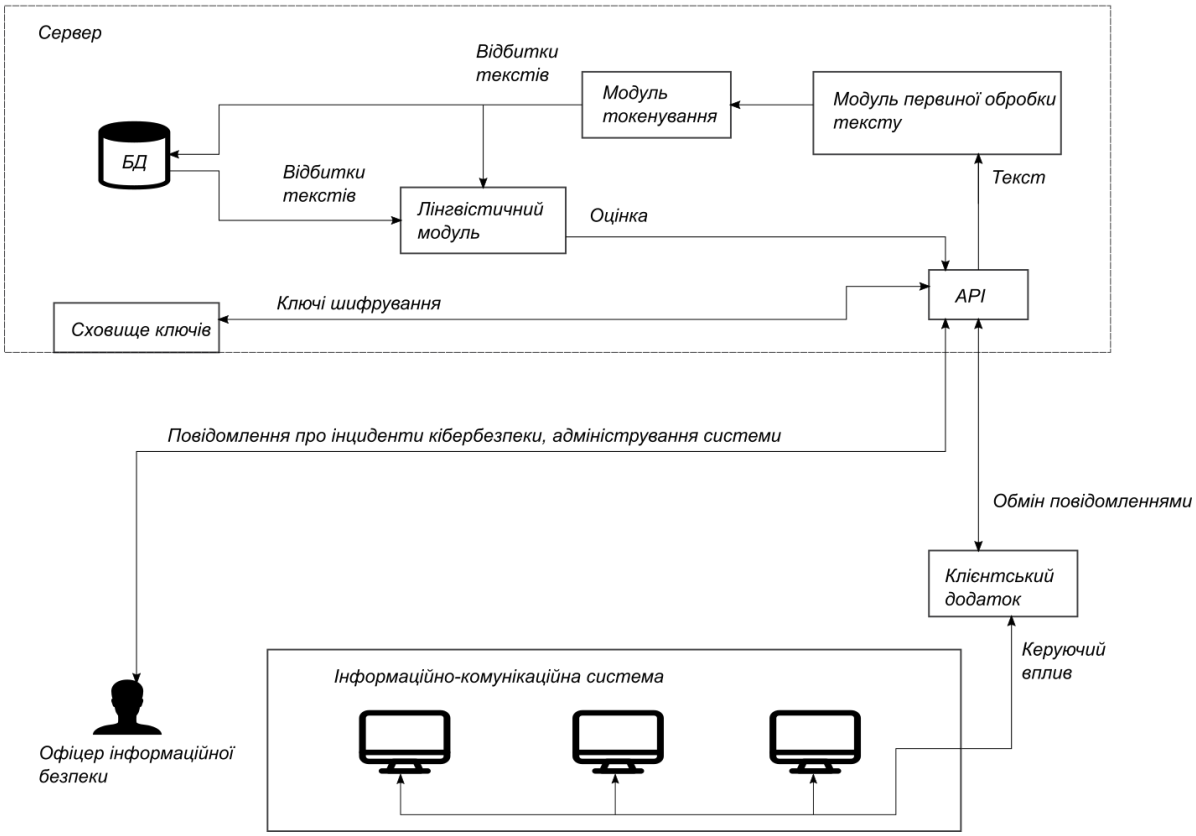


Рисунок 2.6 – Архітектура системи запобігання витоку даних

Серед внутрішніх компонентів сервера виділяються кілька ключових елементів, кожен із яких виконує важливу функцію у загальній архітектурі системи захисту. База даних є ядром для зберігання цифрових «відбитків» конфіденційної інформації, а також даних про користувачів системи, що дозволяє централізовано керувати інформацією. Сховище ключів функціонує як захищений компонент, що відповідає за зберігання та управління криптографічними ключами. Для забезпечення безпеки реалізовано спеціалізований інтерфейс, який виключає можливість несанкціонованого доступу до ключів.

Лінгвістичний модуль займається аналізом цифрових відбитків текстів, збережених у базі даних, визначаючи їх рівень конфіденційності. Він інтегрує

алгоритми обробки природної мови, щоб забезпечити точність та ефективність виявлення потенційних витоків інформації. Модуль первинної обробки тексту готує дані для подальшого аналізу, видаляючи зайві символи, очищуючи текст і нормалізуючи його. Це забезпечує уніфікованість та підвищує якість подальших процесів.

Модуль токенування створює цифрові «відбитки» текстів, дозволяючи ефективно ідентифікувати та порівнювати інформацію. Це ключовий елемент для виявлення потенційних витоків і забезпечення цілісності даних. API виступає в якості універсального інтерфейсу, через який взаємодіють усі компоненти системи, забезпечуючи зручність інтеграції та масштабованість.

Окрім серверних компонентів, важливо врахувати архітектуру клієнтського додатка. Він не лише виконує моніторинг потоків даних на робочих станціях, але й інтегрує механізми шифрування та дешифрування файлів. Додаток відстежує зовнішні підключення, аналізує активність користувачів і забезпечує автоматичне надсилання даних на сервер для подальшого аналізу. Крім того, клієнтський додаток може блокувати несанкціоновану передачу інформації на основі отриманих від сервера інструкцій, реалізовуючи превентивні заходи захисту.

Узагальнюючи, система розробляється як єдиний комплекс взаємодіючих компонентів, здатний забезпечити високий рівень інформаційної безпеки та оперативно реагувати на інциденти, пов’язані з витоком конфіденційних даних.

На рисунку 2.7 зображено архітектуру клієнтського додатку, який є важливим елементом системи захисту конфіденційної інформації. До його складу входить кілька функціональних модулів. Зокрема, Watcher відповідає за моніторинг файлової системи, ідентифікуючи неправомірні дії з файлами, що містять конфіденційну інформацію, а також аналізує інші потенційні канали витоку даних. CryptoAPI виконує криптографічну обробку файлів, зокрема їх шифрування та дешифрування, використовуючи ключі, отримані від сервера. Модуль керуючого впливу (МКВ) реалізує механізми управління потоками даних на робочій станції відповідно до команд сервера.

Схема також відображає взаємодію робочої станції, де забезпечується захист інформації, із сервером, який виконує централізований контроль за станом клієнтського додатку. Сервер не лише відслідковує активність клієнта, але й надає команди для управління потоками даних, які виконуються модулем керуючого впливу. Така структура забезпечує як локальний, так і централізований контроль за діями на робочій станції, значно підвищуючи рівень інформаційної безпеки.

Додатково, клієнтський додаток інтегрується з іншими компонентами системи, забезпечуючи гармонійну роботу в загальному інформаційно-комунікаційному середовищі. Важливим аспектом є його здатність адаптуватися до змін у конфігурації серверу та реалізовувати нові політики захисту, гарантуючи актуальність і надійність у боротьбі з витоками інформації.

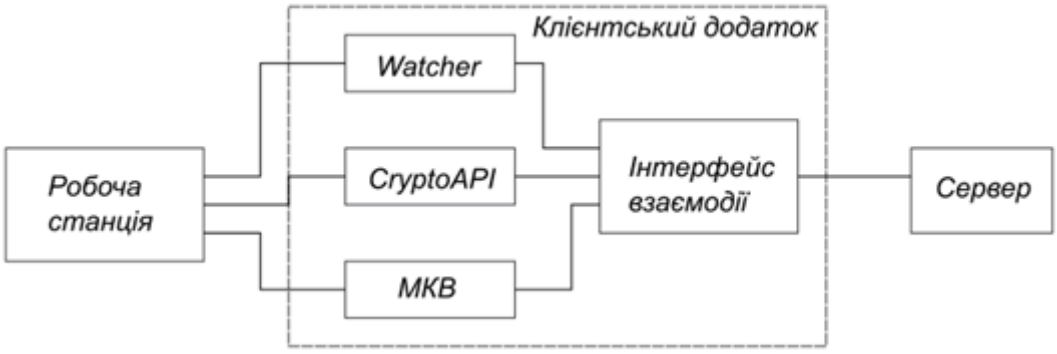


Рисунок 2.7 - Структура клієнтського додатку

Головним недоліком представленої архітектури є необхідність пересилання текстів, що аналізуються, через мережеві канали зв’язку. Це створює підвищене навантаження на інформаційно-комунікаційну мережу, а також додає ризик перехоплення конфіденційних даних у процесі передачі. Для вирішення цієї проблеми можна застосувати два підходи.

Перший підхід полягає у шифруванні текстів перед їх передачею на сервер для аналізу. Це дозволить забезпечити конфіденційність переданих даних і знизити ризик їх перехоплення. Однак, такий метод має свої недоліки. Зокрема, шифрування створює додаткове навантаження як на клієнт, так і на сервер, що

може негативно вплинути на продуктивність системи, особливо у випадках великої кількості одночасних запитів. Крім того, цей підхід не вирішує проблему збільшеного навантаження на мережеве середовище.

Другий підхід передбачає перенесення модулів токенування та первинної обробки тексту з сервера на клієнт. Це дозволяє знизити навантаження як на сервер, так і на мережу, оскільки передача текстів замінюється передачею множини токенів. Передача лише токенів значно зменшує обсяг передаваних даних, що частково вирішує проблему перевантаження мережевого середовища. Крім того, у цьому випадку ризик перехоплення конфіденційної інформації знижується, оскільки передані токени самі по собі не містять повного змісту тексту.

Кожен із підходів має свої переваги та обмеження, тому вибір залежить від конкретних вимог до системи, доступних ресурсів і допустимого рівня ризику. У деяких випадках можливе навіть комбінування обох підходів для досягнення оптимального балансу між безпекою, продуктивністю і ефективністю використання мережевих ресурсів.

Для вирішення зазначеної проблематики був обраний другий підхід, який передбачає перенесення модулів токенування та первинної обробки тексту з сервера на клієнтські додатки. Такий підхід дозволяє знизити навантаження на сервер і мережу, оскільки передаються лише токени замість повних текстів, що значно зменшує обсяг передаваних даних. Також він сприяє підвищенню рівня безпеки, адже токени, навіть у разі перехоплення, не містять критичної інформації в доступному для розуміння вигляді (рис. 2.8).

З урахуванням обраного підходу архітектура системи захисту від витоку даних набуває наступного вигляду. Ключову роль у системі відіграє клієнтський додаток, який інтегрується з робочими станціями користувачів. Він включає в себе модулі первинної обробки тексту і токенування, що забезпечує підготовку даних для подальшого аналізу. На рівні клієнта дані очищуються від незначущих

символів, нормалізуються та перетворюються у множину токенів. Тільки після цього вони передаються на сервер для оцінки.

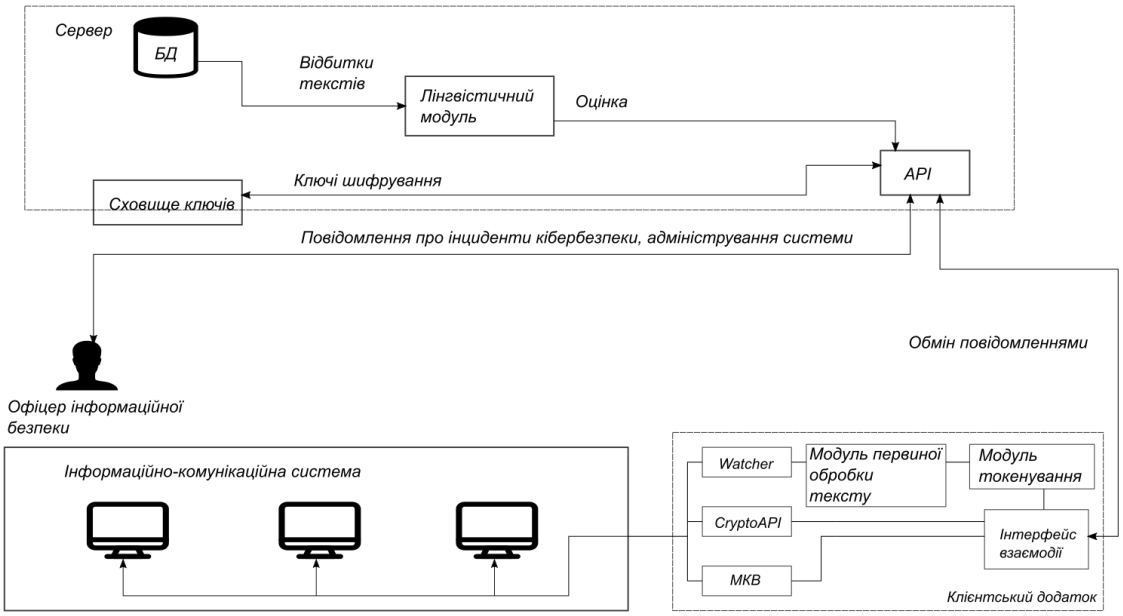


Рисунок 2.8 – Модифікована структура системи

Сервер продовжує виконувати функції центрального керування системою. Його завданням є прийом токенів від клієнтів, проведення лінгвістичного аналізу та надання результатів цього аналізу. Лінгвістичний модуль на сервері співставляє отримані токени з цифровими відбитками конфіденційної інформації, що зберігаються в базі даних, і оцінює, чи містять передані дані потенційно конфіденційну інформацію.

Крім того, сервер взаємодіє зі сховищем ключів криптографічного захисту для забезпечення шифрування і дешифрування файлів на клієнтському боці. Ключі надаються відповідно до політики безпеки та ролей користувачів у системі.

Така архітектура сприяє оптимізації ресурсів і підвищенню ефективності захисту, зберігаючи при цьому високий рівень безпеки та мінімізуючи ризики, пов’язані з передачею конфіденційної інформації через мережу.

2.3 Висновок

У процесі розробки системи захисту від витоку даних було обрано архітектуру, яка поєднує ефективність, безпеку та оптимальне використання ресурсів. Перенесення модулів первинної обробки тексту та токенування з сервера на клієнтські додатки дозволило знизити навантаження на інформаційно-комунікаційну мережу і серверну інфраструктуру. Такий підхід також мінімізує ризик перехоплення конфіденційної інформації, оскільки через мережу передаються лише токени, а не повні тексти.

Сервер залишається ядром системи, забезпечуючи централізоване керування, лінгвістичний аналіз і збереження цифрових відбитків конфіденційної інформації. Інтеграція зі сховищем криптографічних ключів дозволяє клієнтам виконувати шифрування та дешифрування даних, дотримуючись політики безпеки організації. Це забезпечує гнучкість у застосуванні системи до різних сценаріїв використання.

Результатом є багаторівнева система захисту, що ефективно запобігає витоку даних, адаптується до різних умов експлуатації та підвищує рівень інформаційної безпеки організації. Обрані рішення сприяють збереженню балансу між продуктивністю та безпекою, забезпечуючи при цьому високу надійність і масштабованість системи.

Серед внутрішніх компонентів сервера виділяються кілька ключових елементів, кожен з яких виконує свою функцію. База даних слугує сховищем цифрових «відбитків» конфіденційної інформації та даних про користувачів системи. Сховище ключів, як окремий елемент, зберігає криптографічні ключі та передбачає використання захищеного інтерфейсу для взаємодії. Лінгвістичний модуль аналізує цифрові відбитки текстів, збережених у базі даних, і на основі цього визначає рівень конфіденційності аналізованої інформації.

Окрім серверних компонентів, визначино архітектуру клієнтського додатка, який здійснюватиме моніторинг потоків даних на робочих станціях користувачів.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ СИСТЕМИ

3.1 Вибір інструментів

Для розробки системи еами обрано мову програмування C#. Програмування на C# базується на парадигмі об'єктно-орієнтованого програмування, що забезпечує ефективну реалізацію концепцій ООП, а також підтримує компонентно-орієнтований підхід [33, 34].

C# бере свої витоки з сімейства мов C, що відображається навіть у її назві. Проте ця мова значно відрізняється від своїх попередників завдяки численним інноваціям. Вони значно полегшують процес розробки програмного забезпечення, роблять його більш продуктивним та сприяють зручнішому налагодженню програм.

Ця мова підтримує об'єктно-орієнтоване програмування, що є ключовим підходом до створення масштабованих, структурованих та легких у підтримці програмних продуктів. Крім того, C# має багатий набір інструментів і бібліотек, що полегшують розробку, особливо у середовищі Microsoft .NET, яке забезпечує широку підтримку для створення серверних і клієнтських рішень [35].

Мова C# поєднує у собі переваги строгої типізації, що мінімізує ймовірність помилок, із зручним синтаксисом, який сприяє підвищенню продуктивності роботи розробників. Завдяки підтримці новітніх технологій, таких як асинхронне програмування, робота з мережею, обробка даних і безпечна пам’ять, C# є ідеальним вибором для створення складних систем захисту інформації.

Програми мовою C#, функціонують на платформі .NET, яка забезпечує роботу у віртуальному середовищі виконання, відомому як Common Language Runtime (CLR), та використовують набір бібліотек класів. CLR, розроблена компанією Microsoft, представляє собою загальну мовну інфраструктуру, яка стала міжнародним стандартом. Вона слугує базовою основою для створення середовища розробки та виконання, забезпечуючи стабільність і продуктивність програм [36].

Серед ключових переваг мови C# також виділяється можливість розробки на основі платформи .NET Core. Ця платформа створена на основі класичного .NET Framework, але має модульну архітектуру з відкритим вихідним кодом, що надає додаткові можливості для адаптації під потреби розробника. .NET Core є кросплатформенною, сумісною з більшістю сучасних операційних систем, включаючи Windows, macOS та Linux. Це дозволяє створювати універсальні додатки, які працюють на різних пристроях, зберігаючи високий рівень продуктивності та функціональності. Така архітектура сприяє підвищенню гнучкості, що робить мову C# і платформу .NET одним із провідних рішень у сучасній розробці програмного забезпечення.

Для зберігання даних, необхідних для функціонування розроблюваної системи, було обрано систему управління базами даних MS SQL Server. Ця СУБД є однією з найпопулярніших як у персональному, так і в корпоративному середовищі, завдяки своїй надійності, гнучкості та високій продуктивності. MS SQL Server належить до реляційних клієнт-серверних систем управління базами даних із розподіленою архітектурою, що забезпечує її широке використання в сучасних інформаційних системах [37].

Основними перевагами MS SQL Server є її надійність і високий рівень безпеки, зокрема завдяки вбудованим інструментам шифрування даних і засобам контролю доступу. Висока продуктивність системи дозволяє ефективно обробляти великі обсяги інформації, а зручність у використанні й адмініструванні робить її привабливим вибором як для початківців, так і для досвідчених спеціалістів.

Для роботи з даними в MS SQL Server використовується мова Transact-SQL (T-SQL), яка є спільною розробкою Microsoft та Sybase. Ця мова розширює стандарт SQL додатковими функціями для виконання складних запитів, обробки даних і програмування на рівні бази даних. У поєднанні з інструментами, такими як SQL Server Management Studio, T-SQL забезпечує гнучкість у створенні,

управлінні й оптимізації баз даних, що відповідає високим вимогам сучасних інформаційних систем [38].

Для взаємодії з MS SQL Server на вебсервері використовується об’єктно-реляційна технологія відображення (ORM) – Entity Framework Core. Ця технологія дозволяє абстрагуватися від низькорівневих аспектів структури даних і працювати з таблицями бази даних як з об'єктами, що полегшує процес розробки. Завдяки Entity Framework Core, замість роботи з фізичними елементами бази даних, такими як таблиці, індекси, первинні й вторинні ключі, програмісти можуть оперувати з об'єктами на концептуальному рівні, що значно спрощує роботу з даними в додатку [39].

Основною перевагою вибору Entity Framework Core є можливість використання LINQ-запитів для ефективної вибірки даних з бази даних. LINQ дозволяє створювати потужні й гнучкі запити для вибірки об’єктів, навіть якщо ці об'єкти пов'язані складними реляційними зв’язками. Цей підхід також дає можливість оптимізувати запити, що покращує продуктивність додатку, оскільки LINQ надає інструменти для компіляції запитів без необхідності писати складний SQL-код вручну. Використання Entity Framework Core допомагає зберігати код чистим і підтримуваним, одночасно забезпечуючи ефективне управління даними та зменшення кількості помилок, що можуть виникати при прямому використанні SQL-запитів.

Вебсервер є програмним забезпеченням, яке призначене для обробки HTTP-запитів і надання відповідей на них. Він виступає посередником між клієнтами та сервером, виконуючи функції збереження, обробки та передавання даних. Клієнтами вебсерверу можуть бути різноманітні пристрої, такі як веб-браузери, інші вебсервери, програми або будь-які пристрої, що потребують доступу до ресурсів серверу через Інтернет [40].

Для створення вебсерверу, який буде виконувати управління системою захисту від витоків даних і слугуватиме його ядром, я обрав платформу ASP.NET Core від компанії Майкрософт. Ця платформа створена для розробки веб-додатків

різних рівнів складності: від простих вебсторінок до складних систем з багатою логікою. ASP.NET Core працює на основі кросплатформного середовища .NET Core, що дає змогу розгортати додатки на найбільш популярних операційних системах, таких як Windows, Mac OS і Linux. Це дозволяє розробляти додатки, які можуть функціонувати на різних платформах, що розширює їхню доступність та сумісність. Хоча традиційно для розробки та розгортання додатків на ASP.NET Core використовуються операційні системи Windows, платформа більше не обмежує нас лише цією ОС. Тепер додатки можуть бути запущені на Linux та Mac OS. Для розгортання веб-додатків можна використовувати не лише традиційний вебсервер IIS, але й кросплатформний сервер Kestrel, що забезпечує ще більшу гнучкість і можливості для масштабування [41].

Для створення користувацького інтерфейсу системи керування була обрана архітектура модель-представлення-контролер (MVC) [42]. Цей архітектурний шаблон проектування дозволяє розподіляти додаток на три основні компоненти: модель даних, представлення та контролер.

Модель відповідає за управління даними та бізнес-логікою системи. Вона містить структури даних, правила валідації, операції з базою даних та алгоритми обробки інформації. Модель забезпечує незалежність даних від способу їх відображення та взаємодії з користувачем, що дозволяє використовувати одні й ті самі дані для різних типів інтерфейсів.

Представлення (View) реалізує візуальний інтерфейс системи та відповідає за відображення даних користувачу. Цей компонент включає форми введення, таблиці, графіки, повідомлення та інші елементи інтерфейсу. Представлення отримує дані від моделі та форматує їх для зручного сприйняття користувачем, при цьому не містить логіки обробки даних.

Контролер виступає посередником між моделлю та представленням, обробляючи користувацькі дії та координуючи взаємодію між компонентами. Він приймає команди від користувача через інтерфейс, передає їх моделі для обробки,

а потім оновлює представлення відповідно до результатів операцій. Контролер також забезпечує навігацію по системі та управління станом додатку.

Така структура сприяє відокремленню логіки обробки даних від візуальної частини інтерфейсу, що дає змогу значно зменшити взаємозалежність між компонентами. Це, в свою чергу, дозволяє легко вносити зміни в один з елементів системи, не впливаючи на інші частини додатку. Завдяки такому підходу, зокрема, забезпечується гнучкість у зміні представлення користувацького інтерфейсу без необхідності редагування основної бізнес-логіки, що робить систему більш масштабованою та зручною для подальших оновлень і вдосконалень (рис. 3.1).

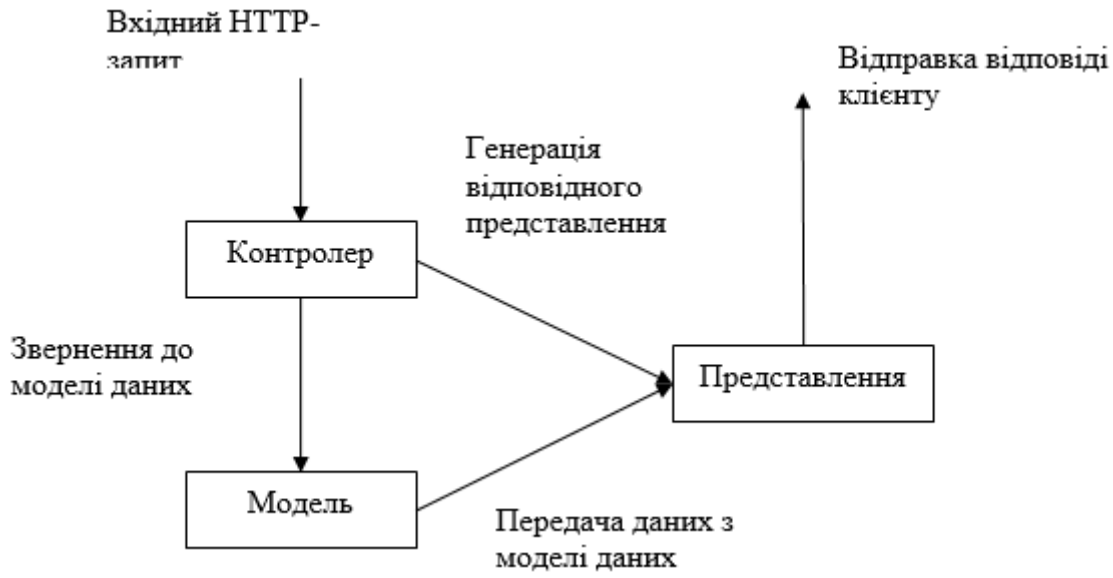


Рисунок 3.1 - Шаблон MVC [42]

Такий поділ компонентів додатку (рисунок 2.1) сприяє реалізації концепції розподіленої відповідальності, де кожен компонент має чітко визначену роль. Це дозволяє спростити розробку, тестування та підтримку окремих елементів системи, оскільки зміни в одній частині системи не потребують масштабних змін в інших. Модель даних виступає повністю незалежним компонентом, в той час як контролер і представлення залишаються відносно автономними. Зміни в контролері чи представленні, як правило, не впливають на модель, але будь-які

зміни в моделі зазвичай вимагають коригування інших компонентів, щоб забезпечити консистентність усієї системи.

Для передачі повідомлень між клієнтами та вебсервером у реальному часі в системі використовується технологія SignalR. Для реалізації цієї технології на платформі .NET компанія Майкрософт створила бібліотеку SignalR Core. Це потужне рішення для розробки додатків, які потребують обміну даними в режимі реального часу, що є основною вимогою для нашої системи. SignalR дозволяє встановити двонаправлений зв'язок між клієнтом і сервером, забезпечуючи миттєву передачу даних і команд. В основі цієї технології лежить концепція хабу, який виступає як точка взаємодії для підключення клієнтів, де обробляється вся логіка повідомлень та обміну даними між клієнтами і сервером, відповідно до визначених бізнес-правил.

Для розробки програмних компонентів системи було обрано інтегроване середовище розробки Visual Studio Community [43]. Це потужний інструмент, який надає розширений функціонал для редагування, налагоджування та створення додатків.

Цей набір інструментів дозволяє суттєво підвищити ефективність розробки, зменшити час на відлагодження та тестування, а також знизити ймовірність помилок. Завдяки попередньому досвіду роботи з цим середовищем розробки, вибір Visual Studio Community був природним і обґрунтованим для реалізації даної кваліфікаційної роботи.

3.2 Розробка алгоритмів функціонування системи

На основі розробленої архітектури системи захисту від витоків даних, розпочинається реалізація алгоритмів функціонування її компонентів. Алгоритм, представлений на рисунку 3.2, описує процес обробки запитів клієнтів до серверу. Першим етапом є розгортання необхідних елементів, серед яких контекст даних,

бібліотека Entity Framework Core, що підключена до бази даних під управлінням MS SQL Server, а також контролери та хаби для підключення через технологію SignalR.

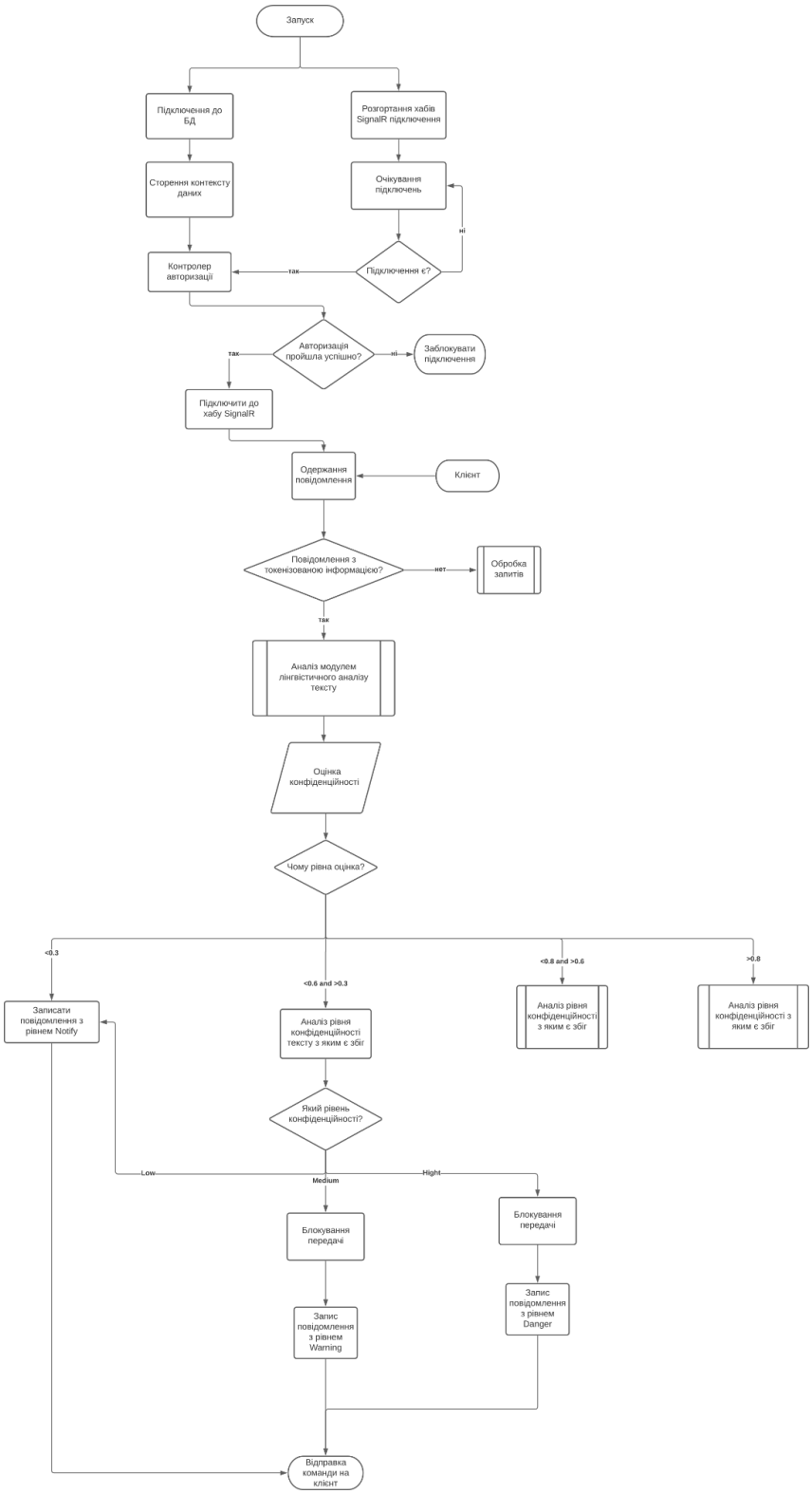


Рисунок 3.2 - Алгоритм перевірки запитів на конфіденційність інформації

Розгортання сервісів здійснюється при запуску сервера, і це відбувається один раз для кожного сервісу, паралельно. Після успішного розгортання сервісів хаб SignalR чекає на підключення клієнтів. Як тільки клієнт підключається, його запит перенаправляється до контролера авторизації. Після успішної авторизації клієнту надається токен, який підтверджує його автентичність і необхідно передавати при кожному подальшому запиті до системи. У разі невдалої авторизації підключення блокується на певний час. Після отримання токену клієнт і сервер утримують зв'язок через двонаправлений канал, по якому відбувається передача повідомлень у реальному часі за допомогою технології SignalR. Сервер має можливість детально налаштовувати параметри підключень, забезпечуючи належний рівень безпеки та конфіденційності даних.

Час утримання активного підключення клієнта в системі визначається за допомогою механізму моніторингу бездіяльності, що дозволяє автоматично завершувати підключення при відсутності активних запитів протягом визначеного часу. Це дозволяє забезпечити безпеку та оптимізувати використання ресурсів сервера. Водночас система підтримує схему роботи автентифікації та авторизації, яка дозволяє гарантовано перевіряти достовірність користувачів на кожному етапі взаємодії з сервером.

У разі отримання повідомлення від клієнта, система передає його на обробку до відповідного обробника подій. Якщо повідомлення містить токенізовану інформацію, воно підлягає перевірці на предмет можливого витоку даних. Це здійснюється через порівняння цифрових "відбитків" тексту з тими, що зберігаються в базі даних. Лексичний аналіз оцінює рівень конфіденційності документа, що дозволяє визначити, чи є дане повідомлення потенційно небезпечним. Оцінка конфіденційності результатує у наданні рішення: чи блокувати передачу інформації або дозволити її. Водночас, у базу даних заноситься запис з рівнем повідомлення (Notify, Warning або Danger), який залежить від результатів оцінки, а також інформацією про клієнта, змістом

повідомлення, часом і датою інциденту. Після завершення обробки цього повідомлення система переходить до обробки наступного запиту або чекає на нові повідомлення, забезпечуючи безперервний моніторинг і захист від витоків даних.

Процес авторизації клієнтів у системі захисту від витоків даних розпочинається із запиту клієнта до контролера, що є частиною архітектури MVC. Клієнт передає свій логін і пароль, які контролер обробляє, звертаючись до бази даних для перевірки наявності відповідних даних. Контролер використовує контекст даних для доступу до бази, де за допомогою відповідних запитів витягуються логіни користувачів та хешовані паролі.

Після того, як система перевіряє правильність введених даних, використовуючи алгоритм порівняння хешів паролів, контролер генерує JSON Web Token (JWT) за допомогою спеціального генератора токенів [44]. Цей токен є тимчасовим і містить зашифровану інформацію про авторизованого користувача, що дозволяє йому отримати доступ до подальших запитів і операцій без необхідності повторної авторизації. Для цього використовуються стандарти JWT Bearer, що забезпечують безпечну передачу та перевірку автентичності користувача через HTTP заголовки.

JWT токен, що отримується після успішної авторизації, передається клієнту та використовується для всіх подальших запитів до сервера. Якщо авторизація не вдалася, система повертає помилку, і клієнт не зможе отримати доступ до захищених ресурсів.

Після того як клієнт надає свої логін та пароль, система з допомогою ASP.NET Core Identity обчислює хеш введеного тексту та порівнює його з хешем, збереженим в базі даних. Якщо хеші збігаються, це підтверджує правильність введених даних, і клієнт проходить етап авторизації (рис. 3.3).

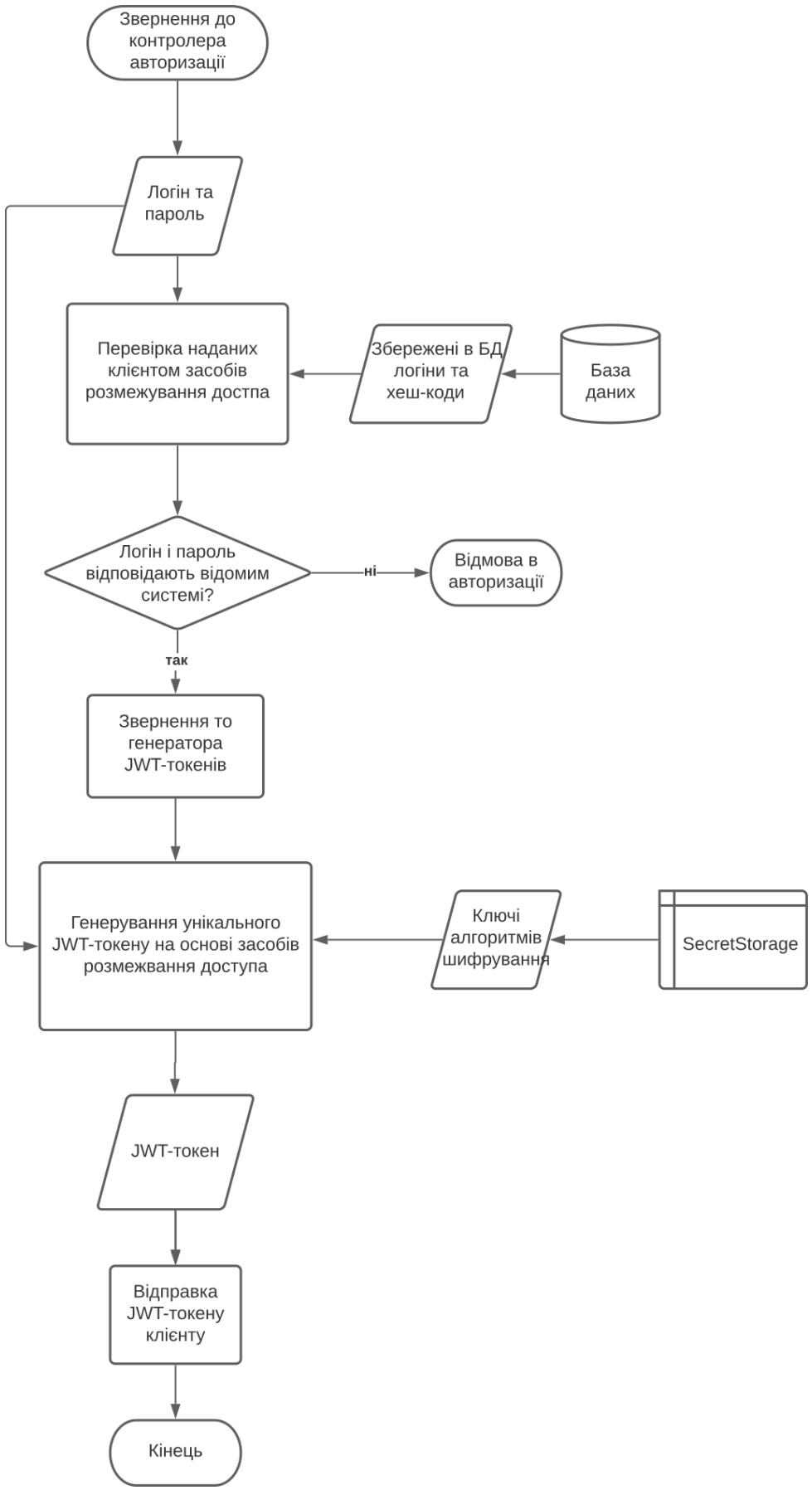


Рисунок 3.3 – Блок-схема авторизації

Наступним кроком є звернення до сервісу генерації JWT токенів. В цьому етапі використовуються ключі шифрування, що зберігаються у захищеному секретному сховищі. Ці ключі застосовуються для симетричного шифрування, що дозволяє безпечно генерувати токен. На основі отриманого ключа шифрування, логіну користувача та іншої інформації, система виконує хешування і шифрування з використанням симетричного ключа. У результаті генерується JWT токен, що підтверджує авторизацію клієнта, і цей токен відправляється клієнту.

JWT токен має тимчасовий характер і слугує підтвердженням того, що клієнт успішно пройшов авторизацію. Відправлений токен може бути використаний для доступу до захищених ресурсів системи протягом встановленого часу, після чого клієнт повинен пройти повторну авторизацію для отримання нового токену.

Після ініціалізації модуля лексичного аналізу як сервісу в середовищі ASP.NET Core, система переходить до підготовки до отримання і обробки запитів. В цей момент запускаються два паралельних процеси. Перший з них передбачає створення черги для обробки текстів, що підлягають аналізу на можливий витік інформації. Другий процес займається обробкою запитів, що надходять зовні до модуля лексичного аналізу.

Коли надійде новий запит на аналіз, система записує текст, що підлягає перевірці, разом з додатковою інформацією про клієнта, який ініціював запит, до черги для подальшої обробки. Цей процес забезпечує ефективне управління запитами і дозволяє мати контроль над черговістю аналізу текстів, а також гарантує, що кожен запит буде правильно оброблений. Після цього текст буде передано для подальшого лексичного аналізу, в ході якого здійснюється перевірка на наявність конфіденційної інформації, що може вказувати на можливий витік даних (рис. 3.4).

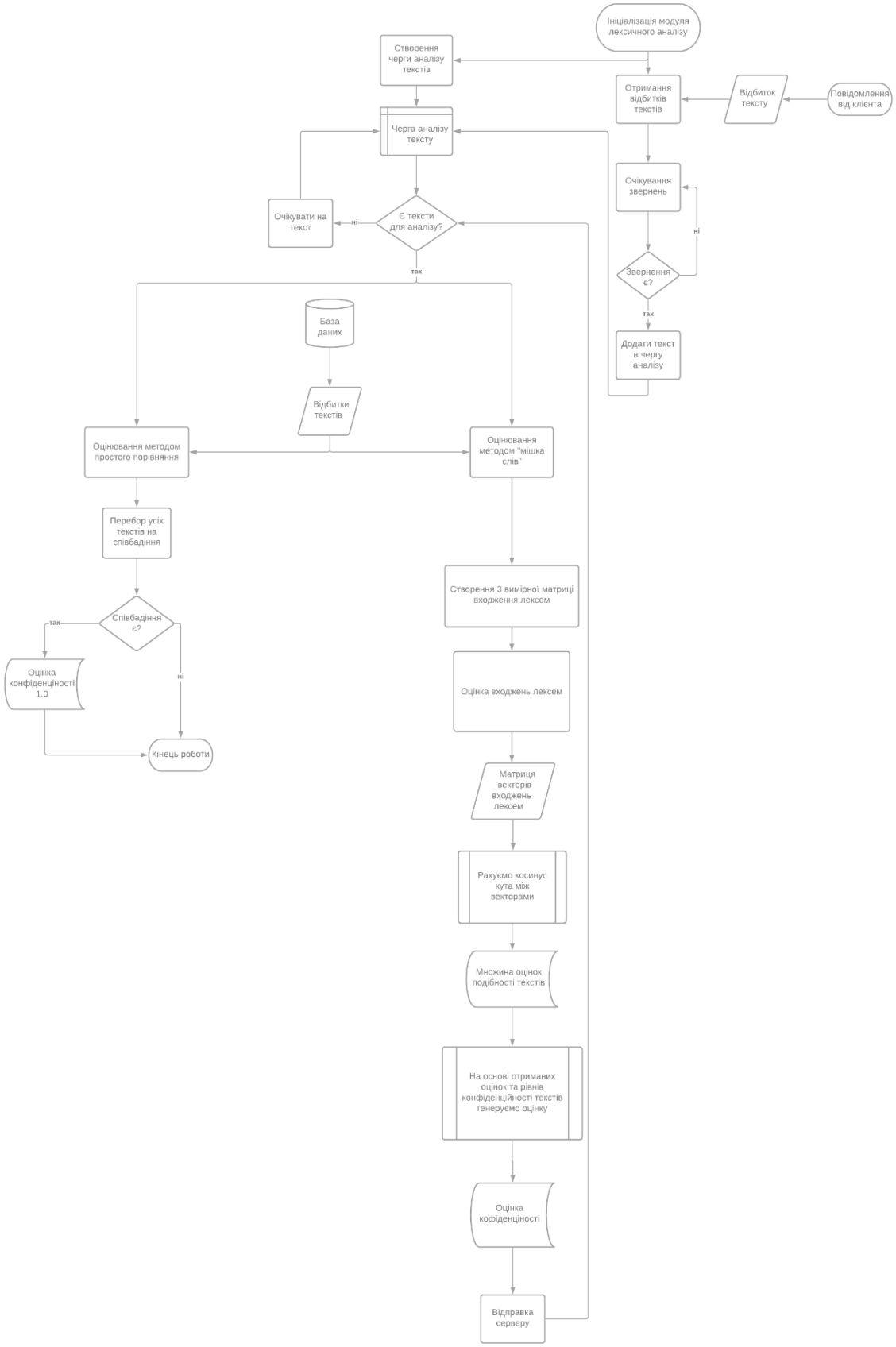


Рисунок 3.4 – Блок-схема модуля лексичного аналізу тексту

Основна мета черги – це впорядкування запитів на аналіз текстів і оптимізація роботи модуля лексичного аналізу. Якщо в черзі знаходиться неопрацьований відбиток тексту, ініціалізується процес оцінювання рівня конфіденційності, що виконується за допомогою двох методів. Перший метод – це просте порівняння, при якому здійснюється перебір відбитків текстів, збережених в базі даних як конфіденційні, з відбитком аналізованого тексту. Другий метод – це метод «мішка слів», що передбачає створення матриці, де рядки представляють лексеми (токени) аналізованого тексту, а стовпці – це токени збережених текстів. У кожній клітинці матриці вказується кількість входжень лексеми в обидва тексти.

На основі цієї матриці формується векторне подання текстів у багатовимірному просторі. Після цього розраховується скалярний добуток між векторами аналізованого відбитку і збереженими векторами. Це дозволяє оцінити подібність текстів і рівень конфіденційності аналізованого тексту. Як результат, система отримує числову оцінку, що відображає рівень схожості текстів, що дозволяє визначити ймовірний рівень конфіденційності. Наступним етапом є вибір тексту з найбільшою оцінкою, що надається тексту з найвищим рівнем конфіденційності.

Розроблена система э більш надійною та масштабованою завдяки використанню асинхронного програмування та потокобезпечного логування.

Основні відмінності та переваги цієї реалізації включають асинхронну обробку даних через async/await, що дозволяє системі ефективно працювати з великими обсягами даних не блокуючи основний потік виконання. Додано потокобезпечне логування через механізм lock, що запобігає конфліктам при одночасному записі від різних потоків (рис. 3.5).

```
using System;
using System.Collections.Generic;
using System.Text.RegularExpressions;
using System.Security.Cryptography;
using System.Text;
using System.Threading.Tasks;
using System.IO;

namespace DLPSystem
{
    public class DLPConfiguration
    {
        public Dictionary<string, string> Rules { get; set; }
        public List<string> SensitiveKeywords { get; set; }
        public string LogFilePath { get; set; }
    }
```

Рисунок 3.5 – Використані бібліотеки

Система використовує строго типізовані класи для конфігурації та результатів сканування, що робить код більш надійним та легшим для підтримки. Додано можливість розширення функціоналу через конфігурацію при ініціалізації.

Реалізовано ефективне хешування чутливих даних використовуючи SHA256, що забезпечує надійний захист конфіденційної інформації. Система підтримує метадані для кожного сканування, що дозволяє краще відстежувати та аналізувати інциденти.

Основні модулі обробки системи включають лінгвістичний модуль, який відповідає за аналіз текстів, модуль первинної обробки для нормалізації даних та модуль токенування для створення цифрових відбитків. Лінгвістичний модуль здійснює глибокий аналіз структури тексту, виявляючи ключові лексеми та фрази, що можуть бути ознаками конфіденційної інформації. Модуль первинної обробки виконує попередню обробку даних, зокрема очищення від зайвих символів і приведення тексту до стандартного формату, що забезпечує більш точну подальшу обробку. Модуль токенування здійснює перетворення тексту на

цифрові відбитки у вигляді токенів, які можуть бути порівняні з відбитками з бази даних для виявлення потенційно небезпечної інформації.

Ці модулі працюють послідовно, забезпечуючи точний та ефективний аналіз текстів для виявлення конфіденційної інформації. Кожен етап доповнює попередній, забезпечуючи високий рівень точності та ефективності в процесі виявлення витоків даних.

На рисунку 3.6 наведено приклад формування словника правил та набору ключових слів чутливої інформації.

```
public DLPSystem(DLPConfiguration config)
{
    _rules = config.Rules ?? new Dictionary<string, string>
    {
        { "credit_card", @"\b\d{4}[-\s]?\d{4}[-\s]?\d{4}[-\s]?\d{4}\b" },
        { "email", @"\b[A-Za-z0-9._%+-]+@[A-Za-z0-9.-]+\.[A-Z|a-z]{2,}\b" },
        { "phone_ua", @"\b(?:\+38)?\s?0\d{2}[\s-]?\d{3}[\s-]?\d{2}[\s-]?\d{2}\b"
        { "passport_ua", @"\b[А-ЩЬЮЯҐЄІЇ]{2}\d{6}\b" }
    };

    _sensitiveKeywords = config.SensitiveKeywords ?? new List<string>
    {
        "конфіденційно",
        "службова інформація",
        "комерційна таємниця",
        "внутрішнє використання",
        "не для поширення"
    };
```

Рисунок 3.6 – Формування словника правил

3.3 Тестування системи

Для проведення тестування системи була налаштована мережа з кількох віртуальних машин. На цих машинах було встановлено клієнтське програмне

забезпечення системи захисту від витоку даних, а також додаток, який імітував діяльність користувача, виконуючи операції з переміщення та копіювання текстів і файлів. Віртуальна машина, яка працювала під управлінням Linux Ubuntu, виконувала роль сервера, на якому був розгорнутий контейнер Docker, що містив сервер керування системою.

Додатки, що імітували діяльність користувача, використовували спільне сховище для документів, які брали участь в операціях з копіювання та переміщення файлів. Така конфігурація дозволила провести тестування реальних сценаріїв роботи системи та перевірити її здатність ефективно виявляти й блокувати можливі витоки даних.

На рисунку 3.7 представлений графік, що показує кількість запитів та фактичних витоків даних, які були скоєні під час тестування. Початкове значення кількості операцій становило 1000 на годину, після чого поступово зменшувалось із кроком 100 операцій. Кількість витоків була задана випадковим чином оператором тестування, щоб оцінити здатність системи виявляти і реагувати на різні сценарії витоків даних під час навантаження.

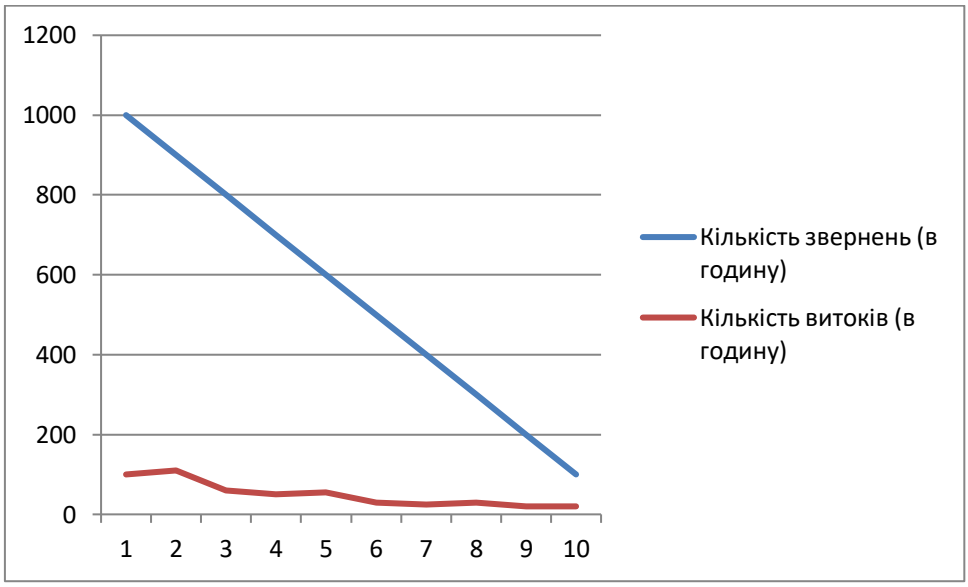


Рисунок 3.7 – Графік результатів тестування

Графік успішно заблокованих витоків даних на рисунку 3.8 демонструє ефективність системи захисту в процесі тестування. Кількість успішно виявлених витоків збільшується із збільшенням числа запитів та витоків, що були скоєні. Блокування витоків здійснюється автоматично після їх виявлення, і цей процес також зображено на графіку. Під час тестування система продемонструвала високу точність у виявленні витоків, а блокування відбулося у відповідності до встановлених порогових значень конфіденційності та рівня ризику.

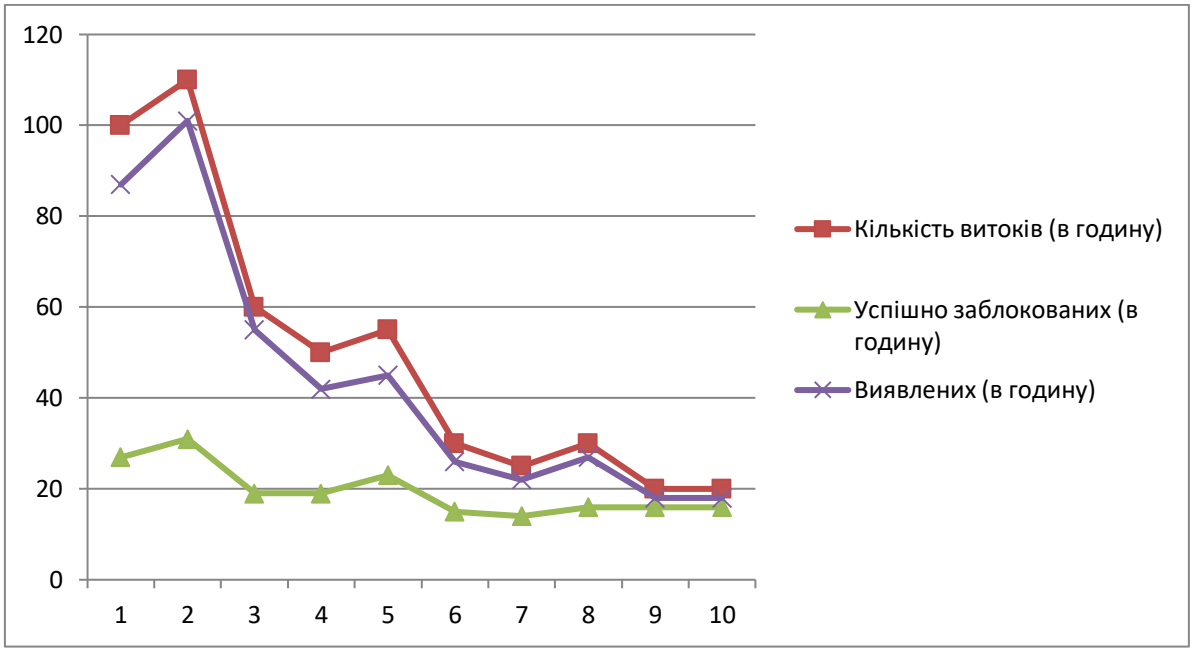


Рисунок 3.8 – Графік успішно заблокованих витоків

Як видно з рисунків 3.8 кількість виявлених витоків є досить високою, досягаючи близько 91%. Однак кількість успішних блокувань значно залежить від обсягу запитів. Це свідчить про те, що ефективність блокувань критично залежить від часу, необхідного для обробки кожного запиту. Причиною такої залежності є архітектура модуля лінгвістичного аналізу тексту, де використовується черга запитів. Така організація роботи призводить до того, що модуль не встигає завершити аналіз тексту до того, як операції з конфіденційною інформацією вже будуть виконані.

Цей підхід, хоча і дозволяє впорядкувати обробку запитів, значно знижує швидкість реакції на потенційні витоки даних. Крім того, варто зазначити, що тестування проводилось на одному фізичному сервері, що також могло вплинути на результати, зокрема на швидкість обробки запитів. Дане тестування виявило проблеми в архітектурі модуля лексичного аналізу, що вимагає оптимізації. Для підвищення ефективності роботи системи необхідно звернути особливу увагу на потужності серверного обладнання, а також удосконалити метод оцінки рівня конфіденційності. Одним із можливих рішень є розпаралелювання процесів аналізу та блокування для зменшення часу реакції.

3.4 Висновок

У цьому розділі ми вибрали найбільш ефективні інструменти для розробки нашого проекту, що дозволяє забезпечити швидкість, надійність і високий рівень ефективності. Використання сучасних технологій не лише сприяє оптимізації процесу розробки, а й забезпечує безпеку даних, легкість у їх управлінні та зручність при розгортанні проекту на різних платформах. Інтеграція передових рішень дозволяє досягти високої якості та продуктивності системи, гарантуючи її стабільність та масштабованість в умовах зростаючих вимог.

Програмна реалізація системи захисту від витоків даних показала високий рівень виявлення витоків, зокрема до 90%. Однак ефективність блокування інформації виявилася залежною від кількості запитів, що зумовлено архітектурою модуля лексичного аналізу та чергою запитів. Час обробки кожного запиту обмежує швидкість реагування на витоки, що знижує ефективність блокування при великих навантаженнях.

Додатки, які імітували діяльність користувача, використовували спільне сховище для документів, що брали участь у процесах копіювання та переміщення файлів. Така організація дозволила провести тестування реальних сценаріїв

роботи системи, оцінити її здатність ефективно виявляти та блокувати потенційні витоки даних.

Для покращення результатів потрібно вдосконалити методи аналізу тексту, підвищити потужність серверів та застосувати розпаралелювання процесів аналізу для зменшення часу обробки запитів. Удосконалення архітектури модуля лексичного аналізу та оптимізація роботи системи дозволить підвищити ефективність блокування витоків та покращити загальну продуктивність системи.

ВИСНОВКИ

Дана кваліфікаційна робота доводить важливість і перспективність розробок пов’язаних із забезпеченням безпеки в кіберпросторі. Така тенденція пов’язана з впровадженням інформаційних технологій у сучасне суспільство. Під час виконання кваліфікаційної роботи було проведено глибоке дослідження проблем у сфері інформаційної безпеки та існуючих загроз. У роботі розглянуті основні проблеми безпеки в кіберпросторі, а також заходи, спрямовані на захист від потенційних загроз. Зокрема, акцент було зроблено на витоках конфіденційної інформації, що є наслідком дій внутрішніх порушників системи – осіб, що мають доступ до організаційних ресурсів.

Згідно з аналізом статистики, більшість інцидентів у сфері безпеки спричиняють не зовнішні хакери, а співробітники, які вже мають авторизований доступ до системи. Така ситуація вимагає перегляду існуючих підходів до забезпечення безпеки та розробки нових рішень для захисту як від зовнішніх, так і від внутрішніх загроз.

У межах роботи була спроектована система захисту від витоків даних, що забезпечує превентивний захист конфіденційної інформації. На основі технічного завдання була розроблена та реалізована система за допомогою мови програмування C#, а також використання платформи ASP.Net Core для адміністрування та технології SignalR для передачі повідомлень у реальному часі.

Під час тестування розробленої системи захисту від витоків даних було виявлено недолік, пов'язаний із стійкістю до навантаження. Проте цей недолік не є критичним через обмежену сферу застосування системи. Однак це може стати основою для подальших наукових досліджень, зокрема для розробки нових методів аналізу витоків даних, які можна інтегрувати в існуючу систему, що дозволить значно підвищити ефективність виявлення конфіденційної інформації серед великої кількості даних, що циркулюють в інформаційно-комунікаційному середовищі.

Ці результати відкривають перспективи для подальших наукових досліджень та розробки нових методів і технологій, здатних підвищити ефективність виявлення конфіденційної інформації та забезпечити стійкість системи до великих навантажень, що в свою чергу може сприяти розвитку систем захисту від витоків даних у більш широкому контексті.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Data Loss Prevention: концепції та практичне застосування : навчальний посібник / за ред. М. А. Іванченко. – Харків : ХНУРЕ, 2023. – 156 с.

2. Бурячок В. Л. Основи інформаційної та кібернетичної безпеки: навчальний посібник / В. Л. Бурячок, Р. В. Киричок, П. М. Складанний – К., 2018. – 320 с.

3. Інформаційна безпека: навчальний посібник / [Ю. Я. Бобало, І. В. Горбатий, М. Д. Кіселичник, А. П. Бондарєв та ін.]; за заг. ред. д-ра техн. наук, проф. Ю. Я. Бобала та д-ра техн. наук, доц. І. В. Горбатого. – Львів: Видавництво Львівської політехніки, 2019. – 580 с.

4. Про захист інформації у інформаційно-комунікаційних системах : Закон України від 04.07.2020 р. №80/94 -ВР. URL: https://zakon.rada.gov.ua/laws/show/80/94-%D0%B2%D1%80#Text (дата звернення: 15.02.2025)

5. Інформаційна безпека держави: навчальний посібник/ В.І. Гур'єв, Д.Б. Мехед, Ю.М. Ткач, І.В. Фірсова. – Ніжин: ФОП Лук'яненко В.В. ТПК «Орхідея», 2018. – 166 с.

6. Kim D. Fundamentals of information systems security / David Kim, Michael G. Solomon. – Third edition. – Burlington :Jones & Bartlett Learning, 2018. – 571 p.

7. Загальний регламент про захист даних. URL: https://uk.wikipedia.org/wiki/ Загальний_регламент_про_захист_даних (дата звернення: 01.04.2025).

8. Дикий О. В. Стандарти інформаційної безпеки: компаративне дослідження / О. В. Дикий, М. О. Флюнт // Право та державне управлінняю – 2019. – № 2 (35), том 1. – С. 80-87

9. Оліфіренко, С. М. Побудова системи моніторингу витоків даних для малих та середніх підприємств : автореф. дис. ... канд. техн. наук : 05.13.06 / С. М. Оліфіренко ; Харк. нац. ун-т радіоелектроніки. – Харків, 2022. – 20 с.

10. ДСТУ ISO/IEC 27032:2016. Інформаційні технології. Методи захисту. Настанови щодо кібербезпеки. – Чинний від 2016-27-12. – Київ : ДП «УкрНДНЦ», 2018. – [50] с.

11. НД ТЗІ 3.7-001-99. Методичні вказівки щодо розробки технічного завдання на створення комплексної системи захисту інформації в автоматизованій системі. – Чинний від 28 квітня 1999 р. – Київ : ДСТСЗІ СБ, 1999. – [35] с.

12. Системи контролю доступу: що це таке і як працює. URL: https://zakarpattya.net.ua/News/200909-Systemy-kontroliu-dostupu-shcho-tse-take-i-iak-pratsiuie (дата звернення: 11.02.2025).

13. Cyber-Physical Security: Protecting Critical Infrastructure at the State and Local Level / Editors : Robert M. Clark, Simon Hakim. – Cham, Switzerland : Springer, 2016. – 360 p.

14. Сидоров, К. П. Аналіз ринку DLP-рішень в Україні та світі / К. П. Сидоров, Л. М. Ткаченко // Економіка і суспільство. – 2023. – № 48. URL: https://economyandsociety.in.ua/index.php/journal/article/view/2158 (дата звернення: 15.11.2025).

15. Fundamentals of InformationSystems Security / Editors : David Kim, Michael G. Solomon. – Burlington, Massachusetts : Jones & Bartlett Learning, 2018. – 548 p.

16. Baranov O.A. Internet of Things (IoT): A Review of Legal Issues // Internet of Things: Scientific Conference. October 24, 2017, Kyiv. / Order. VM Furashev, S. Yu. Petryayev. - K .: NTUU "Igor Sikorsky Kyiv Polytechnic Institute" Publishing House "Polytechnic". 2017

17. Інформаційна безпека держави: навчальний посібник/ В.І. Гур'єв, Д.Б. Мехед, Ю.М. Ткач, І.В. Фірсова. – Ніжин: ФОП Лук'яненко В.В. ТПК «Орхідея», 2018. – 166 с.

18. Про основні засади забезпечення кібербезпеки України : ЗаконУкраїни від 01.08.2021 р. №2163-VIII. URL: https://ips.ligazakon.net/document/JH1N268B (дата звернення: 23.04.2025)

19. Information Security Standard: Information Technology Resource Management. Virginia Information Technologies Agency (VITA), 2016. – 183 p.

20. Гребенніков В.В. Комплексні системи захисту інформації: проєктування, впровадження, супровід. / В.В. Гребенніков – Ужгород: Ужгородський національний університет, 2013. – 161 с.

21. Enterprise Data Loss Prevention: A Comprehensive Guide / T. Anderson, K. White, J. Brown. – California : O'Reilly Media, 2022. – 412 p.

22. Modern Approaches to Information Security: DLP Systems and Beyond / edited by A. Miller. – London : Academic Press, 2023. – 298 p.

23. Practical Information Security Management: A Complete Guide to Planning and Implementation/ Tony Campbell. – Australia: Burns Beach, 2016. – 253 p.

24. Defense Counterintelligence and Security Agency Assessment and Authorization Process Manual – Quantico, Virginia : Defense Counterintelligence and Security Agency, 2020. – 163 p. Кравець, О. Я. Технології виявлення аномалій у DLP-системах на основі машинного навчання / О. Я. Кравець, Д. О. Мельник, С. І. Гришко // Кібербезпека: освіта, наука, техніка. – 2022. – № 4 (16). – С. 23–35.

25. Лисенко, Н. В. Ефективність впровадження DLP-систем в українських компаніях / Н. В. Лисенко // Економіка та управління підприємствами. – 2023. – № 1. – С. 89–96.

26. Міщенко, В. А. Інтеграція DLP-систем з іншими засобами інформаційної безпеки / В. А. Міщенко, К. С. Поляков // Захист інформації. – 2022. – Т. 24, № 3. – С. 145–153.

27. Новіков, П. Г. Правові аспекти використання DLP-систем в Україні / П. Г. Новіков // Інформація і право. – 2023. – № 2 (45). – С. 67–74.

28. Al-shawi M. Designing for Cisco Network Service Architectures (ARCH) Foundation Learning Guide: CCDP ARCH 300-320 / Marwan Al-shawi, André Laurent. – Indianapolis : Cisco Press, 2017. – 941 p.

29. Cyber-Development, Cyber-Democracy and Cyber-Defense: Challenges, Opportunities and Implications for Theory, Policy and Practice / Elias G. Carayannis, David F. J. Campbell, Marios Panagiotis Efthymiopoulos. – New York : Springer, 2014. – 360 p.

30. Барановський, О. М. DLP-системи як основа захисту конфіденційної інформації підприємства / О. М. Барановський, С. В. Кравченко // Кібербезпека: освіта, наука, техніка. – 2022. – № 3 (15). – С. 45–56.

31. Васильєв, Д. П. Методи виявлення витоків даних у корпоративних мережах : монографія / Д. П. Васильєв. – Київ : НТУ «КПІ», 2023. – 284 с.

32. Головко, І. А. Впровадження систем запобігання витокам даних на підприємствах України / І. А. Головко, Т. М. Сергієнко // Науковий вісник Ужгородського національного університету. Серія: Міжнародні економічні відносини та світове господарство. – 2023. – Вип. 47. – С. 67–72.

33. Грищенко, В. О. Архітектура DLP-систем для захисту персональних даних / В. О. Грищенко // Інформаційні системи та мережі : збірник наукових праць. – Львів : Львівська політехніка, 2022. – Вип. 4. – С. 125–134.

34. Жуков, С. М. Порівняльний аналіз сучасних DLP-рішень / С. М. Жуков, О. В. Романенко // Вісник НТУ «ХПІ». Серія: Інформатика та моделювання. – 2022. – № 2 (8). – С. 78–89.

35. Іваненко, А. С. Методологія побудови систем захисту від витоків інформації : дис. ... канд. техн. наук : 05.13.21 / А. С. Іваненко ; НАН України, Ін-т проблем моделювання в енергетиці. – Київ, 2023. – 189 с.

36. Коваленко, Т. В. Класифікація загроз інформаційній безпеці в контексті DLP-систем / Т. В. Коваленко // Інформаційна безпека. – 2023. – № 2 (50). – С. 112–120.

37. Петренко, А. І. DLP-системи в хмарному середовищі: виклики та перспективи / А. І. Петренко, О. В. Сидоренко // Інформаційні технології та комп'ютерна інженерія. – 2023. – № 2 (57). – С. 45–52.

38. Романчук, В. С. Методи криптографічного захисту в DLP-рішеннях / В. С. Романчук // Безпека інформації. – 2022. – Т. 28, № 4. – С. 234–241.

39. Тимошенко, Д. В. Використання штучного інтелекту в DLP-системах / Д. В. Тимошенко // Штучний інтелект. – 2023. – № 2. – С. 98–107.

40. Федоренко, М. О. Моделювання процесів виявлення витоків інформації в корпоративних мережах / М. О. Федоренко, Ю. А. Шевченко // Моделювання та інформаційні технології. – 2022. – Вип. 95. – С. 67–75.

41. Чернецький, І. Г. DLP-системи: сучасний стан та перспективи розвитку : монографія / І. Г. Чернецький, А. В. Мельник. – Тернопіль : ТНТУ, 2023. – 168 с.

42. Янченко, О. С. Методи оцінки ефективності DLP-систем / О. С. Янченко, В. І. Коржик // Системи обробки інформації. – 2023. – № 2 (171). – С. 78–86.

43. Digital Forensics and Data Loss Prevention / edited by R. Johnson, M. Smith. – 2nd ed. – New York : Springer, 2023. – 345 p.

44. Douglas J. Landoll Information Security Policies, Procedures, and Standards: A Practitioner’s Reference / Douglas J. Landoll. – Boca Raton : CRC Press Taylor & Francis Group, 2016. – 246 p.

ДОДАТОК А

Програмний код

```
using Microsoft.AspNetCore.Identity.EntityFrameworkCore;
using Microsoft.EntityFrameworkCore;
using System;
using System.Linq;
using System.Threading.Tasks;

namespace ServerSignalR.Models
{
   public class ApplicationContext : IdentityDbContext<User, Role, Guid>
   {
      public DbSet<Notify> Notifies { get; set; }
      public DbSet<Text> Texts { get; set; }
      public DbSet<Token> Tokens { get; set; }

      public ApplicationContext(DbContextOptions<ApplicationContext> options)
         : base(options)
      {
         Database.EnsureCreated();
      }
      protected override void OnModelCreating(ModelBuilder modelBuilder)
      {
         modelBuilder.Entity<User>().Property(p => p.Id).ValueGeneratedOnAdd();
         base.OnModelCreating(modelBuilder);
      }
   }
}


using Microsoft.AspNetCore.Identity;
using System;

namespace ServerSignalR.Models
{
   public class User:IdentityUser<Guid>
   {

   }
}
using System;
using System.Collections.Generic;

namespace ServerSignalR.Models
{
   public class Token
   {
      public Guid Id { get; set; }
      public string Value { get; set; }
      public List<Text> Texts { get; set; }
   }
}
```

```
using System;
using System.Collections.Generic;

namespace ServerSignalR.Models
{
    public class Text
    {
        public Guid Id { get; set; }
        public string Name { get; set; }
        public LevelConfidence LevelConfidence { get; set; }
        public Guid UserId { get; set; }
        public User User { get; set; }
        public List<Token > Tokens { get; set; }
    }
}
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;

namespace ServerSignalR.Models
{
    public class Notify
    {
        public Guid Id { get; set; }
        public DateTime DateTime { get; set; }
        public string Message { get; set; }
        public Guid UserId { get; set; }
        public User User { get; set; }
        public NotifyType NotifyType { get; set; }
    }
}
using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore;
using Microsoft.IdentityModel.Tokens;
using ServerSignalR.JWT;
using ServerSignalR.Models;
using System;
using System.Collections.Generic;
using System.IdentityModel.Tokens.Jwt;
using System.Security.Claims;
using System.Threading.Tasks;

namespace ServerSignalR.Controllers
{
    public class AccountController : Controller
    {
        private ApplicationContext _context;
        public AccountController(ApplicationContext context)
        {
            _context = context;
        }
        [HttpPost("/token")]
        public async Task<IActionResult> Token(string username, string password)
        {
            var identity = await GetIdentity(username, password);
```

```
            if (identity == null)
            {
                return BadRequest("Invalid username or password.");
            }

            var now = DateTime.UtcNow;
            var jwt = new JwtSecurityToken(
                    issuer: AuthOptions.ISSUER,
                    audience: AuthOptions.AUDIENCE,
                    notBefore: now,
                    claims: identity.Claims,
                    expires: now.Add(TimeSpan.FromMinutes(AuthOptions.LIFETIME)),
                    signingCredentials: new SigningCredentials(AuthOptions.GetSymmetricSecurityKey(),
SecurityAlgorithms.HmacSha256));
            var encodedJwt = new JwtSecurityTokenHandler().WriteToken(jwt);

            var response = new
            {
                access_token = encodedJwt,
                username = identity.Name
            };
            return Json(response);
        }

        private async Task<ClaimsIdentity> GetIdentity(string username, string password)
        {
            User person = await _context.Users.Include(r => r.Role).FirstOrDefaultAsync(x => x.Email ==
username && x.Password == password);
            if (person != null)
            {
                var claims = new List<Claim>
                {
                    new Claim(ClaimsIdentity.DefaultNameClaimType, person.Email),
                    new Claim(ClaimsIdentity.DefaultRoleClaimType, person.Role.Name)
                };
                ClaimsIdentity claimsIdentity =
                new ClaimsIdentity(claims, "Token", ClaimsIdentity.DefaultNameClaimType,
                    ClaimsIdentity.DefaultRoleClaimType);
                return claimsIdentity;
            }
            return null;
        }
    }
}
using Microsoft.AspNetCore.Authentication.JwtBearer;
using Microsoft.AspNetCore.Builder;
using Microsoft.AspNetCore.Hosting;
using Microsoft.AspNetCore.HttpsPolicy;
using Microsoft.EntityFrameworkCore;
using Microsoft.Extensions.Configuration;
using Microsoft.Extensions.DependencyInjection;
using Microsoft.Extensions.Hosting;
using Microsoft.IdentityModel.Tokens;
using ServerSignalR.JWT;
using ServerSignalR.Models;
using System;
```

```
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;

namespace ServerSignalR
{
   public class Startup
   {
      public Startup(IConfiguration configuration)
      {
         Configuration = configuration;
      }

      public IConfiguration Configuration { get; }

      // This method gets called by the runtime. Use this method to add services to the container.
      public void ConfigureServices(IServiceCollection services)
      {
         string connection =
"Server=(localdb)\\mssqllocaldb;Database=DiplomDB;Trusted_Connection=True;";
         services.AddDbContext<ApplicationContext>(options => options.UseSqlServer(connection));

         services.AddAuthentication(JwtBearerDefaults.AuthenticationScheme)
            .AddJwtBearer(options =>
            {
               options.RequireHttpsMetadata = false;
               options.TokenValidationParameters = new TokenValidationParameters
               {
                  ValidateIssuer = true,
                  ValidIssuer = AuthOptions.ISSUER,
                  ValidateAudience = true,
                  ValidAudience = AuthOptions.AUDIENCE,
                  ValidateLifetime = true,
                  IssuerSigningKey = AuthOptions.GetSymmetricSecurityKey(),
                  ValidateIssuerSigningKey = true,
               };
               options.Events = new JwtBearerEvents
               {
                  OnMessageReceived = context =>
                  {
                     var accessToken = context.Request.Query["access_token"];
                     var path = context.HttpContext.Request.Path;
                     if (!string.IsNullOrEmpty(accessToken) &&
                        (path.StartsWithSegments("/chat")))
                     {
                        context.Token = accessToken;
                     }
                     return Task.CompletedTask;
                  }
               };
            });
         services.AddSignalR();
         services.AddControllers();
      }

      public void Configure(IApplicationBuilder app)
```

```
        {
            app.UseDeveloperExceptionPage();

            app.UseHttpsRedirection();
            app.UseDefaultFiles();
            app.UseStaticFiles();

            app.UseRouting();

            app.UseAuthentication();
            app.UseAuthorization();

            app.UseEndpoints(endpoints =>
            {
                endpoints.MapControllers();
                endpoints.MapHub<MessageHub>("/message");
            });
        }
    }
}
using System;
using System.IO;
using System.Security.Permissions;
using System.Threading.Tasks;

namespace Watcher.FileManagment
{
    public class FileManager
    {
        public delegate void FileManagerNotifyHandler(string message);
        public event FileManagerNotifyHandler Notify;

        public async Task RunAsync(string path)
        {
            await Task.Run(() => Run(path));
        }
        private void Run( string path)
        {
            // If a directory is not specified, exit program.
            if (!string.IsNullOrEmpty(path))
            {
                // Display the proper way to call the program.
                Notify("Usage: Watcher.exe (directory)");
                return;
            }

            // Create a new FileSystemWatcher and set its properties.
            using (FileSystemWatcher watcher = new FileSystemWatcher())
            {
                watcher.Path = path;
                watcher.IncludeSubdirectories = true;

                // Watch for changes in LastAccess and LastWrite times, and
                // the renaming of files or directories.
                watcher.NotifyFilter = NotifyFilters.LastAccess
                                     | NotifyFilters.LastWrite
```

```
                                | NotifyFilters.FileName
                                | NotifyFilters.DirectoryName;

            // Only watch text files.
            watcher.Filter = "*.*";

            // Add event handlers.
            watcher.Changed += OnChanged;
            watcher.Created += OnChanged;
            watcher.Deleted += OnChanged;
            watcher.Renamed += OnRenamed;

            // Begin watching.
            watcher.EnableRaisingEvents = true;

            // Wait for the user to quit the program.
            //Notify("Press 'q' to quit the sample.");
            while (true) ;

        }
    }

    // Define the event handlers.
    private void OnChanged(object source, FileSystemEventArgs e) =>
        // Specify what is done when a file is changed, created, or deleted.
        Notify($"File: {e.FullPath} {e.ChangeType}");

    private void OnRenamed(object source, RenamedEventArgs e) =>
        // Specify what is done when a file is renamed.
        Notify($"File: {e.OldFullPath} renamed to {e.FullPath}");
  }
}
using System.Collections.Generic;
using Watcher.LexicalAnalyze.Params;

namespace Watcher.LexicalAnalyze.Core
{
  public interface ILexicalData
  {
      List<Text> Texts { get; set; }
      List<Lexam> Lexams { get; set; }
  }
}using Watcher.LexicalAnalyze.Params;

namespace Watcher.LexicalAnalyze.Core
{
  public interface ILexicalSearch
  {
    public LevelOfSimilarity Search(string lexes);
  }
}
namespace Watcher.LexicalAnalyze.Params
{
  public enum LevelOfSimilarity
  {
    UltraLow,
```

```
        Low,
        Medium,
        Hight,
        UltraHight,
        Similyar
    }
            return await Task.FromResult(text.ToString());
        }
    }
}
using System.IO;
using System.Threading.Tasks;

namespace Watcher.LexicalAnalyze.TextRead
{
    public class TxtReader : IFileRead
    {
        public async Task<string> Read(string path)
        {

            using StreamReader sr = new StreamReader(path);
            string result = await sr.ReadToEndAsync();
            return result;
        }
    }
}
using System.Collections.Generic;
using Watcher.LexicalAnalyze.Core;
using Watcher.LexicalAnalyze.Params;

namespace Watcher.LexicalAnalyze
{
    public class LexicalData : ILexicalData
    {
        public List<Text> Texts { get; set; } = new List<Text>();
        public List<Lexam> Lexams { get; set; } = new List<Lexam>();
    }
}
using System.IO;
using System.Linq;
using System.Threading.Tasks;
using Watcher.LexicalAnalyze.Core;
using Watcher.LexicalAnalyze.Params;
using Watcher.LexicalAnalyze.TextRead;

namespace Watcher.LexicalAnalyze
{
    public class LexicalAnalyzer
    {
        private ILexicalData lexicalData;
        private ILexicalSearch lexicalSearch;
        public LexicalAnalyzer(ILexicalData lexicalData, ILexicalSearch lexicalSearch)
        {
            this.lexicalSearch = lexicalSearch;
            this.lexicalData = lexicalData;
        }
```

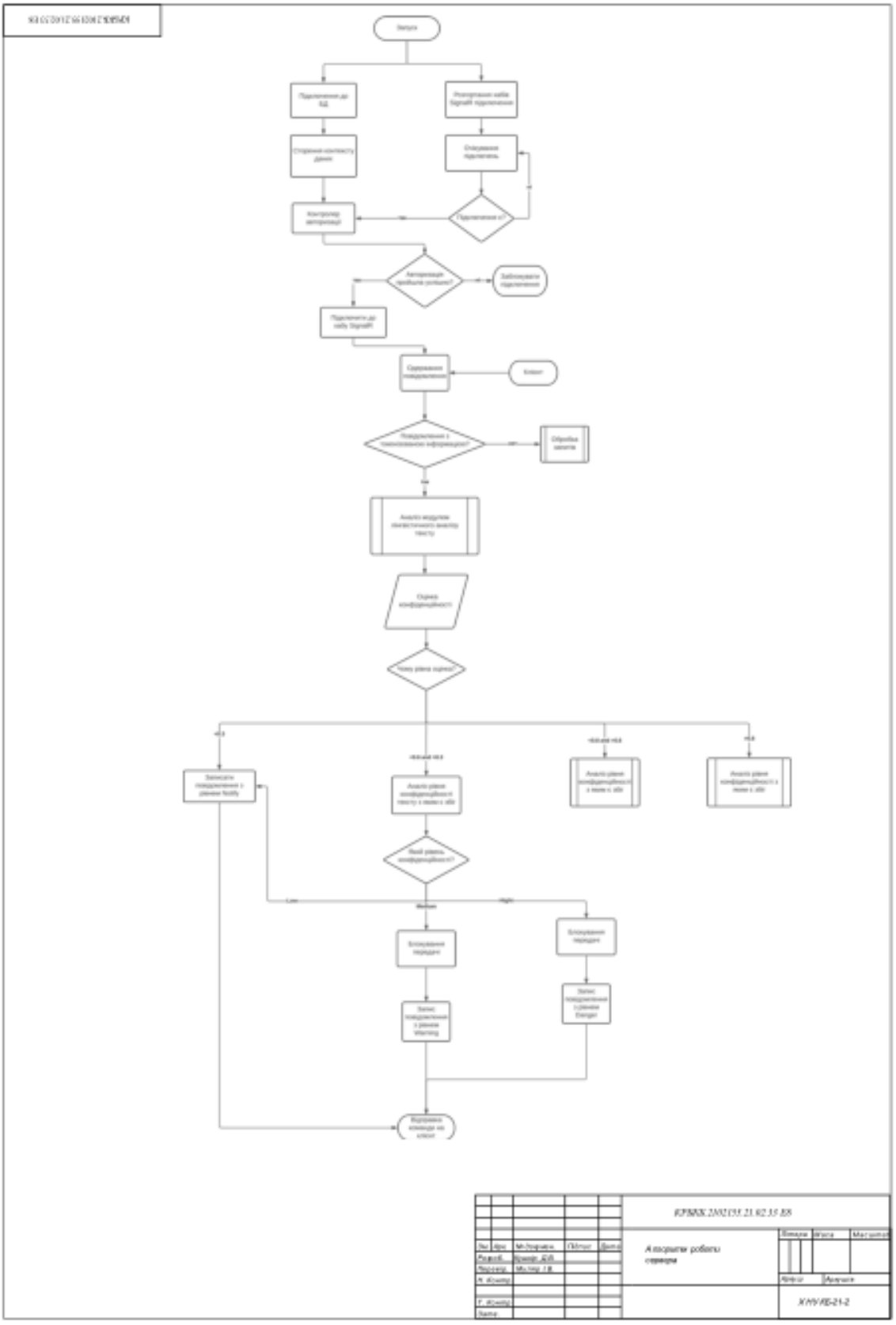
```
    public LexicalAnalyzer(ILexicalData lexicalData) : this(lexicalData, new
WordBagsLexicalSearch(lexicalData))
    {

    }

    private string[] Delimiter = { ".", ";", ",", "(", ")", "+", "-", "*", "/", "=", ">", "<","\\","[","]","\r","\n"," " };

    public async Task<string> ReadFileAsync(string path)
    {
      if (!File.Exists(path))
      {
        throw new FileNotFoundException();
      }
      switch (path.Split(".").Last())
      {
        case "txt":
          return await new TxtReader().Read(path);
        case "doc":
        case "docx":
          return await new DocReader().Read(path);
        case "pdf":
          return await new PDFReader().Read(path);
        default:
          throw new FileFormatException("File format is not suported");
      }
    }
    public async Task AddTextAsync(string textName, string path, TextPrivacy textPrivacy =
TextPrivacy.Low)
    {
      string buffer = await ReadFileAsync(path);
      Text text = new Text();
      text.Privacy = textPrivacy;
      text.TextName = textName;
      text.Tokens = Tokenization(buffer);
      lexicalData.Texts.Add(text);
    }
    private string Tokenization(string text)
    {
      string result="";
      string[] textAfterSplit = text.Split(Delimiter,System.StringSplitOptions.RemoveEmptyEntries);
      foreach (var t in textAfterSplit)
      {
        int indexOfLexam = lexicalData.Lexams.FindIndex(p => p.NormolizedLex == t.ToUpper());
        if (indexOfLexam==-1)
        {
          var lexam = new Lexam { Lex = t, NormolizedLex = t.ToUpper() };
          lexicalData.Lexams.Add(lexam);
          result += lexam.Value + " ";
        }
        else
        {
          result += lexicalData.Lexams[indexOfLexam].Value + " ";
        }

      }
```

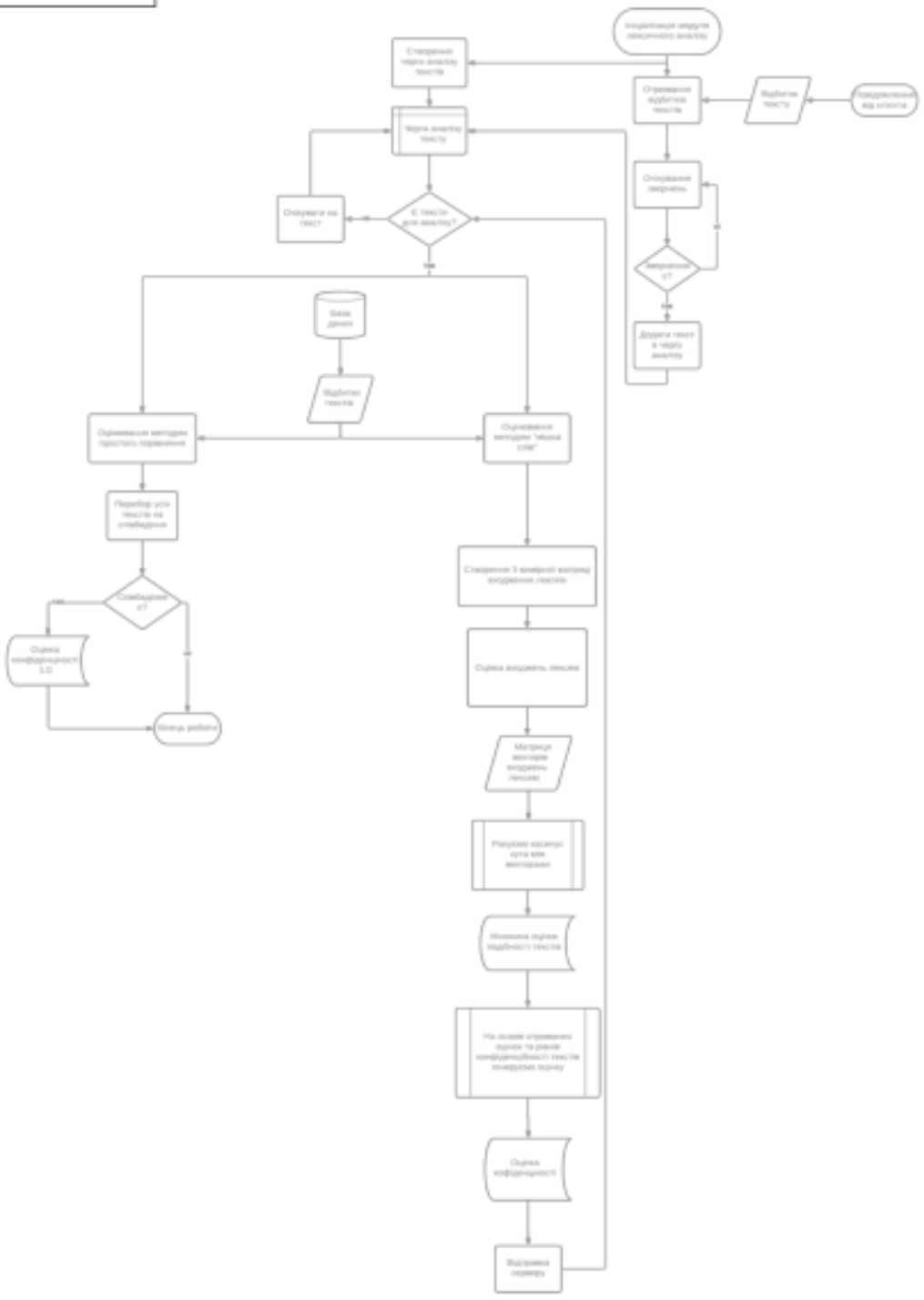
```
            return result;
        }
        public async Task<LevelOfSimilarity> GetSimiliarityScoreAsync(string path)
        {
            string buffer =await ReadFileAsync(path);
            buffer = Tokenization(buffer);
            return lexicalSearch.Search(buffer);
        }

    }
}
using System;
using System.Collections.Generic;
using System.Linq;
using Watcher.LexicalAnalyze.Core;
using Watcher.LexicalAnalyze.Params;

namespace Watcher.LexicalAnalyze
{
    public class WordBagsLexicalSearch : ILexicalSearch
    {
        private readonly List<Lexam> Lexams;
        private readonly List<Text> Texts;
        public WordBagsLexicalSearch(ILexicalData lexicalData)
        {
            Lexams = lexicalData.Lexams;
            Texts = lexicalData.Texts;
        }
        public LevelOfSimilarity Search(string lexes)
        {
            float maxMark = 0, bufMark;
            List<string> verifiableWordBug = GetWordBug(lexes);
            foreach(var t in Texts)
            {
                bufMark = getScoreOfSimilyarity(verifiableWordBug, GetWordBug(t.Tokens));
                if (bufMark > maxMark)
                    maxMark = bufMark;
            }
            if (maxMark == 1.0)
                return LevelOfSimilarity.Similyar;
            else if (maxMark >= 0.9)
                return LevelOfSimilarity.UltraHight;
            else if (maxMark >= 0.8)
                return LevelOfSimilarity.Hight;
            else if (maxMark >= 0.65)
                return LevelOfSimilarity.Medium;
            else if (maxMark >= 0.35)
                return LevelOfSimilarity.Low;
            else
                return LevelOfSimilarity.UltraLow;
        }
        private float getScoreOfSimilyarity(List<string> verifiableWordBug,List<string> originalWordBug)
        {
            int score = 0;
            if (verifiableWordBug.Count <= originalWordBug.Count)
```

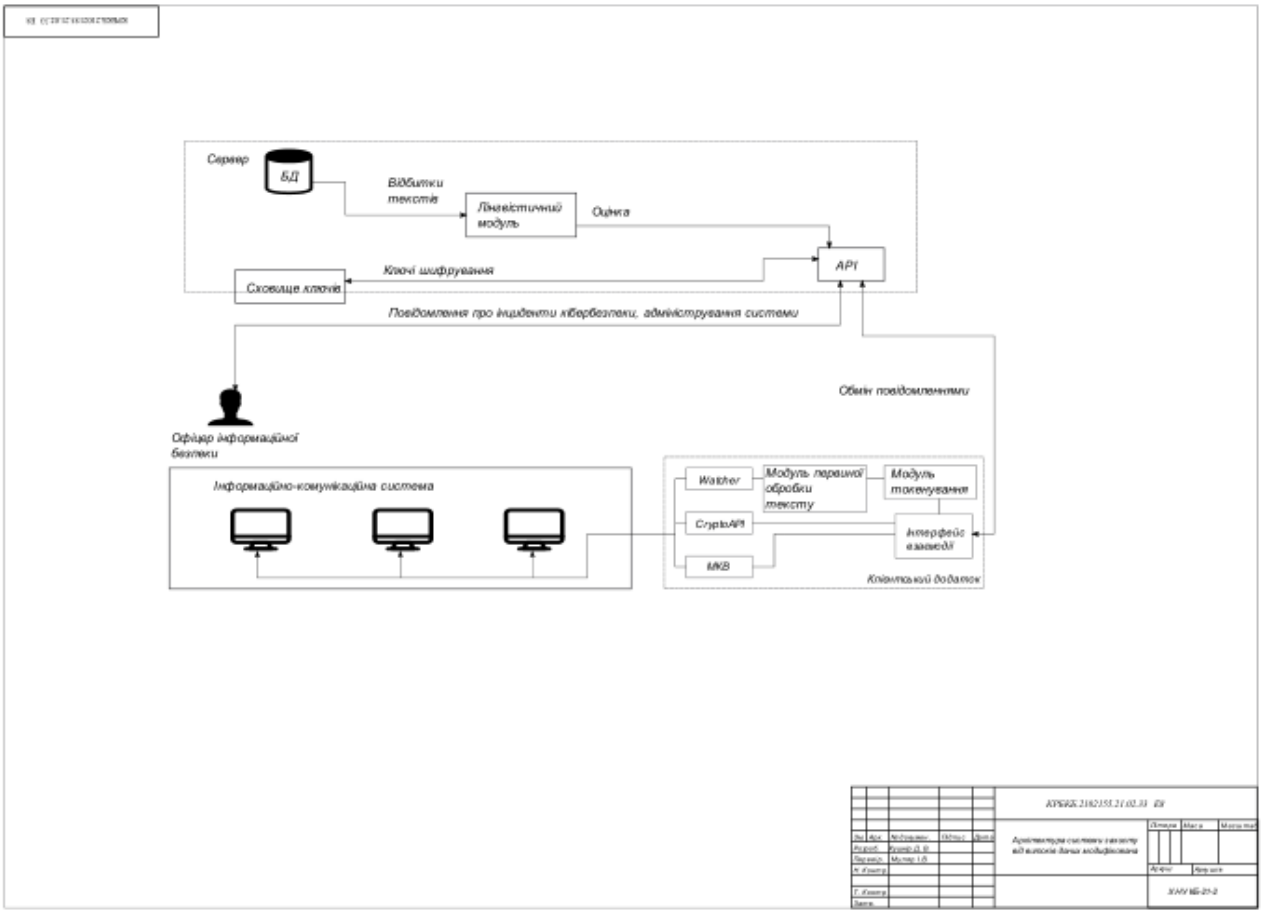
```
            {
                foreach (var t in verifiableWordBug)
                    if (originalWordBug.Any(predicate => predicate == t))
                        score++;
                return (float)score / (float)verifiableWordBug.Count;
            }
            else
            {
                foreach (var t in originalWordBug)
                {
                    if (verifiableWordBug.Any(predicate => predicate == t))
                        score++;
                }
                return (float)score / (float)originalWordBug.Count;
            }
        }
        private List<string> GetWordBug(string text)
        {
            List<string> result = new List<string>();
            foreach(var t in text.Split(" ",StringSplitOptions.RemoveEmptyEntries))
            {
                if (result.IndexOf(t) == -1)
                {
                    result.Add(t);
                }
                else continue;
            }
            return result;

        }
    }
}
```

ДОДАТОК Б

Копія графічної частини




Инициализация модуля лексического анализа
Отправка задачи на текст
Задачи на текст
Клиентский вид клиента
Очередь задач анализу текста
Очередь на текст
Есть текст для анализа?
Ожидание завершения
Завершено ли?
Да
Нет
Задача анализа текста в очередь анализа
База данных
Выборка текста
Определение метрики простого перемещения
Определение метрики "мешок слов"
Перевод всех текстов на словарную
Словарная ли?
Оценка конфиденциальности 1.0
Конец работы
Сохранение 3 вложений векторов входящих лексем
Оценка входимость лексем
Матрица векторов входимость лексем
Рассчет косинус пути мер векторами
Множество оценок подобности текстов
На основе отправленных оценок по уровню конфиденциальности текстов генерируем оценку
Оценка конфиденциальности
Выполнена сервису
КРКИБ.200115.21.02.53 ВВ
Алгоритмы работы модуля лексического анализу
Литера
Масса
Масштаб
Лист
Листов
ХНУ КБ-21-2


Сервер
БД
Відбитки текстів
Лінгвістичний модуль
Оцінка
API
Ключі шифрування
Сховище ключів
Повідомлення про інциденти кібербезпеки, адміністрування системи
Обмін повідомленнями
Офіцер інформаційної безпеки
Інформаційно-комунікаційна система
Watcher
Модуль первинної обробки тексту
Модуль токенування
CryptoAPI
Інтерфейс взаємодії
WKB
Клієнтський додаток

Завідувачу кафедри кібербезпеки
к.т.н., доц. Кльоцу Ю.П.
Кушнір Давида Володимировича
ПІБ здобувача вищої освіти

Студента ФІТ, 4 курсу, групи КБ-21-2

ЗАЯВА

З правилами чинного Положення «Про систему забезпечення академічної доброчесності у Хмельницькому національному університеті» від 31.08.2023, згідно з яким виявлення плагіату є підставою для відмови в допуску кваліфікаційної роботи до захисту та застосування заходів дисциплінарної та академічної відповідальності, ознайомлена. Про використання програмно-технічних засобів для перевірки кваліфікаційних робіт здобувачів вищої освіти на плагіат оповіщена та надаю свою згоду на обробку та збереження університетом моєї роботи в інституційному репозитарії університету.

Також надаю університету право на передачу моєї роботи для обробки та збереження в базах даних програмно-технічних засобів (Unicheck та Anti-Plagiarism) та використання роботи для виявлення плагіату в інших роботах, які перевіряються програмно-технічними засобами та користувачами, що мають доступ до цих програмно-технічних засобів, виключно в обмежених цілях для виявлення плагіату в текстах робіт.

Робота для перевірки університетом надається в друкованому та електронному варіанті. Електронна версія моєї роботи збігається (ідентична) з друкованою.

06.06.2025
дата

підпис

Протокол аналізу звіту подібності науковим керівником

Заявляю, що я ознайомився (-лась) з Повним звітом подібності, який був згенерований Системою виявлення і запобігання плагіату щодо роботи:

Автор: Кушнір Давид Володимирович

Співавтор:

Назва: Система протидії витокам даних на підприємстві з використанням DLP

Науковий керівник:

Підрозділ: Кафедра кібербезпеки

Коефіцієнт подібності 1:1.5%

Коефіцієнт подібності 2:0%

Мікропробіли: 0

Заміна букв: 2

Інтервали: 0

Білі знаки: 0

Дата створення звіту: 2025-06-04 06:16:43.0

Після аналізу Звіту подібності констатую наступне:

☒ **Запозичення, виявлені в роботі є законними і не є плагіатом. Рівень подібності не перевищує допустимої межі. Таким чином робота незалежна і приймається.**

☐ **Запозичення не є плагіатом, але перевищено граничне значення рівня подібностей. Таким чином робота повертається на доопрацювання.**

☐ **Виявлено запозичення і плагіат або навмисні текстові спотворення (маніпуляції), як передбачувані спроби укриття плагіату, які роблять роботу невідповідною вимогам законодавства (Ст. 32. ЗУ Про вищу освіту, пункт 3.1, Ст. 42. ЗУ Про освіту) та вимог НАЗЯВО (Критерій 5), а також кодексу етики і процедурам. Таким чином робота не приймається.**

Обгрунтування:

05.06.2025р.

Anti-Plagiarism (UA) v-15.281 Educational

The maximum coincidence with one document 0.0%

Dictionaries check: en_US, ru_RU, ua_UA. **Errors in the documents: 6%**

ID: 243247 Title: Система протидії витокам даних на підприємстві з використанням DLP Added in a DB: 2025-06-03 Authors: Кушнір Давид Володимирович Heads: Муляр І.В. Consultants: Opponents:	Document		Sum coincidence on the DB	
	Symbols	Lexemes	Symbols	Lexemes
	69712	1071	726 (1%)	11 (1%)

Plagiarism sources

ID	Description	Plagiarism presence in the document	
		Symbols	Lexemes

РІШЕННЯ ЕКСПЕРНОЇ КОМІСІЇ

КАФЕДРИ КІБЕРБЕЗПЕКИ

ПРО ДОПУСК КВАЛІФІКАЦІЙНОЇ РОБОТИ ДО ЗАХИСТУ

Підтверджуємо ознайомлення з результатом звіту подібності щодо роботи, генерованого системою виявлення текстових збігів/ідентичності/схожості:

Назва: Система протидії витокам даних на підприємстві з використанням DLP

Автор: Кушнір Давид
Спеціальність: 125 – Кібербезпека
Освітня програма:Кібербезпека
Науковий керівник:Ігор МУЛЯР, канд. техн. наук, доцент
Після аналізу звіту подібності зроблено такий висновок:

№	Висновок	Позначка про відповідність
1	Запозичення, виявлені в роботі, є законними і не є плагіатом (далі – зазначаються підстави віднесення запозичень до правомірних). Робота приймається до захисту.	відповідає
2	Виявлені запозичення не є плагіатом, розміщені в розділах, які не описують безпосередньо авторське дослідження, але кількість цитат перевищує обсяг, виправданий поставленою метою роботи (далі – зазначаються детальні та аргументовані підстави віднесення запозичень до правомірних). Робота приймається до захисту, але має бути відкоригована. Відкоригований варіант має бути поданий на кафедру за 2 дні до захисту, разом із заявою щодо самостійності виконання письмової роботи та ідентичності друкованої та електронної версії роботи.	
3	Виявлені запозичення не є плагіатом, але частково розміщені в розділах, які описують безпосередньо авторське дослідження, а кількість цитат перевищує обсяг, виправданий поставленою метою роботи. В зв'язку з цим мета роботи та поставлені завдання не були досягнені. Робота може бути допущена до захисту (наступного року) після того як буде відкоригована та допрацьована і успішно пройде повторну перевірку на академічний плагіат.	
4	Робота містить навмисні текстові спотворення, передбачувані спроби укриття запозичень або інші прояви академічного плагіату. Робота містить фабрикацію або фальсифікацію даних. Робота не допускається до захисту.	
5	Інше:	

Підтвердження:

Оригінальність тексту роботи за результатами перевірки системою StrikePlagiarism складає 99%, оригінальність тексту роботи за результатами перевірки системою Anti-Plagiarism складає 97,7%.

Згідно з правилами чинного Положення «Про систему забезпечення академічної доброчесності у Хмельницькому національному університеті» від 24.09.2024, авторська робота, обсяг оригінального тексту у відсотках до загального обсягу матеріалу в якій складає 90-100%, визначається роботою з високою унікальністю тексту і допускається до захисту.

Керівник роботи _____ Ігор МУЛЯР

Гарант ОП _____ Віктор ЧЕШУН

Завідувач кафедри кібербезпеки _____ Юрій КЛЬОЦ

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

РЕЦЕНЗІЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ

освітнього ступеня «бакалавр»

Студент Кушнір Давид Володимирович
Тема Система протидії витокам даних на підприємстві з використанням DLP

Спеціальність 125 – Кібербезпека

Обсяг кваліфікаційної роботи освітньо-кваліфікаційного рівня «бакалавр»:

кількість листів креслень 3; кількість сторінок записки 69.

1. Короткий зміст роботи та прийнятих рішень У кваліфікаційній роботі, відповідно до поставленого завдання, проведене дослідження предметної області, та існуючі рішення в галузі пошуку враливостей. Також розроблено архітектуру DLP системи, та реалізовано серверну частину та клієнтський додаток. Налаштовано систему, та проведено тестування автоматизованої системи.

2. Висновок про відповідність кваліфікаційної роботи завданню У кваліфікаційній роботі повністю виконано поставлене завдання як у теоретичній, так і в практичній частині

3. Характеристика виконання кожного розділу роботи, ступінь використання останніх досягнень науки і техніки і передових методів роботи: У розділі 1 розглянуто загрози інформаційної безпеки, технології захисту даних, і зформовано постановку задачі. У розділі 2 обґрунтовано обраний підхід для реалізації системи, та описано архітектуру запобігання витоків даних. У розділі 3 розроблено програмну реалізацію системи, описано інструментарій розробки та технологію реалізації системи. Також, проведено налаштування та тестування системи

4. Позитивні сторони роботи Робота базується на детальному аналізі загроз інформаційної безпеки, розглядаються також можливі типи атак на вебдодатки. Робота над такою системою є надзвичайно актуальною та практично значущою. Можна отримати глибокі знання у сфері веббезпеки та вдосконалите навички програмування на C#. Це дозволить зробити внесок у спільноту кібербезпеки, підвищити свою цінність на ринку праці та зіткнутися з цікавими технічними викликами.

5. Негативні сторони роботи Під час тестування розробленої системи захисту від витоків даних було виявлено недолік, пов'язаний із стійкістю до навантаження

6. Оцінка графічного оформлення та пояснювальної записки роботи Графічне оформлення кваліфікаційної роботи відповідає темі роботи та виконане з дотриманням стандартів. В цілому, графічне оформлення є якісним, а пояснювальна записка відповідає нормам оформлення.

7. Відгук про роботу в цілому Кваліфікаційна робота заслуговує позитивної оцінки, оскільки весь матеріал роботи є структурованим, чітким та послідовним. Усі розділи роботи мають логічну послідовність, що сприяє зрозумінню викладеного матеріалу в рамках теми роботи. Графічний матеріал допомагає наочно продемонструвати доцільність та ефективність прийнятих рішень у проєктуванні та супроводі розробленої системи протидії витокам даних

8. Інші зауваження

9. Оцінка кваліфікаційної роботи Враховуючи всі позитивні сторони кваліфікаційної роботи, а також негативні сторони, які не зменшують практичну цінність отриманих результатів і загальну якість роботи, рекомендованою оцінкою є «добре»

РЕЦЕНЗЕНТ (прізвище, ім’я, по батькові, посада, місце роботи)
Пивовар Олег Сергійович, к.т.н., доцент кафедри ТМІТ

«10» 06 2025р

__ 2025.

Пивовар О.С (підпис)