МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ ЕКОНОМІЧНИЙ УНІВЕРСИТЕТ ІМЕНІ ВАДИМА ГЕТЬМАНА

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ
«ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ В ЕКОНОМІЦІ»

Кафедра системного аналізу та кібербезпеки

ОСВІТНЬО-ПРОФЕСІЙНА ПРОГРАМА «КІБЕРБЕЗПЕКА»
Галузь знань 12 «Інформаційні технології»
Спеціальність 125 «Кібербезпека»

Форма навчання: очна (денна)

КВАЛІФІКАЦІЙНА БАКАЛАВРСЬКА РОБОТА

на тему: **«Розробка програмного засобу для виявлення фішингових електронних листів із використанням методів штучного інтелекту»**

здобувача: Стасюка Вадима Вікторовича _____

Науковий керівник: Ст. викл. Кондратюк А.Г.. _____

Робота допущена до захисту перед екзаменаційною комісією з атестації здобувачів вищої освіти (ЕК)

Завідувач кафедри: д.ф.-м.н., проф. Джалладова І.А.

_____

КИЇВ 2026

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ ЕКОНОМІЧНИЙ УНІВЕРСИТЕТ ІМЕНІ ВАДИМА ГЕТЬМАНА

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ
«ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ В ЕКОНОМІЦІ»

Кафедра системного аналізу та кібербезпеки

ОСВІТНЬО-ПРОФЕСІЙНА ПРОГРАМА «КІБЕРБЕЗПЕКА»
Галузь знань 12 «Інформаційні технології»
Спеціальність 125 «Кібербезпека»

ПОГОДЖЕНО
Керівник проектної групи (гарант)
освітньо-професійної програми
А.В. Бєгун
(підпис) (ініціали, прізвище)
_____ 2026 р.

ЗАТВЕРДЖУЮ:
Завідувач кафедри
І.А. Джалладова
(підпис) (ініціали, прізвище)
_____ 2026 р.

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ

здобувача вищої освіти: Стасюка Вадима Вікторовича
(прізвище, ім'я, по батькові)

очної (денної) **форми навчання**
очної (денної), заочної, дистанційної

на підготовку кваліфікаційної бакалаврської роботи

на тему: «Розробка програмного засобу для виявлення фішингових електронних листів із використанням методів штучного інтелекту»

Тему затверджено наказом ректора Університету від «___» _____ 2026 р. №__

Кваліфікаційна бакалаврська робота виконується на матеріалах
_____
_____

План кваліфікаційної бакалаврської роботи

Розділ 1 | Аналіз фішингових атак в електронній пошті та методів їх виявлення

(назва розділу)

Розділ 2 | Теоретичні основи виявлення фішингових електронних листів із використанням методів штучного інтелекту

(назва розділу)

Розділ 3 | Розробка програмного засобу для виявлення фішингових електронних листів

(назва розділу)

Об’єкт дослідження:

Об’єктом дослідження є процеси виявлення та аналізу фішингових електронних листів у системах електронної пошти.

Предмет дослідження:

Предметом дослідження є методи та засоби виявлення фішингових електронних листів із використанням технологій штучного інтелекту, аналізу вкладень, URL-адрес та репутаційних сервісів.

Мета кваліфікаційної бакалаврської роботи:

Метою кваліфікаційної бакалаврської роботи є дослідження сучасних методів виявлення фішингових електронних листів та розробка програмного засобу для автоматизованого аналізу електронних повідомлень із використанням методів штучного інтелекту, сервісу VirusTotal та платформи OpenRouter.

Конкретні завдання, які здобувач повинен виконати для досягнення поставленої мети:

- проаналізувати сучасні види фішингових атак в електронній пошті та механізми їх реалізації;
- дослідити існуючі методи виявлення фішингових електронних листів;
- проаналізувати характерні ознаки фішингових повідомлень, вкладень та вебпосилань;
- дослідити можливості використання методів штучного інтелекту для аналізу електронних листів;

• дослідити можливості використання сервісу VirusTotal для перевірки вкладень та URL-адрес;

• розробити архітектуру програмного засобу для автоматизованого аналізу електронної пошти;

• реалізувати модуль отримання електронних листів через протокол IMAP;

• реалізувати модуль перевірки вкладень та вебпосилань із використанням VirusTotal API;

• реалізувати модуль інтелектуального аналізу електронних листів із використанням платформи OpenRouter;

• провести тестування розробленого програмного засобу та оцінити ефективність його роботи.

Завдання підготував науковий керівник _____ Кондратюк А.Г.

(підпис)(прізвище ініціали)

Завдання одержав здобувач _____ Стасюк В.В.

(підпис)(прізвище ініціали)

РЕФЕРАТ

Кваліфікаційна бакалаврська робота: 64 с., 26 рис., 6 табл., 1 додатки, 30 джерел.

Об’єкт дослідження – процеси виявлення та аналізу фішингових електронних листів у системах електронної пошти.

Предмет дослідження – методи та засоби виявлення фішингових електронних листів із використанням технологій штучного інтелекту, аналізу вкладень, URL-адрес та репутаційних сервісів.

Мета кваліфікаційної роботи полягає у дослідженні сучасних методів виявлення фішингових електронних листів та розробленні програмного засобу для автоматизованого аналізу повідомлень електронної пошти із використанням методів штучного інтелекту.

Методи дослідження: аналіз та узагальнення наукових джерел, порівняльний аналіз існуючих методів виявлення фішингових повідомлень, аналіз вмісту електронних листів, аналіз вкладень та URL-адрес, використання репутаційних сервісів, методи штучного інтелекту, проєктування програмних систем, програмна реалізація та експериментальне тестування.

У роботі досліджено сучасні фішингові атаки в електронній пошті, їх основні види, механізми реалізації та методи протидії. Проведено аналіз існуючих підходів до виявлення фішингових електронних листів, розглянуто особливості використання методів штучного інтелекту в задачах аналізу електронних повідомлень, а також досліджено можливості застосування репутаційних сервісів для перевірки вкладень та вебпосилань. Особливу увагу приділено аналізу ознак фішингових листів, методам виявлення соціальної інженерії та використанню великих мовних моделей для автоматизованого аналізу повідомлень.

У практичній частині роботи розроблено програмний засіб для виявлення фішингових електронних листів, реалізований із використанням платформи Electron та середовища Node.js. Система забезпечує підключення до поштової скриньки через

протокол IMAP, автоматичне отримання електронних листів, перевірку вкладень і URL-

адрес через сервіс VirusTotal та інтелектуальний аналіз повідомлень за допомогою платформи OpenRouter. За результатами аналізу формується детальний протокол перевірки із зазначенням рівня ризику, виявлених загроз та рекомендацій для користувача.

Практичне значення отриманих результатів полягає у можливості використання розробленого програмного засобу для підвищення рівня безпеки електронної пошти, автоматизації процесу виявлення фішингових повідомлень та зниження ризику компрометації конфіденційної інформації внаслідок фішингових атак.

Ключові слова: ФІШИНГ, ЕЛЕКТРОННА ПОШТА, ФІШИНГОВИЙ ЛИСТ, ШТУЧНИЙ ІНТЕЛЕКТ, OPENROUTER, VIRUSTOTAL, IMAP, КІБЕРБЕЗПЕКА, АНАЛІЗ ВКЛАДЕНЬ, URL-АДРЕСА.

ВІДГУК
на кваліфікаційну бакалаврську роботу

студента Стасюка Вадима Вікторовича,

на тему: «Розробка програмного засобу для виявлення фішингових електронних листів із використанням методів штучного інтелекту»

Актуальність теми: с*учасні фішингові повідомлення все частіше створюються із використанням автоматизованих інструментів і технологій штучного інтелекту, що значно ускладнює їх виявлення традиційними засобами захисту. Більшість існуючих систем фільтрації електронної пошти базується на сигнатурному або евристичному аналізі, який не завжди дозволяє ефективно виявляти нові та модифіковані фішингові кампанії. У зв'язку з цим актуальним завданням є розробка програмних засобів, що поєднують класичні методи аналізу електронних повідомлень із сучасними методами штучного інтелекту.*

Повнота розкриття теми: *кваліфікаційна робота Стасюка Вадима Вікторовича має на меті розробку програмного засобу для виявлення фішингових електронних листів із використанням методів штучного інтелекту. Зміст роботи відповідає завданню.*

Теоретичний рівень: *представлена робота свідчить про достатній рівень засвоєння автором навчального матеріалу з дисциплін, що викладаються на кафедрі.*

Практична значущість: *особливістю роботи є її прикладна практична спрямованість: розроблений програмний продукт може бути використаний як в корпоративному сегменті так і в особистому використанні для виявлення фішингових електронних листів.*

Самостійність виконання роботи: *автором проведено аналіз існуючих методів та програмних засобів виявлення фішингових атак. Самостійно обрано інструментальні засоби, що дозволяють виконати поставлені завдання. Під час роботи Стасюк В. В. проявив розумну ініціативу та самостійність, показав уміння відокремлювати головне від другорядного.*

Якість оформлення, загальна та спеціальна грамотність: *пояснювальна записка виконана згідно вимог до кваліфікаційних робіт, що свідчить про загальну грамотність, але містить стилістичні помилки.*

Переваги та недоліки: *до переваг можна віднести повноту проведеного дослідження, аналіз та формування стратегії впровадження операційного центру безпеки та практична частина яка висвітлює доступність рішення. Недоліком можна вважати врахування тільки фінансових показників для обґрунтування вибору запропонованого рішення.*

Загальна оцінка роботи та висновок щодо рекомендації до захисту в ЕК: *робота відповідає вимогам до кваліфікаційних робіт та може бути оцінена добре, а Стасюк В. В.. рекомендується до захисту роботи перед екзаменаційною комісією.*

Науковий керівник

старший викладач	_____	Андрій Кондратюк
(посада)	(підпис)	(ініціали, прізвище)

“ *10* ” *червня* 2026р.

РЕЦЕНЗІЯ

на кваліфікаційну роботу

студента Стасюка Вадима Вікторовича гр. ІК-402
(прізвище, ініціали)

на тему: «Розробка програмного засобу для виявлення фішингових електронних листів із використанням методів штучного інтелекту»

Актуальність теми: Тема кваліфікаційної роботи є актуальною, оскільки фішингові електронні листи залишаються одним із найпоширеніших інструментів реалізації кіберзагроз. Щороку кількість атак, спрямованих на викрадення конфіденційної інформації користувачів через електронну пошту, зростає, а методи соціальної інженерії стають дедалі складнішими. У таких умовах особливої актуальності набуває розробка програмних засобів, здатних автоматично аналізувати електронні повідомлення та виявляти потенційно небезпечні листи із використанням сучасних технологій штучного інтелекту.

Наукова новизна: Науковий інтерес становить запропонований підхід до виявлення фішингових електронних листів, який поєднує репутаційний аналіз вкладень і вебпосилань із використанням великих мовних моделей. Автором реалізовано механізм інтеграції сервісу VirusTotal для перевірки вкладень та URL-адрес, а також платформу OpenRouter для інтелектуального аналізу змісту повідомлень. Такий підхід дозволяє підвищити якість виявлення сучасних фішингових атак, що не завжди можуть бути визначені традиційними методами.

Якість проведеного аналізу: Проведене дослідження свідчить про належний рівень підготовки здобувача. У роботі виконано аналіз сучасних фішингових атак, розглянуто методи їх виявлення, досліджено особливості використання сервісу VirusTotal та великих мовних моделей для аналізу електронних повідомлень. Матеріал викладено логічно та послідовно, а отримані результати відповідають поставленій меті дослідження.

Уміння користуватися літературними джерелами: Під час виконання кваліфікаційної роботи автор використав достатню кількість наукових, навчальних та технічних джерел. Особливу увагу приділено сучасним підходам до виявлення фішингових повідомлень, використанню технологій штучного інтелекту та документації програмних інтерфейсів, які застосовувалися під час розробки програмного засобу. Джерела опрацьовано системно та використано для обґрунтування прийнятих рішень.

Практична цінність висновків і рекомендацій: Практична цінність роботи полягає у розробці функціонального програмного засобу, який забезпечує автоматизоване отримання електронних листів через протокол IMAP, перевірку вкладень та URL-адрес за допомогою VirusTotal, а також інтелектуальний аналіз змісту повідомлень із використанням сучасних мовних моделей. Розроблений програмний продукт може бути використаний для підвищення рівня захисту електронної пошти як окремих користувачів, так і невеликих організацій.

Переваги та недоліки: До переваг роботи слід віднести актуальність тематики, поєднання теоретичного дослідження із практичною реалізацією програмного продукту, використання сучасних сервісів аналізу кіберзагроз та технологій штучного інтелекту. Позитивне враження справляє реалізований механізм комплексного аналізу електронних листів, який враховує як технічні характеристики повідомлення, так і його змістове наповнення. Разом із тим у роботі доцільно було б більш детально дослідити кількісні показники ефективності розробленого програмного засобу на розширеній вибірці електронних повідомлень та провести порівняльний аналіз із існуючими комерційними рішеннями. Проте зазначені зауваження не знижують загальної позитивної оцінки виконаної роботи та мають рекомендаційний характер.

Загальний висновок і оцінка роботи: Кваліфікаційна робота Стасюка Вадима Вікторовича виконана на належному науково-технічному рівні та відповідає вимогам, які висуваються до бакалаврських робіт за спеціальністю 125

«Кібербезпека та захист інформації». Автор продемонстрував знання сучасних методів виявлення кіберзагроз, уміння застосовувати технології штучного інтелекту для розв'язання практичних завдань та навички розробки програмного забезпечення. Вважаю, що кваліфікаційна робота заслуговує на високу оцінку та рекомендується до захисту в Екзаменаційній комісії.

Рецензент

к.т.н. доцент, завідувач кафедри інженерії програмної інженерії та технологій ім. Тараса Шевченка

(посада)

О.Є. Гладкова

(підпис) (ініціали, прізвище)

Печатка

“___”_____20__ р.

ЗМІСТ

ВСТУП

У сучасному цифровому середовищі електронна пошта є одним із найбільш поширених засобів комунікації між користувачами, організаціями та державними установами. Водночас вона залишається одним із головних каналів реалізації кіберзагроз, серед яких особливе місце займають фішингові атаки. Фішинг являє собою вид соціальної інженерії, спрямований на отримання конфіденційної інформації користувачів шляхом маскування під легітимні сервіси, організації або окремих осіб. Зловмисники використовують фішингові електронні листи для викрадення облікових даних, банківської інформації, персональних даних та розповсюдження шкідливого програмного забезпечення.

Актуальність теми дослідження обумовлена постійним зростанням кількості фішингових атак та вдосконаленням методів їх реалізації. Сучасні фішингові повідомлення все частіше створюються із використанням автоматизованих інструментів і технологій штучного інтелекту, що значно ускладнює їх виявлення традиційними засобами захисту. Більшість існуючих систем фільтрації електронної пошти базується на сигнатурному або евристичному аналізі, який не завжди дозволяє ефективно виявляти нові та модифіковані фішингові кампанії. У зв'язку з цим актуальним завданням є розробка програмних засобів, що поєднують класичні методи аналізу електронних повідомлень із сучасними методами штучного інтелекту.

Перспективним напрямом підвищення ефективності виявлення фішингових листів є використання великих мовних моделей, здатних аналізувати зміст повідомлень, контекст листування, характер посилань та вкладень, а також формувати аргументовані висновки щодо рівня потенційної небезпеки повідомлення. Додаткове використання спеціалізованих сервісів аналізу репутації файлів та вебресурсів дозволяє підвищити достовірність результатів перевірки та зменшити кількість помилкових спрацьовувань.

Метою роботи є розробка програмного засобу для виявлення фішингових електронних листів із використанням методів штучного інтелекту, який забезпечує

автоматизований аналіз повідомлень електронної пошти, перевірку вкладень та гіперпосилань, а також формування висновку щодо рівня їхньої небезпеки.

Для досягнення поставленої мети необхідно вирішити такі завдання:

– проаналізувати сучасні фішингові атаки в електронній пошті та методи їх виявлення;

– дослідити існуючі програмні засоби захисту від фішингових повідомлень;

– проаналізувати можливості використання методів штучного інтелекту для виявлення фішингових електронних листів;

– дослідити принципи аналізу вкладень та гіперпосилань за допомогою спеціалізованих сервісів кібербезпеки;

– спроєктувати архітектуру програмного засобу для аналізу електронної пошти;

– реалізувати модуль отримання та обробки електронних листів;

– реалізувати модуль перевірки вкладень та гіперпосилань;

– реалізувати модуль інтелектуального аналізу електронних листів;

– провести тестування розробленого програмного засобу та оцінити ефективність його роботи.

Об'єктом дослідження є процес виявлення фішингових електронних листів у системах електронної пошти.

Предметом дослідження є методи, алгоритми та програмні засоби аналізу електронних листів із використанням технологій штучного інтелекту, сервісів перевірки репутації вебресурсів і вкладень.

У роботі використано такі методи дослідження: аналіз наукових джерел та сучасних підходів до виявлення фішингу, методи структурного аналізу та проєктування програмних систем, методи обробки електронних повідомлень, методи аналізу репутації вебресурсів та файлів, а також методи штучного інтелекту для класифікації та оцінювання рівня небезпеки електронних листів.

Практичне значення одержаних результатів полягає у розробці програмного засобу, який дозволяє автоматизувати процес перевірки електронних листів на наявність ознак фішингу, здійснювати аналіз вкладень і гіперпосилань, а також

надавати користувачу детальний протокол перевірки з поясненням виявлених загроз. Розроблений програмний засіб може використовуватися як додатковий інструмент підвищення рівня кібербезпеки користувачів електронної пошти.

Кваліфікаційна робота складається зі вступу, трьох розділів, висновків, списку використаних джерел та додатків. У першому розділі проведено аналіз фішингових атак та сучасних методів їх виявлення. У другому розділі розглянуто теоретичні основи побудови систем виявлення фішингових електронних листів із використанням методів штучного інтелекту. У третьому розділі наведено опис розробленого програмного засобу, особливості його реалізації та результати практичного тестування.

РОЗДІЛ 1. АНАЛІЗ ФІШИНГОВИХ АТАК В ЕЛЕКТРОННІЙ ПОШТІ ТА МЕТОДІВ ЇХ ВИЯВЛЕННЯ

1.1 Поняття та сутність фішингових електронних листів

Стрімкий розвиток інформаційних технологій та широке використання електронної пошти в особистій і професійній діяльності сприяли не лише покращенню комунікаційних процесів, але й появі нових кіберзагроз. Однією з найпоширеніших загроз інформаційній безпеці є фішинг, який належить до категорії атак соціальної інженерії та спрямований на отримання конфіденційної інформації користувачів шляхом обману. [12; 13]

Термін «фішинг» походить від англійського слова phishing, що є видозміненим написанням слова fishing – «рибальство». Аналогія полягає в тому, що зловмисник за допомогою спеціально підготовлених повідомлень намагається «виловити» у користувача конфіденційні дані, зокрема логіни, паролі, реквізити банківських карток, персональну інформацію або інші відомості, що можуть бути використані для несанкціонованого доступу до інформаційних ресурсів.

Фішинговий електронний лист являє собою повідомлення, яке імітує лист від відомої організації, державної установи, банку, поштового сервісу або іншого довіреного джерела. Основною метою такого повідомлення є спонукання користувача виконати певну дію: перейти за посиланням, завантажити файл, ввести облікові дані або надати конфіденційну інформацію.

Особливістю сучасних фішингових листів є високий рівень правдоподібності. Зловмисники використовують офіційні логотипи компаній, копіюють дизайн легітимних повідомлень, підробляють адреси відправників та застосовують психологічні методи впливу на користувачів. Найчастіше у змісті таких листів використовується фактор терміновості, повідомлення про блокування облікового запису, необхідність підтвердження особистих даних, отримання виграшу або виконання фінансових операцій. [4; 15]

Типовий механізм реалізації фішингової атаки наведено на рисунку 1.1.

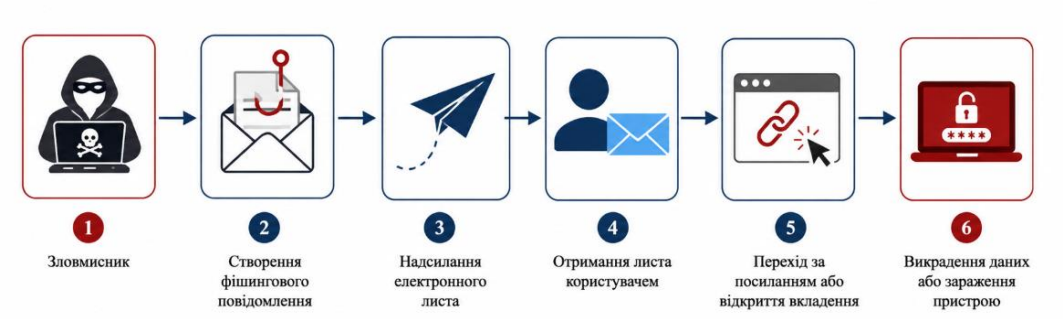


Рисунок 1.1 – Загальна схема реалізації фішингової атаки електронною поштою

Значну небезпеку становлять фішингові листи, що містять вкладення. Як правило, вкладення можуть містити шкідливе програмне забезпечення, макроси для офісних документів або архіви із прихованими виконуваними файлами. Після відкриття таких вкладень на пристрій користувача може бути встановлено програмне забезпечення для викрадення даних, віддаленого керування системою або шифрування файлів.

Іншим поширеним методом є використання підроблених вебресурсів. У такому випадку електронний лист містить гіперпосилання на вебсторінку, зовнішній вигляд якої практично повністю повторює оригінальний сайт банку, поштового сервісу чи іншої організації. Користувач, не підозрюючи про загрозу, вводить свої облікові дані, які одразу передаються зловмиснику.

Для підвищення ефективності атак кіберзлочинці часто використовують методи маскування доменних імен. Зокрема, застосовуються домени, що зовні нагадують адреси відомих сервісів, відрізняючись лише однією або кількома літерами. Такий підхід отримав назву typosquatting та широко використовується під час проведення фішингових кампаній. [15; 17]

Незважаючи на розвиток засобів захисту електронної пошти, фішингові повідомлення залишаються ефективним інструментом атак через використання людського фактора. Навіть сучасні механізми фільтрації не завжди здатні виявляти

нові або модифіковані фішингові кампанії, що робить актуальним застосування інтелектуальних методів аналізу повідомлень.

Таким чином, фішингові електронні листи є серйозною загрозою для інформаційної безпеки користувачів та організацій. Вони можуть використовуватися для викрадення конфіденційної інформації, розповсюдження шкідливого програмного забезпечення та отримання несанкціонованого доступу до інформаційних систем. Ефективне виявлення таких повідомлень потребує комплексного підходу, який поєднує класичні методи аналізу та сучасні технології штучного інтелекту.

1.2 Основні типи та механізми реалізації фішингових атак

З розвитком інформаційних технологій та засобів захисту кіберзлочинці постійно вдосконалюють методи проведення фішингових атак. Якщо раніше фішингові кампанії були орієнтовані переважно на масову розсилку однотипних повідомлень, то сьогодні вони дедалі частіше використовують персоналізований підхід, елементи соціальної інженерії та автоматизовані засоби формування контенту. Це призводить до появи нових різновидів фішингових атак та ускладнює процес їх виявлення. [4; 13]

Залежно від способу реалізації та цільової аудиторії фішингові атаки можна поділити на декілька основних категорій.

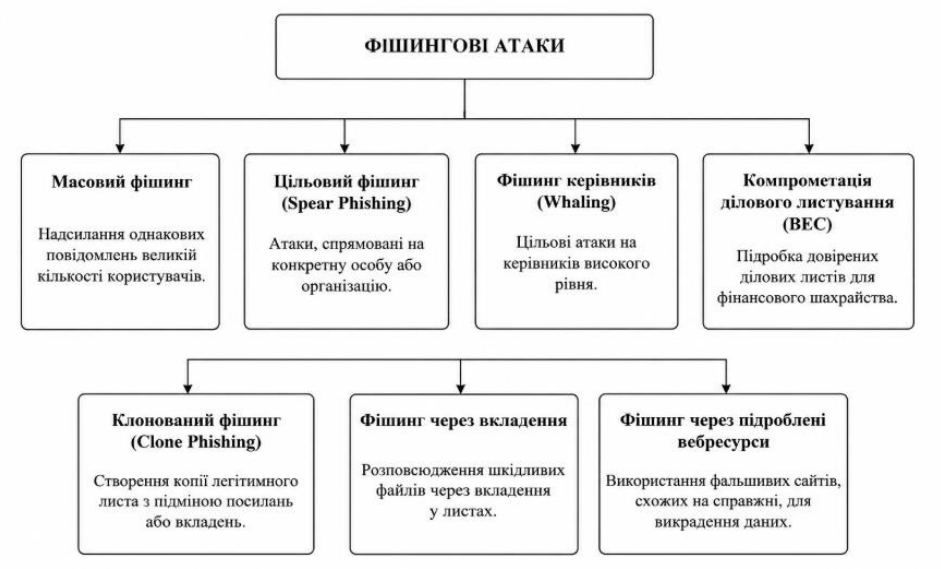


Рисунок 1.2 – Класифікація фішингових атак

Найбільш поширеним різновидом є масовий фішинг. У цьому випадку зловмисники здійснюють розсилку великої кількості електронних листів без попереднього вивчення потенційних жертв. Такі повідомлення зазвичай містять інформацію про блокування акаунта, необхідність підтвердження особистих даних, виграш у лотереї або інші стимули, що спонукають користувача до виконання певних дій.

Цільовий фішинг (Spear Phishing) характеризується більш високим рівнем персоналізації. Перед проведенням атаки зловмисники збирають інформацію про конкретну особу або організацію з відкритих джерел, соціальних мереж чи інших ресурсів. Отримані відомості використовуються для створення повідомлення, яке максимально нагадує легітимне службове або особисте листування. Завдяки цьому ймовірність успішної атаки суттєво зростає.

Особливим різновидом цільового фішингу є атаки типу Whaling. У цьому випадку основною ціллю виступають керівники компаній, власники бізнесу, директори та інші особи, що мають розширені права доступу до інформаційних ресурсів організації. Успішна реалізація такої атаки може призвести до значних фінансових збитків або витоку конфіденційної інформації. [12; 15]

Окрему категорію становить компрометація ділового листування (Business Email Compromise, BEC). Під час таких атак зловмисник видає себе за представника керівництва компанії, партнера або контрагента та надсилає повідомлення із вимогою здійснити фінансовий переказ, надати доступ до корпоративних ресурсів або передати важливу документацію. Особливістю цього виду атак є відсутність шкідливих вкладень або посилань, що значно ускладнює їх автоматичне виявлення.

Клонований фішинг (Clone Phishing) полягає у створенні копії реального електронного листа, який раніше був надісланий користувачу. При цьому зловмисник замінює вкладення або гіперпосилання на шкідливі аналоги, після чого повторно надсилає повідомлення від імені довіреного відправника. Оскільки зовнішній вигляд листа практично не відрізняється від оригіналу, користувачі часто не помічають підробку.

Значну небезпеку становлять фішингові атаки із використанням вкладень. У таких повідомленнях можуть міститися виконувані файли, архіви, документи Microsoft Office з макросами або PDF-файли із прихованими шкідливими сценаріями. Після відкриття вкладення відбувається зараження пристрою користувача або викрадення його даних.

Ще одним поширеним механізмом є використання підроблених вебресурсів. У листі розміщується гіперпосилання на вебсторінку, яка зовні повторює дизайн легітимного сайту банку, поштового сервісу або соціальної мережі. Після введення облікових даних користувачем інформація передається безпосередньо зловмисникам.

Для підвищення ефективності атак кіберзлочинці активно використовують різноманітні методи приховування шкідливої активності. До таких методів належать скорочення URL-адрес, використання схожих доменних імен, застосування міжнародних доменів із символами, візуально схожими на латинські літери, а також використання багаторівневих перенаправлень між вебресурсами.

Тип атаки	Характеристика	Основна мета атаки
Масовий фішинг	Надсилання великої кількості однотипних повідомлень широкому колу користувачів.	Отримання логінів, паролів та іншої конфіденційної інформації.
Цільовий фішинг (Spear Phishing)	Формування персоналізованих повідомлень для конкретного користувача або організації.	Компрометація облікового запису або інформаційних ресурсів.
Фішинг керівників (Whaling)	Атаки, спрямовані на керівний склад організації або інших важливих осіб.	Отримання доступу до критичних корпоративних даних або фінансових операцій.
Компрометація ділового листування (BEC)	Підроблення службових повідомлень від імені довірених осіб всередині або поза організацією.	Проведення шахрайських фінансових операцій та отримання конфіденційної інформації.
Клонований фішинг (Clone Phishing)	Використання копії реального листа із заміною посилань або вкладень на шкідливі.	Поширення шкідливого програмного забезпечення або перенаправлення на фішингові ресурси.
Фішинг через вкладення	Розповсюдження шкідливих файлів через вкладення в електронних листах.	Зараження пристрою шкідливим програмним забезпеченням та отримання несанкціонованого доступу.
Фішинг через підроблені вебресурси	Використання фальшивих вебсторінок, схожих на офіційні ресурси, для введення користувачів в оману.	Викрадення облікових даних користувачів та персональної інформації.

Таблиця 1.1 – Характеристика основних типів фішингових атак

Аналіз сучасних фішингових кампаній свідчить про поступове зміщення акценту від масових атак до більш складних та персоналізованих методів соціальної інженерії. Використання великих мовних моделей та автоматизованих інструментів генерації тексту дозволяє створювати повідомлення високої якості, які практично не містять граматичних помилок та максимально наближені до реального ділового листування. У зв'язку з цим особливої актуальності набуває розробка інтелектуальних систем аналізу електронної пошти, здатних враховувати не лише технічні характеристики повідомлення, але й його змістовне наповнення. [13; 22]

1.3 Аналіз сучасних методів виявлення фішингових електронних листів

Зростання кількості фішингових атак та постійне вдосконалення методів їх реалізації зумовлюють необхідність розробки ефективних засобів виявлення шкідливих електронних повідомлень. Сучасні системи захисту використовують різноманітні методи аналізу листів, які відрізняються принципами роботи, точністю виявлення загроз та обчислювальною складністю. Кожен із методів має власні переваги та недоліки, тому на практиці часто застосовується їх комбінування.

Рисунок 1.3 – Класифікація методів виявлення фішингових електронних листів

Одним із найстаріших підходів є сигнатурний аналіз. Його принцип полягає у порівнянні електронного листа з базою відомих шаблонів, сигнатур або ознак раніше виявлених фішингових повідомлень. Якщо повідомлення містить збіг із відомою сигнатурою, воно позначається як потенційно небезпечне. [16]

Перевагою сигнатурного аналізу є висока швидкість роботи та незначне навантаження на систему. Водночас головним недоліком такого підходу є неможливість ефективного виявлення нових або модифікованих атак, інформація про які ще відсутня в базах сигнатур.

Іншим поширеним методом є евристичний аналіз. На відміну від сигнатурного підходу, він базується на пошуку підозрілих ознак у структурі та змісті повідомлення. До таких ознак можуть належати використання підозрілих доменів, наявність скорочених URL-адрес, невідповідність між відображуваним ім'ям

відправника та реальною адресою, використання термінових закликів до дії або запити конфіденційної інформації. [2; 17]

Евристичний аналіз дозволяє виявляти невідомі раніше фішингові кампанії, проте може генерувати значну кількість помилкових спрацьовувань. Крім того, ефективність такого підходу значною мірою залежить від якості сформованих правил перевірки.

Широкого поширення набули репутаційні методи аналізу. Їх принцип полягає у використанні спеціалізованих баз даних та сервісів, що містять інформацію про репутацію доменів, IP-адрес, вебресурсів та файлів. Якщо об'єкт уже був помічений у шкідливій активності, система може оперативно повідомити про потенційну небезпеку.

Одним із найбільш відомих сервісів репутаційного аналізу є VirusTotal. Він агрегує результати перевірок великої кількості антивірусних рішень та систем кіберзахисту, що дозволяє отримати комплексну оцінку безпечності файлів та вебресурсів. [24]

Подальший розвиток засобів виявлення фішингу привів до використання методів машинного навчання. У таких системах на основі набору попередньо розмічених даних формується модель, здатна автоматично класифікувати повідомлення як безпечні або фішингові. Для навчання можуть використовуватися різноманітні характеристики листів, зокрема текст повідомлення, заголовки, вкладення, URL-адреси та інші параметри.

Основними перевагами методів машинного навчання є можливість автоматичного виявлення прихованих закономірностей та адаптація до нових типів атак. Водночас для побудови якісної моделі необхідна велика кількість навчальних даних, а процес навчання може вимагати значних обчислювальних ресурсів.

Останніми роками особливу увагу привертають методи штучного інтелекту, засновані на використанні великих мовних моделей. Такі моделі здатні аналізувати зміст повідомлень на семантичному рівні, оцінювати логіку викладення тексту, виявляти маніпулятивні прийоми та формувати обґрунтовані висновки щодо рівня небезпеки листа.

На відміну від класичних підходів, великі мовні моделі враховують контекст повідомлення, що дозволяє ефективніше виявляти складні та персоналізовані фішингові кампанії. Крім того, вони можуть аналізувати результати роботи інших механізмів захисту та використовувати їх під час прийняття рішення.

Сучасною тенденцією розвитку систем кіберзахисту є використання гібридних підходів, які поєднують декілька методів аналізу. У таких системах результати сигнатурної перевірки, евристичного аналізу, репутаційних сервісів та методів штучного інтелекту об'єднуються для формування єдиного висновку щодо рівня небезпеки повідомлення.

Метод	Переваги	Недоліки
Сигнатурний	Висока швидкість роботи	Не виявляє нові атаки
Евристичний	Виявлення невідомих загроз	Помилкові спрацьовування
Репутаційний	Висока точність перевірки відомих загроз	Залежність від зовнішніх баз даних
Машинне навчання	Адаптивність до нових атак	Необхідність навчальних вибірок
Штучний інтелект	Аналіз контексту та змісту листа	Високі вимоги до ресурсів
Гібридний	Найвища ефективність виявлення	Складність реалізації

Таблиця 1.2 – Порівняльна характеристика методів виявлення фішингу

Аналіз сучасних методів виявлення фішингових електронних листів показує, що жоден із підходів окремо не забезпечує достатнього рівня захисту від усіх видів атак. Найбільш перспективним напрямом є використання гібридних систем, які поєднують переваги різних методів аналізу. Саме такий підхід покладено в основу розроблюваного програмного засобу, який використовує репутаційний аналіз через

сервіс VirusTotal та інтелектуальний аналіз вмісту листів за допомогою великих мовних моделей. [19; 22]

1.4 Аналіз існуючих програмних засобів виявлення фішингових електронних листів

Зростання кількості фішингових атак сприяло появі значної кількості програмних засобів та сервісів, призначених для захисту електронної пошти. Сучасні рішення використовують різноманітні підходи до виявлення загроз, зокрема сигнатурний аналіз, евристичні правила, репутаційні бази даних, технології машинного навчання та методи штучного інтелекту. Аналіз існуючих програмних продуктів дозволяє визначити їх сильні та слабкі сторони, а також обґрунтувати необхідність створення власного програмного засобу.

Одним із найбільш поширених рішень є Microsoft Defender for Office 365. Система забезпечує захист корпоративної електронної пошти від шкідливих вкладень, фішингових повідомлень та небезпечних вебпосилань. Для виявлення загроз використовуються механізми аналізу репутації, поведінковий аналіз та технології машинного навчання. Перевагою рішення є глибока інтеграція з екосистемою Microsoft 365 та централізоване управління політиками безпеки. Недоліком є висока вартість використання та орієнтація переважно на корпоративний сектор.

Значною популярністю користується система захисту Gmail Security, що використовується в поштовому сервісі Gmail. Для аналізу повідомлень застосовуються алгоритми машинного навчання, які дозволяють автоматично класифікувати електронні листи та виявляти потенційно небезпечні повідомлення. Основною перевагою є високий рівень автоматизації процесу захисту. Разом із тим механізми аналізу залишаються закритими для користувача, а детальна інформація щодо прийнятого рішення зазвичай недоступна.

Ще одним відомим рішенням є Proofpoint Email Protection. Дана система забезпечує комплексний захист від фішингових атак, шкідливого програмного

забезпечення та спроб компрометації ділового листування. Для виявлення загроз використовуються технології штучного інтелекту, аналіз поведінки користувачів та механізми перевірки репутації доменів і вебресурсів. Основним недоліком системи є складність налаштування та висока вартість впровадження.

Широкого застосування набуло рішення Mimecast Email Security. Платформа забезпечує багаторівневий аналіз електронних листів, перевірку вкладень, захист від шкідливих URL та аналіз аномальної поведінки користувачів. Перевагою є комплексний підхід до кіберзахисту, однак ефективне використання системи потребує відповідного рівня підготовки адміністраторів.

Серед популярних рішень також варто відзначити Barracuda Email Protection. Система забезпечує захист від спаму, фішингу та шкідливих вкладень. Для аналізу повідомлень використовуються евристичні правила, репутаційні механізми та елементи штучного інтелекту. Основною перевагою є відносна простота впровадження, проте функціональні можливості системи поступаються більш складним корпоративним рішенням.

Окремої уваги заслуговує сервіс VirusTotal, який широко використовується фахівцями з кібербезпеки для перевірки файлів, вебресурсів та інших об'єктів. VirusTotal не є повноцінною системою захисту електронної пошти, проте надає можливість отримувати результати аналізу від великої кількості антивірусних рушіїв та засобів кіберзахисту. Завдяки цьому сервіс часто використовується як додаткове джерело інформації під час аналізу потенційно небезпечних повідомлень. [24; 25]

Незважаючи на високу ефективність сучасних комерційних рішень, більшість із них мають певні обмеження. Частина систем орієнтована виключно на корпоративне використання та потребує значних фінансових витрат. Інші рішення працюють за принципом «чорної скриньки», не надаючи користувачу детального пояснення причин прийняття рішення щодо безпечності повідомлення. Крім того, багато продуктів використовують переважно сигнатурні або репутаційні механізми захисту, що може знижувати ефективність виявлення нових та модифікованих фішингових кампаній.

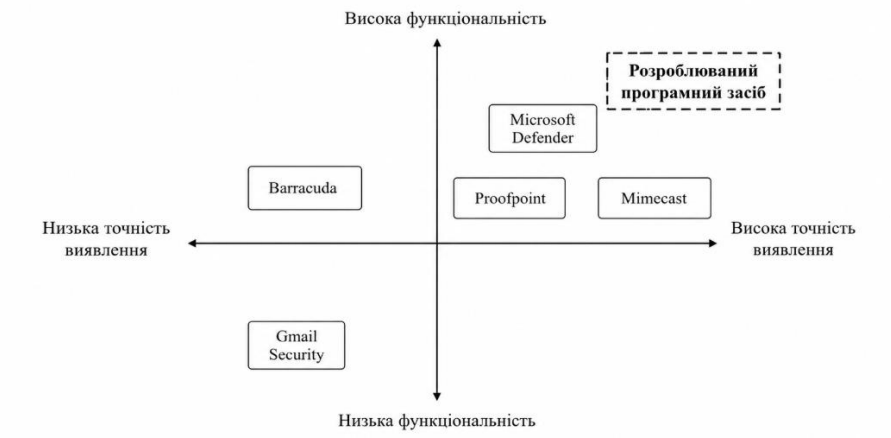


Рисунок 1.4 – Місце розроблюваного програмного засобу серед існуючих рішень

Основною особливістю розроблюваного програмного засобу є використання гібридного підходу до аналізу електронних листів. Система поєднує перевірку вкладень та гіперпосилань за допомогою сервісу VirusTotal із подальшим аналізом змісту повідомлення великими мовними моделями через OpenRouter. На відміну від багатьох існуючих рішень, користувач отримує не лише підсумковий вердикт, але й детальний протокол перевірки з описом виявлених ознак потенційної загрози. [22; 24]

Проведений аналіз існуючих програмних засобів показав, що сучасні рішення забезпечують високий рівень захисту електронної пошти, проте мають низку обмежень, пов'язаних із вартістю впровадження, закритістю алгоритмів або недостатньою пояснюваністю результатів аналізу. Це обумовлює доцільність розробки програмного засобу, який поєднує можливості репутаційного аналізу та методів штучного інтелекту для підвищення ефективності виявлення фішингових електронних листів.

Висновки до розділу 1

У першому розділі було проведено аналіз сучасних фішингових атак в електронній пошті та методів їх виявлення. Встановлено, що фішинг залишається

однією з найбільш поширених загроз інформаційній безпеці, оскільки поєднує технічні засоби впливу з методами соціальної інженерії. Основною метою фішингових атак є отримання конфіденційної інформації користувачів, зокрема облікових даних, фінансової інформації та персональних відомостей.

Розглянуто основні типи фішингових атак, серед яких масовий фішинг, цільовий фішинг (Spear Phishing), атаки на керівництво організацій (Whaling), компрометація ділового листування (Business Email Compromise), клонований фішинг, а також атаки із використанням шкідливих вкладень та підроблених вебресурсів. Визначено, що сучасні фішингові кампанії характеризуються високим рівнем персоналізації та складністю виявлення, що значно підвищує їх ефективність.

Проведено аналіз сучасних методів виявлення фішингових електронних листів. Встановлено, що сигнатурні методи забезпечують високу швидкість роботи, однак не дозволяють ефективно виявляти нові види загроз. Евристичні методи розширюють можливості аналізу за рахунок використання набору правил та ознак, проте можуть генерувати помилкові спрацьовування. Репутаційні методи забезпечують перевірку файлів, доменів та вебресурсів на основі накопичених даних про відомі загрози. Методи машинного навчання та штучного інтелекту дозволяють аналізувати зміст повідомлень і виявляти складні фішингові атаки, які не можуть бути визначені традиційними засобами захисту.

У результаті аналізу існуючих програмних засобів захисту електронної пошти встановлено, що сучасні комерційні рішення забезпечують високий рівень безпеки, проте часто мають обмеження, пов'язані з високою вартістю впровадження, закритістю алгоритмів аналізу або недостатньою деталізацією результатів перевірки. Крім того, більшість систем не надають користувачеві повного пояснення причин віднесення повідомлення до категорії безпечних або небезпечних.

На основі проведеного аналізу зроблено висновок, що найбільш перспективним напрямом є використання гібридного підходу до виявлення фішингових електронних листів. Такий підхід передбачає поєднання репутаційного

аналізу вкладень і вебпосилань із використанням методів штучного інтелекту для аналізу змісту повідомлень та формування обґрунтованого висновку щодо рівня їхньої небезпеки.

Отримані результати аналізу підтверджують доцільність розробки програмного засобу для виявлення фішингових електронних листів із використанням методів штучного інтелекту, що дозволить підвищити ефективність виявлення сучасних кіберзагроз та забезпечити користувача детальною інформацією про результати перевірки електронних повідомлень.

РОЗДІЛ 2. ТЕОРЕТИЧНІ ОСНОВИ ВИЯВЛЕННЯ ФІШИНГОВИХ ЕЛЕКТРОННИХ ЛИСТІВ ІЗ ВИКОРИСТАННЯМ МЕТОДІВ ШТУЧНОГО ІНТЕЛЕКТУ

2.1 Архітектура систем аналізу електронної пошти

Електронна пошта є одним із найбільш поширених засобів обміну інформацією в сучасному інформаційному суспільстві. Вона використовується як для особистого спілкування, так і для організації бізнес-процесів, обміну службовими документами та взаємодії між інформаційними системами. Водночас електронна пошта залишається одним із головних каналів поширення кіберзагроз, серед яких особливе місце займають фішингові повідомлення, шкідливі вкладення та посилання на небезпечні вебресурси.

Для забезпечення належного рівня інформаційної безпеки використовуються спеціалізовані системи аналізу електронної пошти. Основним завданням таких систем є автоматизоване виявлення потенційно небезпечних повідомлень до того, як користувач виконає будь-які дії, що можуть призвести до компрометації облікового запису або зараження комп'ютера шкідливим програмним забезпеченням.

Першим елементом системи є модуль отримання повідомлень. Його функцією є встановлення з'єднання з поштовим сервером, автентифікація користувача та завантаження електронних листів для подальшого аналізу. Для цього найчастіше використовуються стандартні поштові протоколи IMAP або POP3. У сучасних системах перевага надається протоколу IMAP, оскільки він забезпечує можливість роботи з повідомленнями без необхідності їх локального зберігання та підтримує синхронізацію між різними пристроями. [18]

Після отримання повідомлення виконується його попередня обробка. На цьому етапі система виділяє основні елементи листа, зокрема адресу відправника, тему повідомлення, текстовий вміст, HTML-вміст, вкладення та гіперпосилання. Також можуть аналізуватися службові заголовки електронної пошти, такі як Reply-

То, Return-Path та інші параметри, що містять інформацію про походження повідомлення.

Наступним компонентом є модуль аналізу. Саме він виконує основну перевірку повідомлення на наявність ознак фішингової активності. Залежно від архітектури системи можуть використовуватися різні підходи до аналізу: сигнатурний, евристичний, репутаційний, статистичний або інтелектуальний. У сучасних рішеннях найчастіше застосовується комбінований підхід, що дозволяє підвищити точність виявлення загроз.

Особливу роль у процесі аналізу відіграє перевірка вкладень та гіперпосилань. Вкладення можуть містити шкідливі програми, макроси або сценарії, а посилання можуть вести на підроблені вебресурси, призначені для викрадення конфіденційної інформації користувачів. Тому сучасні системи аналізу електронної пошти активно використовують зовнішні сервіси репутаційної перевірки для оцінювання рівня безпеки файлів та URL-адрес. [24]

Після завершення аналізу система формує результат перевірки. Результатом може бути класифікація повідомлення як безпечного, підозрілого або небезпечного. У деяких системах користувач також отримує додаткову інформацію щодо причин прийняття відповідного рішення, переліку виявлених ознак та рекомендацій щодо подальших дій.

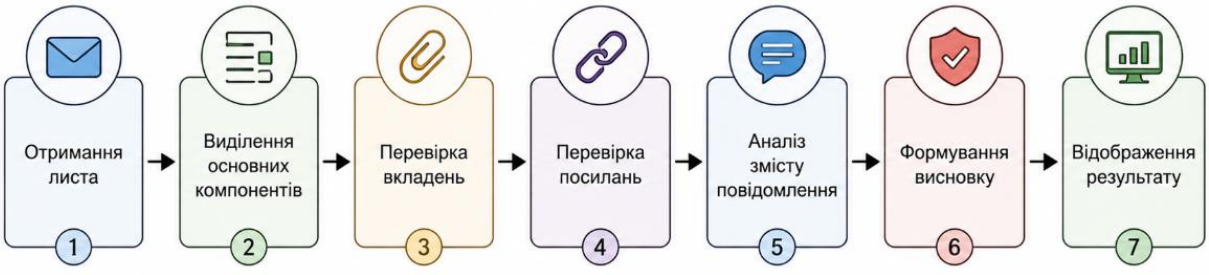


Рисунок 2.1 – Послідовність обробки електронного листа в системі аналізу

Останніми роками архітектура систем аналізу електронної пошти зазнала значних змін у зв'язку з активним впровадженням методів штучного інтелекту. Якщо раніше основна увага приділялася аналізу технічних характеристик

повідомлення, то сучасні рішення здатні аналізувати контекст повідомлення, стиль викладення тексту, логіку побудови звернень та інші складові, які складно формалізувати традиційними методами. [22]

Таким чином, сучасна система аналізу електронної пошти являє собою багаторівневий комплекс програмних компонентів, що забезпечують отримання, обробку та перевірку повідомлень на наявність ознак фішингової активності. Ефективність таких систем значною мірою залежить від використання комплексного підходу, який поєднує технічний аналіз повідомлень, перевірку репутації вкладень і вебресурсів, а також інтелектуальний аналіз змісту електронних листів.

2.2 Аналіз ознак фішингових електронних листів

Ефективність виявлення фішингових електронних листів значною мірою залежить від правильного визначення ознак, які характеризують потенційно небезпечне повідомлення. Під ознаками фішингового листа розуміють сукупність технічних, структурних та змістовних характеристик повідомлення, які можуть свідчити про спробу введення користувача в оману з метою отримання конфіденційної інформації або поширення шкідливого програмного забезпечення.

Сучасні системи аналізу електронної пошти використовують комплексний підхід до виявлення таких ознак. При цьому аналізуються як технічні параметри повідомлення, так і його змістовне наповнення.

Рисунок 2.2 – Основні ознаки фішингових електронних листів

Однією з найважливіших ознак є адреса відправника. Зловмисники часто використовують домени, які візуально нагадують домени відомих організацій або сервісів. Такі домени можуть відрізнятися лише декількома символами, що ускладнює їх візуальне виявлення користувачем. Подібна техніка отримала назву typosquatting та широко застосовується під час проведення фішингових кампаній. [15]

Важливою характеристикою є відповідність між відображуваним ім'ям відправника та реальною електронною адресою. Зловмисники часто використовують назви відомих компаній або організацій, приховуючи справжню адресу відправника. У результаті користувач може помилково вважати повідомлення легітимним.

Значну роль у виявленні фішингових повідомлень відіграє аналіз теми листа. Фішингові повідомлення часто містять слова та фрази, що створюють відчуття терміновості або небезпеки. До таких формулювань належать повідомлення про блокування акаунта, необхідність негайного підтвердження особистих даних, підозрілі фінансові операції або отримання виграшу.

Окрему категорію становлять ознаки, пов'язані зі змістом повідомлення. Фішингові листи часто містять маніпулятивні формулювання, які спонукають користувача діяти швидко та без додаткової перевірки отриманої інформації. Для

цього використовуються психологічні прийоми, засновані на страху, терміновості, зацікавленості або довірі до авторитетних організацій.

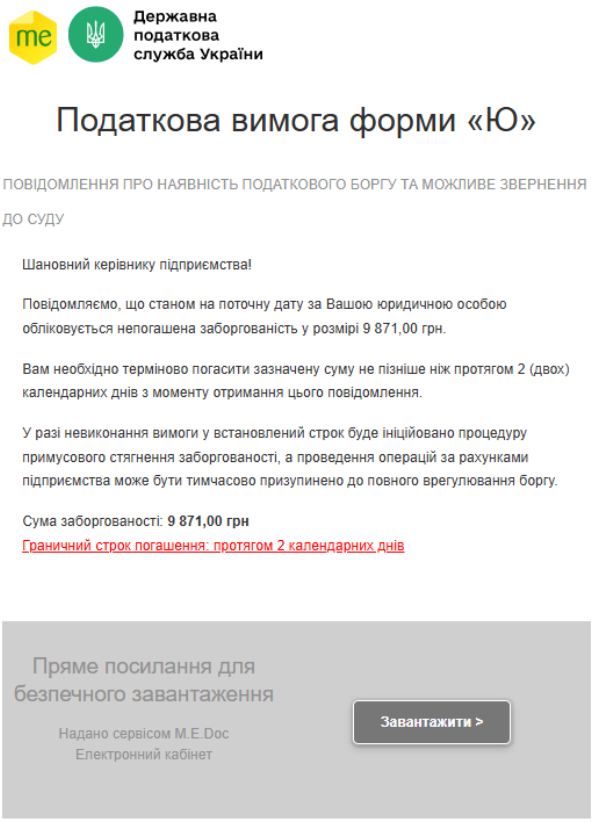


Рисунок 2.3 – Приклад електронного листа з ознаками фішингу

Однією з найнебезпечніших ознак є наявність запиту конфіденційної інформації. У більшості випадків легітимні організації не запитують паролі, реквізити банківських карток, CVV-коди, паспортні дані або іншу конфіденційну інформацію через електронну пошту. Наявність таких запитів є серйозним індикатором потенційної фішингової активності. [12]

Важливим джерелом інформації є гіперпосилання, що містяться у повідомленні. Під час аналізу URL-адрес враховуються доменне ім'я, використання IP-адрес замість доменів, наявність підозрілих доменних зон, багаторівневі перенаправлення та інші ознаки, що можуть свідчити про спробу приховування реального призначення вебресурсу. [24]

Особливу увагу необхідно приділяти вкладенням. Фішингові повідомлення часто містять виконувані файли, архіви, документи з макросами або інші потенційно

небезпечні об'єкти. Відкриття таких вкладень може призвести до встановлення шкідливого програмного забезпечення, викрадення даних або компрометації інформаційної системи.

Додатковим джерелом інформації є аналіз службових заголовків електронної пошти. Зокрема, перевіряються поля Reply-To та Return-Path. Невідповідність зазначених адрес реальній адресі відправника може свідчити про підроблення повідомлення або використання проміжних поштових серверів для приховування справжнього джерела розсилки.

Під час аналізу змісту повідомлень також враховуються мовні особливості тексту. Фішингові повідомлення можуть містити граматичні помилки, змішування різних мов, незвичні конструкції речень або невластиві офіційному листуванню формулювання. Хоча сучасні фішингові кампанії дедалі частіше використовують якісно сформований текст, мовний аналіз залишається важливим джерелом додаткових ознак.

Ознака	Характеристика
Адреса відправника	Використання схожих або підроблених доменів
Ім'я відправника	Невідповідність між назвою та реальною адресою
Тема листа	Використання термінових або маніпулятивних повідомлень
URL-адреси	Підозрілі домени, IP-адреси, скорочені посилання
Вкладення	Виконувані файли, архіви, документи з макросами
Запит даних	Вимога надати пароль або іншу конфіденційну інформацію
Reply-To	Невідповідність адресі відправника
Return-Path	Аномалії маршрутизації повідомлення
Мовні особливості	Граматичні помилки або змішування мов

Таблиця 2.1 – Характеристика ознак фішингових електронних листів

Таким чином, виявлення фішингових електронних листів ґрунтується на аналізі великої кількості різнорідних ознак. Окремі характеристики не завжди дозволяють однозначно визначити рівень небезпеки повідомлення, проте їх комплексне використання значно підвищує точність виявлення потенційних загроз. Саме тому сучасні системи кіберзахисту застосовують багаторівневий підхід до аналізу електронних листів, поєднуючи перевірку технічних параметрів повідомлення з аналізом його змісту та поведінкових характеристик. [16]

2.3 Використання сервісу VirusTotal для аналізу посилань та вкладень

Одним із найбільш ефективних інструментів для перевірки потенційно небезпечних файлів та вебресурсів є сервіс VirusTotal. Даний сервіс широко використовується фахівцями з кібербезпеки, дослідниками інформаційної безпеки та розробниками програмних засобів захисту для аналізу об'єктів на наявність ознак шкідливої активності. [24]

VirusTotal являє собою багатофункціональну платформу, яка агрегує результати роботи великої кількості антивірусних продуктів, систем виявлення загроз та засобів аналізу кіберінцидентів. Завдяки цьому користувач може отримати комплексну інформацію щодо безпечності файлів, URL-адрес, доменів, IP-адрес та інших об'єктів.

Основною перевагою використання VirusTotal є можливість одночасного отримання результатів перевірки від великої кількості незалежних механізмів захисту. Це дозволяє підвищити достовірність аналізу та мінімізувати ризик помилкового визначення безпечного або шкідливого об'єкта.

У системах аналізу електронної пошти сервіс VirusTotal найчастіше використовується для перевірки гіперпосилань та вкладень. Саме ці елементи є основними інструментами реалізації фішингових атак та розповсюдження шкідливого програмного забезпечення.

Під час аналізу URL-адрес система передає посилання до сервісу VirusTotal через програмний інтерфейс API. Після отримання запиту сервіс здійснює перевірку

вебресурсу за власними базами даних та результатами попередніх сканувань. Якщо інформація про ресурс уже наявна в системі, користувач отримує готовий результат перевірки. В іншому випадку запускається новий процес аналізу.

Результатом перевірки URL-адреси є статистична інформація щодо кількості антивірусних рушіїв, які вважають ресурс безпечним, підозрілим або шкідливим. На основі цих даних формується узагальнений висновок щодо рівня потенційної небезпеки вебресурсу.

Не менш важливим є аналіз вкладень. Під час перевірки файлів система використовує криптографічні хеш-функції для формування унікального ідентифікатора файлу. Якщо відповідний файл уже аналізувався раніше, VirusTotal повертає збережені результати перевірки. За відсутності інформації про файл виконується його завантаження до сервісу та запуск нового процесу сканування.

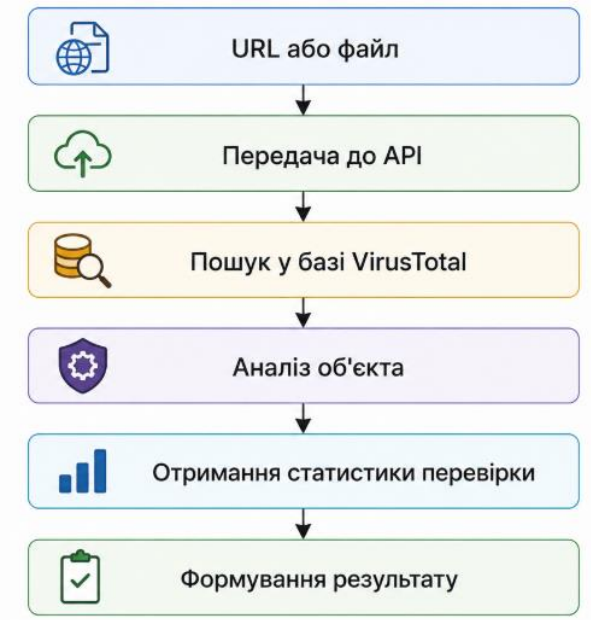


Рисунок 2.4 – Схема перевірки URL та вкладень через VirusTotal

Для інтеграції із зовнішніми програмними системами сервіс VirusTotal надає спеціалізований програмний інтерфейс VirusTotal API. Використання API дозволяє автоматизувати процес перевірки об'єктів та інтегрувати результати аналізу до власних програмних рішень. [25]

Функція API	Призначення
Перевірка URL	Аналіз вебпосилань на наявність загроз
Перевірка файлів	Аналіз вкладень та виконуваних файлів
Отримання статистики	Доступ до результатів перевірки
Перевірка доменів	Аналіз репутації доменних імен
Перевірка IP-адрес	Аналіз мережевих адрес
Отримання історії	Доступ до попередніх результатів сканування

Таблиця 2.2 – Основні можливості VirusTotal API

Використання VirusTotal дозволяє значно підвищити ефективність систем виявлення фішингових повідомлень. На відміну від локальних механізмів перевірки, сервіс забезпечує доступ до глобальної бази знань про кіберзагрози та використовує результати роботи великої кількості незалежних засобів захисту.

Разом із тим використання VirusTotal має певні обмеження. Зокрема, результати аналізу значною мірою залежать від актуальності інформації у базах сервісу. Крім того, перевірка репутації об'єкта не дозволяє оцінити зміст повідомлення або виявити психологічні прийоми соціальної інженерії, що використовуються у фішингових листах.

Таким чином, сервіс VirusTotal є ефективним інструментом репутаційного аналізу посилань та вкладень, який дозволяє виявляти відомі загрози та отримувати детальну інформацію про потенційно небезпечні об'єкти. Проте для забезпечення високої точності виявлення сучасних фішингових атак доцільним є поєднання результатів репутаційного аналізу з методами інтелектуальної обробки змісту електронних повідомлень. [24; 25]

2.4 Використання великих мовних моделей для аналізу електронних листів

Останніми роками одним із найбільш перспективних напрямів розвитку систем кібербезпеки стало використання технологій штучного інтелекту. Особливе місце серед таких технологій займають великі мовні моделі (Large Language Models, LLM), які здатні виконувати складний аналіз текстової інформації, враховуючи не лише окремі слова чи фрази, але й загальний контекст повідомлення. [22]

Великі мовні моделі являють собою нейромережеві системи, навчені на великих обсягах текстових даних. У процесі навчання модель формує уявлення про структуру природної мови, зв'язки між словами та особливості побудови текстів. Завдяки цьому вона може аналізувати зміст повідомлень, визначати їхню тематику, виявляти приховані закономірності та формувати змістовні висновки.

На відміну від традиційних систем аналізу електронної пошти, які переважно працюють із формалізованими правилами та наборами ознак, великі мовні моделі здатні аналізувати повідомлення на семантичному рівні. Це дозволяє враховувати логіку побудови тексту, приховані наміри автора повідомлення та використання маніпулятивних прийомів соціальної інженерії. [23]

Під час аналізу електронних листів великі мовні моделі можуть виконувати декілька важливих функцій. По-перше, вони здатні оцінювати достовірність змісту повідомлення та визначати, наскільки його структура відповідає типовому діловому або офіційному листуванню. По-друге, моделі можуть виявляти психологічний тиск, маніпулятивні формулювання та заклики до негайних дій. По-третє, вони можуть аналізувати взаємозв'язок між окремими елементами листа та формувати комплексну оцінку рівня потенційної небезпеки.

Особливе значення має здатність великих мовних моделей працювати з контекстом повідомлення. Наприклад, лист може не містити очевидних технічних ознак фішингу, проте його зміст може свідчити про спробу соціальної інженерії або несанкціонованого отримання конфіденційної інформації. У таких випадках традиційні механізми захисту можуть не виявити загрозу, тоді як модель штучного інтелекту здатна сформувати більш обґрунтований висновок.

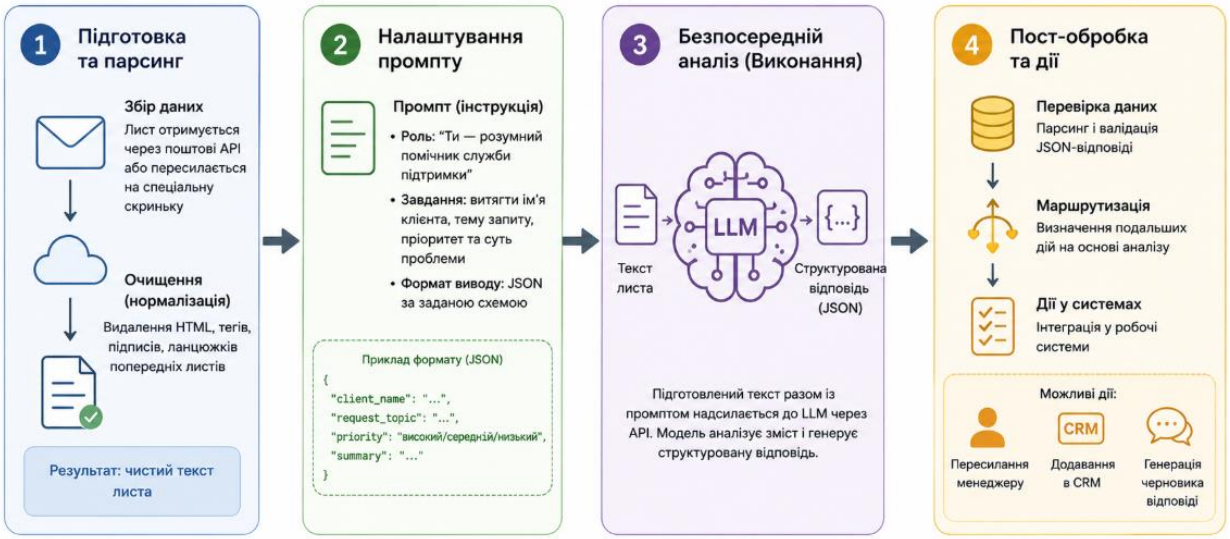


Рисунок 2.5 – Процес аналізу електронного листа за допомогою LLM

Сучасні великі мовні моделі можуть аналізувати не лише текст листа, але й додаткову інформацію, отриману з інших джерел. Наприклад, до моделі можуть передаватися результати перевірки вкладень та гіперпосилань, інформація про відправника, службові заголовки повідомлення та інші характеристики. Такий підхід дозволяє сформувати більш повне уявлення про об'єкт аналізу та підвищити точність класифікації.

Для взаємодії із сучасними мовними моделями широко використовуються спеціалізовані програмні платформи, однією з яких є OpenRouter. Даний сервіс забезпечує єдиний інтерфейс доступу до різних моделей штучного інтелекту та дозволяє інтегрувати можливості великих мовних моделей у власні програмні продукти. [22]

Використання OpenRouter дає можливість гнучко обирати модель для виконання аналізу залежно від поставлених завдань, необхідної точності та доступних обчислювальних ресурсів. Крім того, сервіс забезпечує стандартизований механізм обміну даними між програмним засобом та моделлю штучного інтелекту.

Незважаючи на значні переваги, використання великих мовних моделей має певні обмеження. Зокрема, якість результатів залежить від коректності сформованого запиту, обраної моделі та обсягу наданої інформації. Крім того, для

виконання аналізу необхідний доступ до зовнішніх сервісів та відповідні обчислювальні ресурси.

Таким чином, великі мовні моделі є ефективним інструментом аналізу електронних листів та виявлення складних фішингових атак. Їх використання дозволяє враховувати контекст повідомлення, аналізувати ознаки соціальної інженерії та формувати обґрунтовані висновки щодо рівня потенційної небезпеки листа. Поєднання можливостей великих мовних моделей із репутаційним аналізом посилань та вкладень створює основу для побудови сучасних інтелектуальних систем захисту електронної пошти.

2.5 Постановка задачі розробки програмного засобу

Проведений аналіз сучасних фішингових атак, існуючих методів їх виявлення та наявних програмних засобів захисту електронної пошти показав, що проблема виявлення фішингових електронних листів залишається актуальною. Незважаючи на наявність значної кількості комерційних рішень, багато з них мають певні обмеження, пов'язані зі складністю впровадження, високою вартістю використання або недостатньою прозорістю процесу аналізу повідомлень.

Особливої актуальності набуває проблема виявлення складних фішингових атак, які використовують елементи соціальної інженерії та не містять очевидних технічних ознак шкідливої активності. У таких випадках традиційні методи аналізу не завжди забезпечують необхідний рівень ефективності, що обумовлює доцільність використання методів штучного інтелекту. [22]

Метою розробки є створення програмного засобу, який забезпечує автоматизований аналіз електронних листів, перевірку вкладень та гіперпосилань, використання методів штучного інтелекту для аналізу змісту повідомлень та формування детального протоколу перевірки щодо рівня потенційної небезпеки листа.

Для досягнення поставленої мети програмний засіб повинен забезпечувати виконання таких функцій:

– підключення до поштової скриньки користувача;

– отримання електронних листів із поштового сервера;

– автоматичне оновлення списку повідомлень;

– відображення основної інформації про отримані листи;

– виділення текстового вмісту повідомлень;

– виділення гіперпосилань та вкладень;

– перевірку вебпосилань за допомогою сервісу VirusTotal;

– перевірку вкладень за допомогою сервісу VirusTotal;

– аналіз змісту повідомлення методами штучного інтелекту;

– формування підсумкового висновку щодо рівня небезпеки листа;

– відображення результатів аналізу користувачеві.

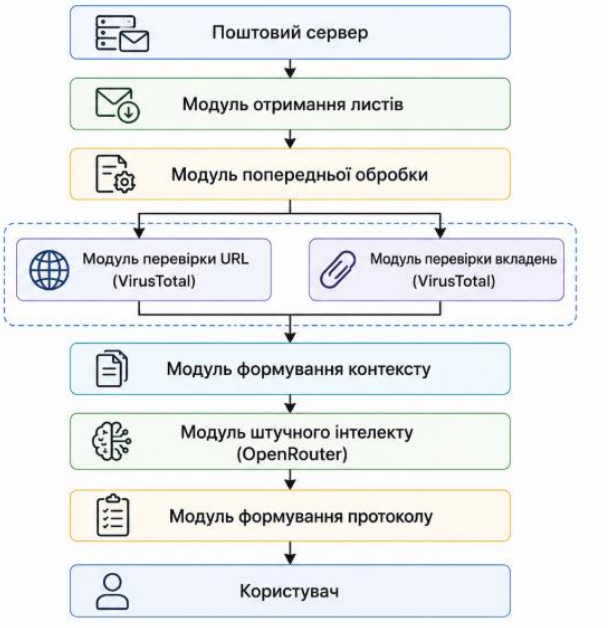


Рисунок 2.6 – Структурна схема розроблюваного програмного засобу

В основу роботи програмного засобу покладено гібридний підхід до виявлення фішингових повідомлень. На відміну від традиційних систем, що використовують лише сигнатурний або репутаційний аналіз, запропоноване рішення поєднує перевірку технічних характеристик повідомлення із семантичним аналізом його змісту.

Після отримання листа система виконує попередню обробку повідомлення та виділяє всі необхідні компоненти для подальшого аналізу. У разі наявності вкладень або гіперпосилань здійснюється їх перевірка через сервіс VirusTotal. Отримані результати використовуються як додатковий контекст під час роботи модуля штучного інтелекту.

До основних вимог, які висуваються до розроблюваного програмного засобу, належать:

– автоматизація процесу аналізу електронних листів;

– забезпечення високої точності виявлення потенційно небезпечних повідомлень;

– можливість перевірки вкладень та вебпосилань;

– підтримка використання сучасних моделей штучного інтелекту;

– формування зрозумілого та деталізованого протоколу перевірки;

– забезпечення зручного графічного інтерфейсу користувача.

Таким чином, сформульовано задачу розробки програмного засобу для виявлення фішингових електронних листів із використанням методів штучного інтелекту. Запропонований підхід передбачає поєднання репутаційного аналізу вкладень та вебпосилань із можливостями великих мовних моделей, що дозволяє підвищити ефективність виявлення сучасних фішингових атак. Отримані результати постановки задачі є основою для проєктування та реалізації програмного засобу, опис яких наведено в наступному розділі.

Висновки до розділу 2

У другому розділі було розглянуто теоретичні основи побудови систем виявлення фішингових електронних листів із використанням методів штучного інтелекту. Проведено аналіз архітектури сучасних систем обробки електронної пошти та визначено основні етапи аналізу повідомлень, які включають отримання листів, попередню обробку вмісту, перевірку вкладень і гіперпосилань, а також формування підсумкового висновку щодо рівня їхньої безпеки.

Досліджено основні ознаки фішингових електронних листів. Встановлено, що ефективність виявлення потенційно небезпечних повідомлень залежить від комплексного аналізу технічних параметрів повідомлення, адреси відправника, вкладень, гіперпосилань, службових заголовків та змісту листа. Визначено, що окремі ознаки не завжди дозволяють однозначно класифікувати повідомлення, тому доцільним є використання багаторівневого підходу до аналізу.

Розглянуто можливості сервісу VirusTotal як засобу репутаційного аналізу вебпосилань та вкладень. Встановлено, що використання даного сервісу дозволяє отримувати результати перевірки від великої кількості антивірусних рушіїв і систем кіберзахисту, що підвищує достовірність оцінювання потенційно небезпечних об'єктів. Водночас визначено, що репутаційний аналіз не дозволяє оцінювати зміст повідомлень та виявляти ознаки соціальної інженерії.

Окрему увагу приділено використанню великих мовних моделей для аналізу електронних листів. Встановлено, що методи штучного інтелекту забезпечують можливість семантичного аналізу тексту, виявлення маніпулятивних прийомів, оцінювання контексту повідомлення та формування обґрунтованих висновків щодо рівня потенційної небезпеки листа. Показано переваги великих мовних моделей порівняно з традиційними підходами до аналізу повідомлень.

На основі проведеного дослідження сформульовано задачу розробки програмного засобу для виявлення фішингових електронних листів із використанням методів штучного інтелекту. Визначено основні функціональні вимоги до системи, розроблено структурну схему програмного засобу та алгоритм його функціонування. Запропоновано використання гібридного підходу, який поєднує репутаційний аналіз вкладень і вебпосилань із семантичним аналізом змісту повідомлень за допомогою великих мовних моделей.

Отримані результати створюють теоретичну основу для розробки програмного засобу, архітектура, реалізація та результати практичного використання якого розглядаються в наступному розділі.

РОЗДІЛ 3. РОЗРОБКА ПРОГРАМНОГО ЗАСОБУ ДЛЯ ВИЯВЛЕННЯ ФІШИНГОВИХ ЕЛЕКТРОННИХ ЛИСТІВ

3.1 Проєктування архітектури програмного засобу

Відповідно до поставленої задачі було спроєктовано програмний засіб для виявлення фішингових електронних листів із використанням методів штучного інтелекту. Основною метою проєктування стало створення системи, яка забезпечує автоматизоване отримання електронних листів із поштової скриньки користувача, аналіз вкладень та гіперпосилань, використання сучасних засобів штучного інтелекту для оцінювання змісту повідомлень та формування детального протоколу перевірки.

Під час проєктування системи було обрано модульний підхід, який дозволяє розділити функціональність програмного засобу на незалежні компоненти. Такий підхід спрощує підтримку програмного забезпечення, забезпечує можливість подальшого розширення функціоналу та підвищує надійність роботи системи.

В основі архітектури програмного засобу лежить клієнт-серверний принцип взаємодії із зовнішніми сервісами. Локальна частина системи відповідає за отримання електронних листів, їх попередню обробку та взаємодію з користувачем. Для аналізу вкладень і гіперпосилань використовується зовнішній сервіс VirusTotal, а для інтелектуального аналізу змісту повідомлень застосовуються великі мовні моделі через платформу OpenRouter. [24, 22]

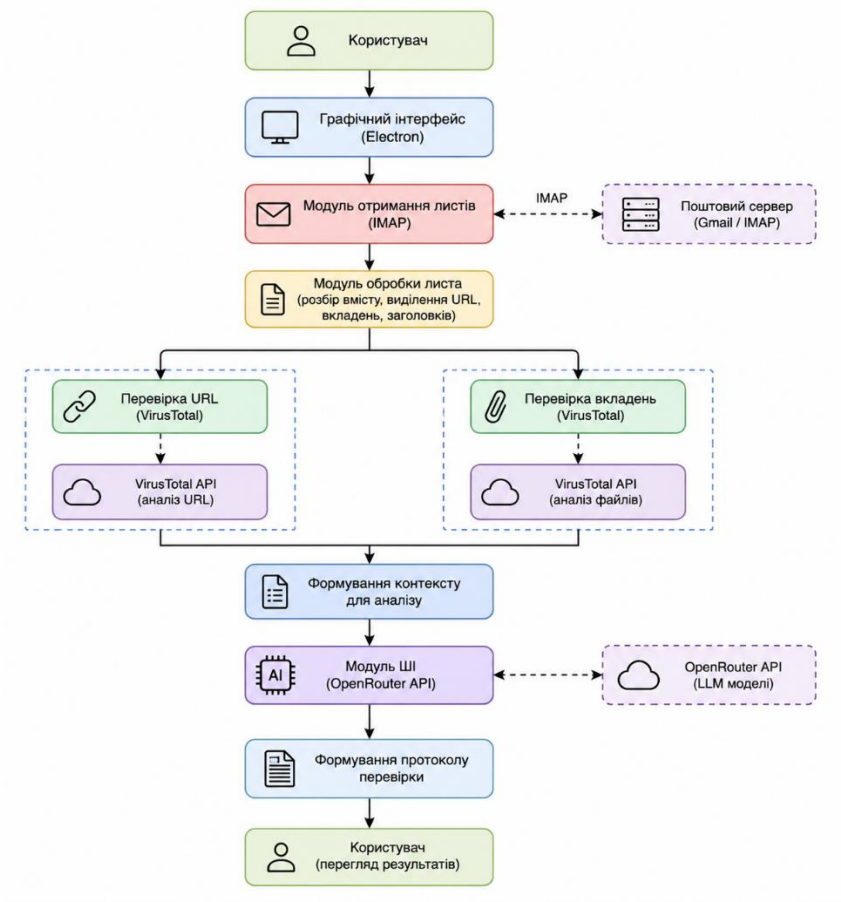


Рисунок 3.1 – Архітектура програмного засобу для виявлення фішингових електронних листів

Першим етапом роботи системи є встановлення з'єднання з поштовим сервером користувача. Для цього використовується протокол IMAP, який забезпечує доступ до поштової скриньки без необхідності завантаження всіх повідомлень на локальний пристрій. Після успішної автентифікації система отримує список доступних листів та забезпечує їх подальшу обробку. [18]

Модуль обробки повідомлень виконує виділення основних компонентів листа, серед яких адреса відправника, тема повідомлення, текстовий вміст, HTML-вміст, вкладення, гіперпосилання та службові заголовки електронної пошти. Отримана інформація використовується на наступних етапах аналізу.

Для перевірки вкладень та вебпосилань використовується окремий модуль взаємодії із сервісом VirusTotal. Даний модуль виконує передачу необхідних даних

до зовнішнього сервісу, отримує результати аналізу та формує структуровану інформацію щодо рівня потенційної небезпеки перевірених об'єктів.

Особливістю розробленого програмного засобу є використання окремого модуля формування контексту аналізу. Його функцією є об'єднання інформації, отриманої з різних джерел, у єдиний структурований набір даних. До такого набору входять текст листа, інформація про відправника, результати перевірки вкладень та URL-адрес, а також інші параметри, що можуть впливати на процес оцінювання рівня ризику.

Подальший аналіз здійснюється модулем штучного інтелекту. Для цього використовується платформа OpenRouter, яка забезпечує доступ до сучасних великих мовних моделей. На основі сформованого контексту модель виконує аналіз змісту повідомлення, виявляє ознаки соціальної інженерії, оцінює потенційні ризики та формує структурований результат перевірки. [22]

```
contextBridge.exposeInMainWorld('api', {
  connect:      (cfg)     => ipcRenderer.invoke('imap:connect', cfg),
  disconnect:   ()        => ipcRenderer.invoke('imap:disconnect'),
  fetchEmails:  (limit)   => ipcRenderer.invoke('imap:fetch', limit),

  aiCheck: (email, apiKey, aiModel, vtResult) =>
    ipcRenderer.invoke('ai:check', { email, apiKey, aiModel, vtResult }),

  vtCheckUrls:  (urls, apiKey) =>
    ipcRenderer.invoke('vt:checkUrls', { urls, apiKey }),

  vtCheckFiles: (attachments, apiKey) =>
    ipcRenderer.invoke('vt:checkFiles', { attachments, apiKey }),
})
```

Рисунок 3.1 – Реалізація взаємодії між основними модулями програмного засобу

Взаємодія між основними компонентами системи реалізована через механізм IPC (Inter-Process Communication) платформи Electron. Після отримання електронного листа через модуль IMAP виконується перевірка URL-адрес та вкладень засобами VirusTotal, після чого сформований контекст передається до модуля штучного інтелекту OpenRouter для виконання аналізу та формування підсумкового протоколу перевірки.

Після завершення аналізу модуль формування протоколу створює підсумковий звіт, який містить рівень ризику, оцінку безпечності повідомлення, перелік виявлених ознак та пояснення отриманого результату. Такий підхід дозволяє користувачу не лише отримати підсумковий висновок, але й зрозуміти причини прийняття відповідного рішення системою.

3.2 Реалізація модуля отримання електронних листів через протокол IMAP

Одним із ключових компонентів розробленого програмного засобу є модуль отримання електронних листів. Його основним призначенням є встановлення з'єднання з поштовим сервером користувача, отримання списку повідомлень та забезпечення подальшої передачі даних до модулів аналізу.

Для реалізації доступу до електронної пошти було обрано протокол IMAP (Internet Message Access Protocol). Даний протокол є одним із найбільш поширених стандартів доступу до поштових скриньок та забезпечує можливість роботи з повідомленнями без необхідності їх повного завантаження на локальний пристрій. На відміну від протоколу POP3, IMAP підтримує синхронізацію стану поштової скриньки між різними пристроями та дозволяє працювати безпосередньо з повідомленнями, що зберігаються на сервері. [18]

У розробленому програмному засобі користувач має можливість самостійно налаштувати параметри підключення до поштового сервера. Для цього реалізовано окреме вікно налаштувань.

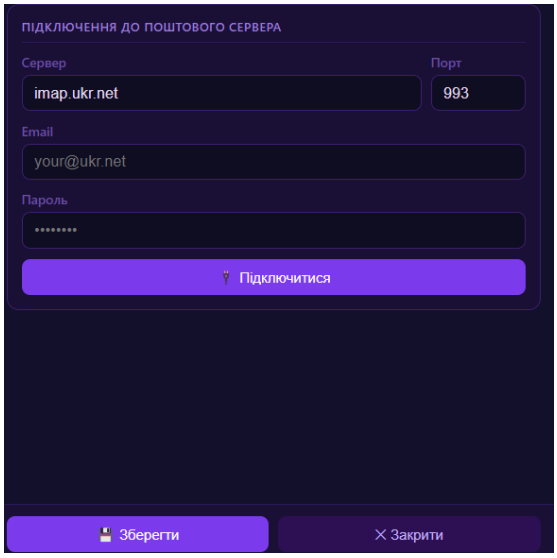


Рисунок 3.2 – Вікно налаштування підключення до поштового сервера

Для встановлення з'єднання необхідно вказати адресу IMAP-сервера, номер порту, адресу електронної пошти та пароль користувача. У випадку використання поштового сервісу Ukr.net застосовується сервер imap.ukr.net із захищеним з'єднанням через порт 993.

Після введення необхідних параметрів виконується процедура автентифікації користувача. У разі успішного встановлення з'єднання система отримує доступ до вмісту поштової скриньки та переходить до етапу завантаження повідомлень.

Особливістю реалізованого рішення є можливість автоматичного оновлення списку повідомлень. Після підключення до поштового сервера система періодично перевіряє наявність нових листів та відображає їх у графічному інтерфейсі користувача. Це дозволяє забезпечити оперативне виявлення потенційно небезпечних повідомлень без необхідності повторного ручного підключення до поштового сервера.

Після отримання листа виконується його первинна обробка. На цьому етапі система отримує основні атрибути повідомлення, серед яких:

– адреса відправника;

– тема повідомлення;

– дата надсилання;

– текстовий вміст листа;

– HTML-вміст листа;

– вкладення;

– службові заголовки електронної пошти.

Отримана інформація використовується для подальшого аналізу та формування структурованого представлення повідомлення.

Важливим етапом роботи модуля є виділення вкладень та гіперпосилань. Для цього здійснюється аналіз MIME-структури повідомлення, що дозволяє визначити наявність файлів, їх типи та інші характеристики. Одночасно із текстового та HTML-вмісту повідомлення виділяються всі URL-адреси для подальшої перевірки через сервіс VirusTotal.

```
$('btn-connect').addEventListener('click', async () => {
  const host     = ($('inp-host').value  || '').trim()
  const port     = parseInt($('inp-port').value || '993', 10)
  const email    = ($('inp-email').value || '').trim()
  const password = $('inp-password').value || ''
  const errBox   = $('conn-error')

  if (!host || !email || !password) {
    errBox.textContent = 'Заповніть усі поля'; errBox.style.display = ''; return
  }
  errBox.style.display = 'none'
  $('btn-connect').disabled = true
  $('btn-connect').textContent = '⏳ Підключення…'
  setStatus('Підключення…')

  const res = await window.settingsApi.imapConnect({ host, port, email, password })

  if (res.ok) {
    await window.settingsApi.saveConnection({ host, port, email, password })
    $('btn-connect').style.display    = 'none'
    $('btn-disconnect').style.display = ''
    setStatus('✅ Підключено як ' + email, '■#86efac')
  } else {
    errBox.textContent = 'Помилка: ' + (res.error || '')
    errBox.style.display = ''
    $('btn-connect').disabled = false
    $('btn-connect').textContent = '🔌 Підключитися'
    setStatus('Помилка підключення', '■#fca5a5')
  }
})
$('btn-disconnect').addEventListener('click', async () => {
  await window.settingsApi.imapDisconnect()
  await window.settingsApi.saveConnection(null)
  $('btn-disconnect').style.display = 'none'
  $('btn-connect').style.display    = ''
  $('btn-connect').disabled = false
  $('btn-connect').textContent = '🔌 Підключитися'
  setStatus('Відключено')
})
```

Рисунок 3.3 – Фрагмент програмного коду підключення до поштового сервера через протокол IMAP

Для підвищення зручності використання програмного засобу реалізовано механізм збереження параметрів підключення. Це дозволяє уникнути необхідності повторного введення налаштувань під час кожного запуску програми та прискорює процес роботи користувача із системою.

Таким чином, реалізований модуль отримання електронних листів забезпечує надійне підключення до поштового сервера через протокол IMAP, автоматизоване отримання повідомлень та підготовку необхідних даних для подальшого аналізу. Отримана інформація використовується наступними модулями системи для перевірки вкладень, гіперпосилань та змісту електронних листів.

3.3 Реалізація модуля перевірки вкладень та посилань із використанням VirusTotal API

Одним із ключових етапів аналізу електронних листів є перевірка вкладень та гіперпосилань на наявність ознак шкідливої активності. Для реалізації даного функціоналу у програмному засобі використано сервіс VirusTotal, який надає можливість автоматизованого аналізу файлів та вебресурсів через програмний інтерфейс API. [24]

Використання VirusTotal дозволяє отримувати результати перевірки від великої кількості антивірусних рушіїв та систем кіберзахисту. Це забезпечує підвищення достовірності оцінювання безпечності об'єктів та зменшує ймовірність пропуску відомих загроз.

```
const VT_BASE = 'https://www.virustotal.com/api/v3'

function vtRequest(method, path, apiKey, body, contentType) {
  return new Promise((resolve, reject) => {
    const data = body ? (typeof body === 'string' ? body : JSON.stringify(body)) : null
    const req = https.request({
      hostname: 'www.virustotal.com',
      path:     '/api/v3' + path,
      method,
      headers: {
        'x-apikey':     apiKey,
        'Accept':       'application/json',
        ...(data ? { 'Content-Type': contentType || 'application/json', 'Content-Length': Buffer.byteLength(data) } : {}),
      },
    }, (res) => {
      let raw = ''
      res.on('data', c => (raw += c))
      res.on('end', () => {
        try { resolve({ status: res.statusCode, data: JSON.parse(raw) }) }
        catch { resolve({ status: res.statusCode, data: { error: raw } }) }
      })
    })
    req.on('error', reject)
    if (data) req.write(data)
    req.end()
  })
}
```

Рисунок 3.4 – Реалізація функції взаємодії з VirusTotal API

Після отримання електронного листа система виконує пошук усіх URL-адрес, що містяться у текстовому або HTML-вмісті повідомлення. Для кожного знайденого посилання формується окремий запит до VirusTotal API.

Під час перевірки URL-адреси сервіс аналізує репутацію вебресурсу та повертає статистику виявлених загроз. Зокрема, система отримує інформацію про кількість антивірусних рушіїв, які класифікували ресурс як безпечний, підозрілий або шкідливий.

```
async function vtCheckUrl(url, apiKey) {
  try {
    const urlId = Buffer.from(url).toString('base64').replace(/=/g, '')
    const { status, data } = await vtRequest('GET', '/urls/' + urlId, apiKey)

    if (status === 200 && data.data?.attributes?.last_analysis_stats) {
      const stats = data.data.attributes.last_analysis_stats
      const meta  = {
        lastAnalysis: data.data.attributes.last_analysis_date,
        reputation:   data.data.attributes.reputation,
        categories:   data.data.attributes.categories,
      }
      return { url, ...vtFormatResult(stats, meta) }
    }

    const formBody = 'url=' + encodeURIComponent(url)
    const scanRes  = await vtRequest('POST', '/urls', apiKey, formBody, 'application/x-www-form-urlencoded')

    if (scanRes.status !== 200) {
      return { url, verdict: 'error', error: 'Не вдалося відправити на сканування' }
    }

    const analysisId = scanRes.data.data?.id
    if (!analysisId) return { url, verdict: 'error', error: 'Немає ID аналізу' }

    const attrs = await vtWaitAnalysis(analysisId, apiKey, 30000)
    if (!attrs) return { url, verdict: 'pending', error: 'Аналіз ще не завершено' }

    return { url, ...vtFormatResult(attrs.stats) }
  } catch (err) {
    return { url, verdict: 'error', error: err.message }
  }
}
```

Рисунок 3.5 – Фрагмент програмного коду перевірки URL-адрес через VirusTotal API

У випадку наявності вкладень система виконує їх окрему перевірку. Для цього використовується механізм аналізу файлів за криптографічним хешем SHA-256. Такий підхід дозволяє швидко визначити, чи аналізувався файл раніше у базі VirusTotal.

Якщо інформація про файл уже присутня у сервісі, система отримує готовий результат перевірки без необхідності повторного завантаження файлу. За відсутності відповідного запису виконується передача вкладення до VirusTotal для проведення нового сканування.

Після завершення аналізу система отримує статистику детектування, яка містить інформацію про кількість антивірусних рушіїв, що виявили ознаки шкідливої активності у файлі.

```
async function vtCheckFile(filename, fileData, apiKey) {
  try {
    const hash = crypto.createHash('sha256').update(fileData).digest('hex')

    const { status, data } = await vtRequest('GET', '/files/' + hash, apiKey)

    if (status === 200 && data.data?.attributes?.last_analysis_stats) {
      const stats = data.data.attributes.last_analysis_stats
      const meta  = {
        name:         data.data.attributes.meaningful_name || filename,
        type:         data.data.attributes.type_description,
        size:         data.data.attributes.size,
        sha256:       hash,
        lastAnalysis: data.data.attributes.last_analysis_date,
      }
      return { filename, hash, ...vtFormatResult(stats, meta) }
    }
    const boundary = '----VTBoundary' + Date.now()
    const parts = [
      '--' + boundary + '\r\n',
      'Content-Disposition: form-data; name="file"; filename="' + filename + '"\r\n',
      'Content-Type: application/octet-stream\r\n\r\n',
    ]
    const header = Buffer.from(parts.join(''))
    const footer = Buffer.from('\r\n--' + boundary + '--\r\n')
    const body   = Buffer.concat([header, fileData, footer])

    const uploadRes = await vtRequest('POST', '/files', apiKey, body.toString('binary'), 'multipart/form-data; boundary=' + boundary)

    if (uploadRes.status !== 200) {
      return { filename, hash, verdict: 'error', error: 'Не вдалося завантажити файл' }
    }
    const analysisId = uploadRes.data.data?.id
    if (!analysisId) return { filename, hash, verdict: 'error', error: 'Немає ID аналізу' }

    const attrs = await vtWaitAnalysis(analysisId, apiKey, 60000)
    if (!attrs) return { filename, hash, verdict: 'pending', error: 'Аналіз ще не завершено' }

    return { filename, hash, ...vtFormatResult(attrs.stats) }
  } catch (err) {
    return { filename, verdict: 'error', error: err.message }
  }
}
```

Рисунок 3.6 –Фрагмент програмного коду перевірки вкладень через VirusTotal API

Результати перевірки, отримані від VirusTotal, використовуються як додаткове джерело інформації під час аналізу електронного листа. Усі дані збираються у структурований набір параметрів, який надалі передається до модуля штучного інтелекту для комплексного оцінювання рівня ризику повідомлення.

Поле	Значення	Звідки
`verdict`	`malicious` / `suspicious` / `possibly_suspicious` / `clean` (підсумковий вердикт)	Обчислюється з `malicious` і `suspicious`
`malicious`	Кількість антивірусних рушіїв, що визначили об'єкт як шкідливий	`stats.malicious`
`suspicious`	Кількість антивірусних рушіїв, що визначили об'єкт як підозрілий	`stats.suspicious`
`total`	Загальна кількість виконаних перевірок (сума всіх значень зі `stats`)	Сума всіх stats: `malicious` + `suspicious` + `harmless` + `undetected`
`hash`	SHA256 хеш файлу (вкладення)	Рахується локально або отримується з VT

Таблиця 3.1 – Дані, що отримуються від VirusTotal API

Особливістю реалізованого модуля є автоматична обробка отриманих результатів та їх інтеграція із загальною системою аналізу повідомлень. Завдяки цьому користувач не взаємодіє безпосередньо із сервісом VirusTotal, а отримує вже підготовлену інформацію у складі підсумкового протоколу перевірки.

Таким чином, реалізований модуль перевірки вкладень та посилань забезпечує автоматизований репутаційний аналіз об'єктів, що містяться в електронному листі. Використання VirusTotal дозволяє своєчасно виявляти відомі загрози та формує додатковий інформаційний контекст для подальшого інтелектуального аналізу повідомлення за допомогою методів штучного інтелекту.

3.4 Реалізація модуля інтелектуального аналізу електронних листів на основі OpenRouter

Центральним компонентом розробленого програмного засобу є модуль інтелектуального аналізу електронних листів, реалізований із використанням

сучасних великих мовних моделей. Основним призначенням даного модуля є комплексний аналіз змісту повідомлення, результатів перевірки вкладень та гіперпосилань, а також формування обґрунтованого висновку щодо рівня потенційної небезпеки листа.

Для забезпечення доступу до великих мовних моделей у програмному засобі використовується платформа OpenRouter. Дана платформа надає уніфікований програмний інтерфейс для взаємодії з різними моделями штучного інтелекту та дозволяє використовувати їх можливості без необхідності прямої інтеграції з окремими сервісами. [22]

```
const AI_URL           = 'https://openrouter.ai/api/v1/chat/completions'
const AI_MODEL_DEFAULT = 'openai/gpt-oss-120b:free'

function httpsPost(url, headers, body) {
  return new Promise((resolve, reject) => {
    const parsed = new URL(url)
    const data   = JSON.stringify(body)
    const req = https.request({
      hostname: parsed.hostname,
      path:     parsed.pathname,
      method:   'POST',
      headers:  { ...headers, 'Content-Length': Buffer.byteLength(data) },
    }, (res) => {
      let raw = ''
      res.on('data', c => (raw += c))
      res.on('end', () => {
        try { resolve({ status: res.statusCode, data: JSON.parse(raw) }) }
        catch { resolve({ status: res.statusCode, data: { error: raw } }) }
      })
    })
    req.on('error', reject)
    req.write(data)
    req.end()
  })
}
ipcMain.handle('ai:check', async (_, { email, apiKey, aiModel, vtResult }) => {
  if (!apiKey) return { ok: false, error: 'API ключ не вказано' }
  const MODEL = (aiModel || AI_MODEL_DEFAULT).trim()
  const toStr = v => {
    if (!v) return ''
    if (typeof v === 'string') return v
    if (typeof v === 'object') return v.text || v.value || ''
    return String(v)
  }
```

Рисунок 3.7 – Фрагмент програмного коду реалізації взаємодії штучного інтелекту

На відміну від класичних систем аналізу електронної пошти, які використовують набір заздалегідь визначених правил, запропонований програмний засіб застосовує контекстний аналіз повідомлень. Перед передачею інформації до мовної моделі формується структурований набір даних, що містить усі характеристики повідомлення, необхідні для прийняття рішення. [23]

До складу інформації, яка передається до моделі штучного інтелекту, входять адреса відправника, тема повідомлення, дата надсилання, текст листа, службові заголовки електронної пошти, список вкладень, список URL-адрес та результати перевірки через VirusTotal.

```
// Витягуємо всі URL з листа для окремого аналізу
const bodyText = toStr(email.body || email.html)
const urlMatches = bodyText.match(/https?:\/\/[^\s"'<>)]+/gi) || []
const urlList = urlMatches.length
  ? 'Знайдені посилання:\n' + urlMatches.slice(0, 10).map(u => '  - ' + u).join('\n')
  : 'Посилань не знайдено'

// Вкладення
const attachList = (email.attachments || []).length
  ? 'Вкладення: ' + email.attachments.map(a => toStr(a.filename || a.name || 'невідомо')).join(', ')
  : 'Вкладень немає'

// Формуємо блок з результатами VirusTotal якщо є
let vtBlock = ''
if (vtResult) {
  const lines = ['=== РЕЗУЛЬТАТИ VIRUSTOTAL ===']
  if (vtResult.urls?.length) {
    lines.push('Перевірені посилання:')
    vtResult.urls.forEach(u => {
      lines.push('  URL: ' + u.url)
      lines.push('  Вердикт: ' + u.verdict + ' (' + (u.malicious || 0) + '/' + (u.total || 0) + ' детекцій)')
    })
  }
  if (vtResult.files?.length) {
    lines.push('Перевірені файли:')
    vtResult.files.forEach(f => {
      lines.push('  Файл: ' + f.filename)
      lines.push('  Вердикт: ' + f.verdict + ' (' + (f.malicious || 0) + '/' + (f.total || 0) + ' детекцій)')
      if (f.hash) lines.push('  SHA256: ' + f.hash)
    })
  }
  vtBlock = lines.join('\n')
}
```

```
const systemPrompt = SYSTEM_PROMPT

const emailData = [
  'ВІД: '          + toStr(email.from),
  'КОМУ: '         + toStr(email.to),
  'ТЕМА: '         + toStr(email.subject),
  'REPLY-TO: '     + toStr(email.headers?.replyTo),
  'RETURN-PATH: '  + toStr(email.headers?.returnPath),
  'ДАТА: '         + toStr(email.date),
  '',
  urlList,
  attachList,
  '',
  vtBlock ? vtBlock + '\n' : '',
  'ТІЛО ЛИСТА:',
  bodyText.slice(0, 2000),
].join('\n')

try {
  const { status, data } = await httpsPost(AI_URL, {
    'Content-Type':  'application/json',
    'Authorization': 'Bearer ' + apiKey,
    'HTTP-Referer':  'http://localhost',
    'X-Title':       'Phishing Detector',
  }, {
    model: MODEL,
    messages: [
      { role: 'system', content: systemPrompt },
      { role: 'user',   content: emailData },
    ],
    temperature: 0.1,
    max_tokens:  700,
  })
```

Рисунок 3.8 – Формування контексту аналізу для мовної моделі

Після збору всієї необхідної інформації формується спеціалізований запит до мовної моделі. Запит містить інструкції щодо виконання аналізу повідомлення та критерії оцінювання рівня ризику. Такий підхід дозволяє забезпечити стандартизований формат отримуваних результатів та підвищує стабільність роботи системи.

У процесі аналізу модель оцінює зміст повідомлення, визначає наявність ознак соціальної інженерії, аналізує стиль викладення тексту, перевіряє логічну узгодженість повідомлення та враховує результати репутаційного аналізу вкладень і посилань.

Особливістю реалізації є використання структурованої відповіді у форматі JSON. Такий формат спрощує автоматичну обробку результатів та їх подальше відображення користувачеві.

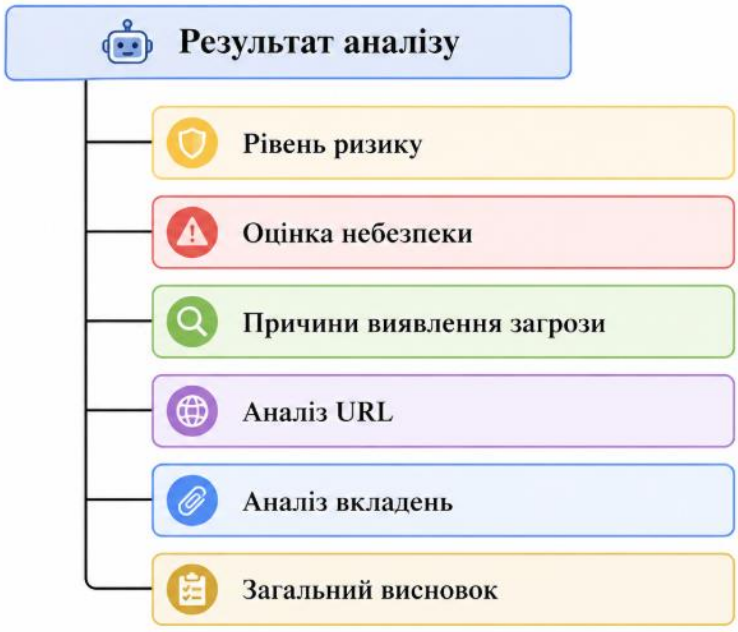


Рисунок 3.9 – Структура відповіді мовної моделі

На основі отриманої відповіді система формує підсумковий протокол перевірки. У протоколі відображається категорія повідомлення, рівень ризику, перелік виявлених підозрілих ознак та текстове пояснення причин прийнятого рішення.

Однією з переваг використання великих мовних моделей є можливість пояснювати результати аналізу. На відміну від багатьох традиційних систем захисту електронної пошти, які лише повідомляють про наявність загрози, розроблений програмний засіб надає користувачеві детальний опис виявлених ризиків та факторів, які вплинули на підсумковий висновок.

Додатковою перевагою реалізованого підходу є можливість аналізу нових типів фішингових повідомлень. Оскільки великі мовні моделі працюють із контекстом повідомлення, вони здатні виявляти ознаки атак навіть у тих випадках, коли конкретні шаблони або сигнатури відсутні у базах даних систем кіберзахисту.

Таким чином, реалізований модуль інтелектуального аналізу забезпечує комплексну оцінку електронних листів із використанням сучасних методів штучного інтелекту. Поєднання результатів репутаційного аналізу та можливостей великих мовних моделей дозволяє підвищити точність виявлення фішингових повідомлень та забезпечує користувача детальним поясненням отриманих результатів.

3.5 Реалізація користувацького інтерфейсу програмного засобу

Для забезпечення зручної взаємодії користувача із системою було реалізовано графічний інтерфейс користувача. Під час розробки інтерфейсу основна увага приділялася простоті використання, наочності відображення результатів аналізу та забезпеченню швидкого доступу до основних функцій програмного засобу.

Для створення інтерфейсу використано платформу Electron, яка дозволяє розробляти кросплатформні настільні застосунки з використанням сучасних вебтехнологій. Застосування Electron забезпечує можливість використання програмного засобу на різних операційних системах без необхідності суттєвих змін програмного коду. [21]

Головне вікно програмного засобу наведено на рисунку 3.10.

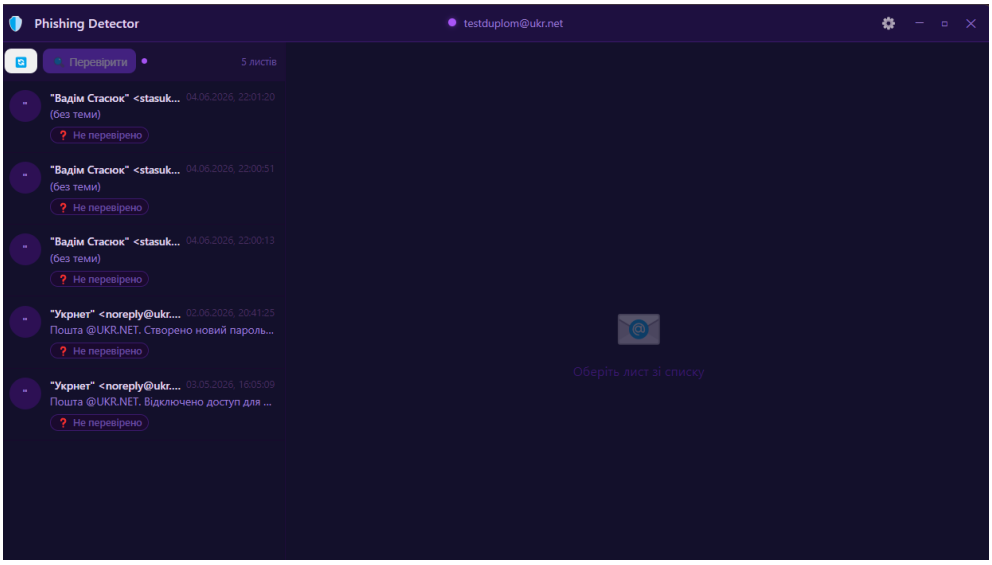


Рисунок 3.10 – Головне вікно програмного засобу

Основним елементом інтерфейсу є список отриманих електронних листів. Користувач має можливість переглядати інформацію про відправника, тему повідомлення та інші характеристики листів. Після вибору повідомлення система виконує його аналіз та відображає результати перевірки у відповідній області інтерфейсу.

Для забезпечення доступу до поштової скриньки реалізовано окреме вікно налаштувань IMAP-підключення, наведене на рисунку 3.11.

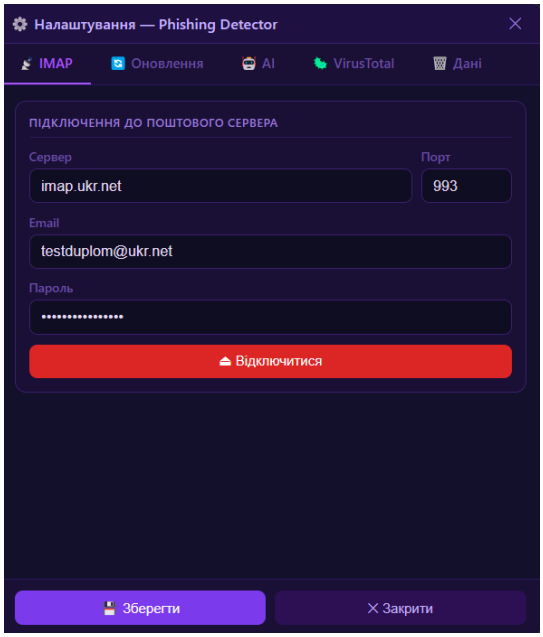


Рисунок 3.11 – Вікно налаштування IMAP-підключення

У вікні налаштувань користувач може вказати адресу IMAP-сервера, номер порту, адресу електронної пошти та пароль. Після збереження параметрів система використовує їх для автоматичного встановлення з'єднання з поштовим сервером.

Для взаємодії з великими мовними моделями реалізовано окреме вікно налаштувань OpenRouter API. Відповідний інтерфейс наведено на рисунку 3.12.

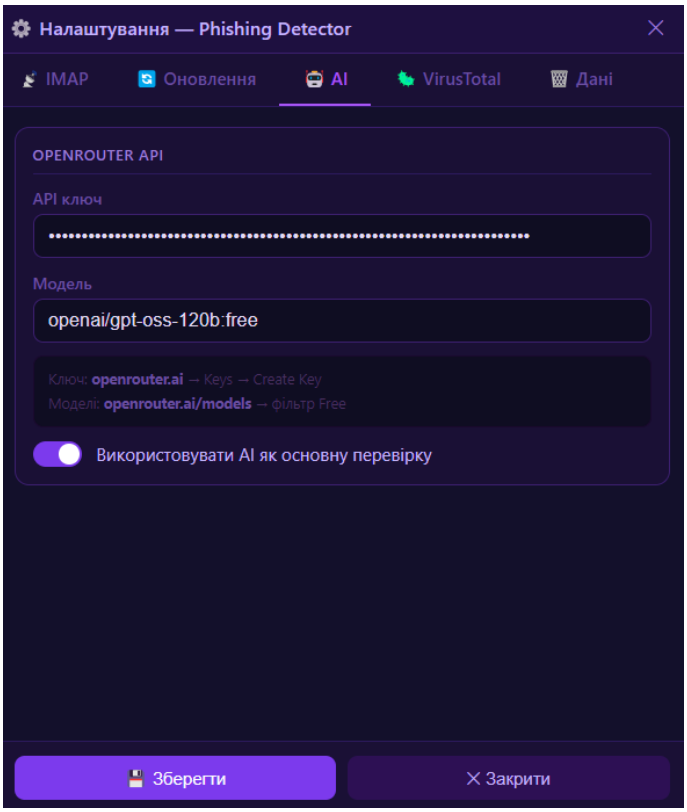


Рисунок 3.12 – Вікно налаштування OpenRouter API

Дане вікно дозволяє користувачу вказати API-ключ та обрати модель штучного інтелекту, яка буде використовуватися для аналізу електронних листів. Такий підхід забезпечує гнучкість налаштування системи та можливість використання різних моделей без необхідності внесення змін до програмного коду.

Для налаштування перевірки вкладень та URL-адрес використовується окреме вікно конфігурації VirusTotal API, наведене на рисунку 3.13.

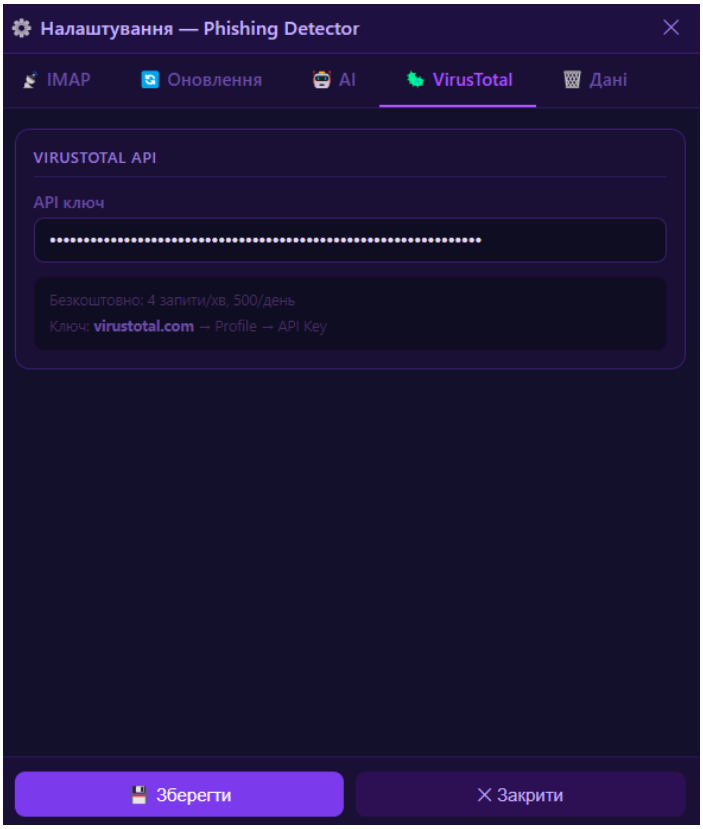


Рисунок 3.13 – Вікно налаштування VirusTotal API

У даному вікні користувач може ввести персональний API-ключ сервісу VirusTotal, після чого система отримує можливість виконувати автоматичний аналіз файлів та вебресурсів.

У результаті перевірки користувач отримує детальну інформацію щодо рівня ризику повідомлення. Протокол містить оцінку небезпеки, опис виявлених підозрілих ознак, результати перевірки вкладень та URL-адрес, а також загальний висновок щодо безпечності повідомлення.

Розроблений інтерфейс забезпечує логічну структуру розташування елементів керування та мінімізує кількість дій, необхідних для проведення аналізу повідомлень. Завдяки цьому програмний засіб може використовуватися як користувачами без спеціальної підготовки у сфері інформаційної безпеки, так і фахівцями з кібербезпеки.

Таким чином, реалізований користувацький інтерфейс забезпечує зручну взаємодію із системою, налаштування параметрів роботи, перегляд отриманих повідомлень та відображення результатів аналізу. Використання платформи Electron

дозволило створити сучасний та функціональний настільний застосунок для виявлення фішингових електронних листів.

3.6 Практичне тестування та оцінювання ефективності розробленого програмного засобу

Після завершення розробки програмного засобу було проведено його практичне тестування з метою перевірки працездатності реалізованих функцій та оцінювання ефективності виявлення фішингових електронних листів. Основними завданнями тестування були перевірка коректності роботи модулів отримання повідомлень, аналізу вкладень та URL-адрес, взаємодії із сервісом VirusTotal, а також оцінювання результатів роботи модуля штучного інтелекту.

Тестування проводилося на персональному комп'ютері із підключенням до реальної поштової скриньки через протокол IMAP. Для перевірки функціональності системи використовувалися електронні листи різних категорій, серед яких безпечні повідомлення, інформаційні розсилки, повідомлення з вкладеннями та листи, що містять ознаки фішингових атак.

Під час тестування перевірялися такі функціональні можливості програмного засобу:

– підключення до поштового сервера;

– отримання списку повідомлень;

– аналіз текстового вмісту листів;

– виділення URL-адрес;

– виділення вкладень;

– перевірка вкладень через VirusTotal;

– перевірка вебпосилань через VirusTotal;

– формування запиту до OpenRouter;

– отримання результатів аналізу;

– формування протоколу перевірки.

На першому етапі було перевірено коректність роботи модуля отримання повідомлень. У результаті тестування встановлено, що система успішно виконує підключення до поштового сервера, отримує список повідомлень та відображає їх у графічному інтерфейсі користувача.

На другому етапі проводилася перевірка роботи механізму аналізу URL-адрес та вкладень. Для цього використовувалися повідомлення, що містили різні типи файлів та вебпосилань. Усі виявлені об'єкти були успішно передані до сервісу VirusTotal та проаналізовані за допомогою зовнішніх механізмів кіберзахисту.

Наступним етапом стала перевірка роботи модуля штучного інтелекту. Під час тестування система формувала структурований запит до OpenRouter, який містив текст повідомлення, інформацію про відправника, результати перевірки вкладень та URL-адрес. На основі отриманих даних мовна модель виконувала комплексний аналіз повідомлення та формувала підсумковий висновок щодо рівня його небезпеки.

Приклад результату аналізу безпечного повідомлення наведено на рисунку 3.14.

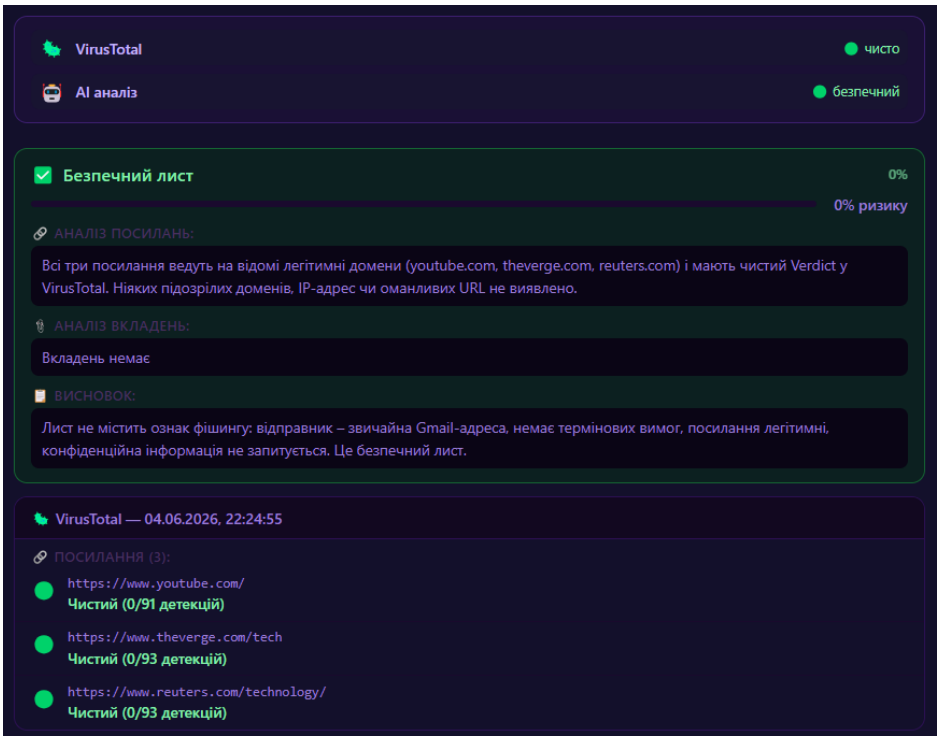


Рисунок 3.14 – Результат аналізу безпечного листа

У результаті аналізу система класифікувала повідомлення як безпечне та не виявила ознак фішингової активності. Всі перевірені вкладення та вебпосилання були визначені як безпечні.

Рисунок 3.15 – Результат аналізу фішингового листа

У даному випадку система виявила низку підозрілих ознак, серед яких використання маніпулятивних формулювань, підозрілі вебпосилання та потенційно небезпечні вкладення. За результатами аналізу повідомлення було класифіковано як небезпечне.

Приклад аналізу повідомлення з вкладенням наведено на рисунку 3.16.

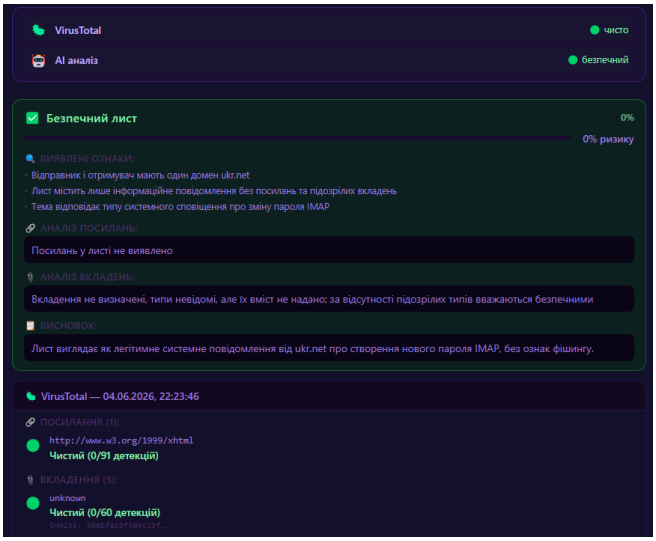


Рисунок 3.16 – Результат аналізу листа з вкладенням

У результаті перевірки вкладення було отримано додаткову інформацію від сервісу VirusTotal, яка була використана під час формування остаточного висновку мовною моделлю.

№	Тип повідомлення	Результат аналізу	Статус
1	Безпечний лист	Безпечний	Успішно
2	Лист із URL	Перевірено	Успішно
3	Лист із вкладенням	Перевірено	Успішно
4	Підозрілий лист	Виявлено ризики	Успішно
5	Фішинговий лист	Виявлено загрозу	Успішно

Таблиця 3.2 – Результати тестування програмного засобу

Для оцінювання ефективності роботи програмного засобу також було виконано порівняння результатів аналізу різних типів повідомлень.

Проведене тестування підтвердило працездатність усіх основних модулів програмного засобу та коректність їх взаємодії. Система успішно виконувала отримання повідомлень, аналіз вкладень і вебпосилань, взаємодію із зовнішніми сервісами та формування підсумкового протоколу перевірки.

Отримані результати свідчать про доцільність використання гібридного підходу, що поєднує репутаційний аналіз через VirusTotal та інтелектуальний аналіз змісту повідомлень за допомогою великих мовних моделей. Такий підхід дозволяє підвищити ефективність виявлення фішингових електронних листів та забезпечує користувача детальною інформацією щодо виявлених ризиків.

Висновки до розділу 3

У третьому розділі було розроблено програмний засіб для виявлення фішингових електронних листів із використанням методів штучного інтелекту. На основі сформульованих у попередніх розділах вимог було спроєктовано архітектуру системи, яка поєднує модулі отримання електронних листів, аналізу вкладень і вебпосилань, інтелектуального аналізу повідомлень та формування підсумкового протоколу перевірки.

Реалізовано модуль отримання електронних листів через протокол IMAP, який забезпечує підключення до поштового сервера користувача, завантаження повідомлень та їх подальшу обробку. Використання протоколу IMAP дозволило забезпечити зручну роботу із поштовою скринькою без необхідності локального зберігання всіх повідомлень.

Розроблено модуль перевірки вкладень та вебпосилань із використанням сервісу VirusTotal. Реалізований механізм дозволяє автоматично аналізувати URL-адреси та вкладення, отримувати результати перевірки від великої кількості антивірусних рушіїв та використовувати їх під час подальшого оцінювання рівня ризику повідомлення.

Особливу увагу приділено реалізації модуля інтелектуального аналізу електронних листів на основі платформи OpenRouter. Використання великих мовних моделей дозволило забезпечити аналіз змісту повідомлень, виявлення ознак соціальної інженерії, оцінювання потенційних ризиків та формування обґрунтованих висновків щодо безпечності листів.

Розроблено графічний інтерфейс користувача, який забезпечує налаштування параметрів підключення, перегляд електронних листів та відображення результатів аналізу у зручному для користувача вигляді. Інтерфейс дозволяє отримувати не лише підсумковий вердикт щодо рівня небезпеки повідомлення, але й детальну інформацію про причини прийнятого рішення.

Проведене практичне тестування підтвердило працездатність усіх реалізованих модулів та коректність їх взаємодії. У процесі тестування програмний засіб успішно виконував отримання повідомлень, перевірку вкладень і URL-адрес,

взаємодію із сервісами VirusTotal та OpenRouter, а також формування підсумкового протоколу аналізу.

Отримані результати свідчать про ефективність використання гібридного підходу, який поєднує репутаційний аналіз та методи штучного інтелекту. Реалізований програмний засіб дозволяє автоматизувати процес виявлення фішингових електронних листів та забезпечує користувача детальною інформацією щодо потенційних загроз.

ВИСНОВКИ

У кваліфікаційній роботі вирішено актуальну задачу розробки програмного засобу для виявлення фішингових електронних листів із використанням методів штучного інтелекту. Актуальність роботи обумовлена постійним зростанням кількості фішингових атак, спрямованих на отримання конфіденційної інформації користувачів, розповсюдження шкідливого програмного забезпечення та компрометацію інформаційних систем.

У процесі виконання роботи було проведено аналіз сучасних фішингових атак та основних механізмів їх реалізації. Розглянуто особливості масового та цільового фішингу, атак типу Business Email Compromise, атак із використанням шкідливих вкладень та підроблених вебресурсів. Встановлено, що сучасні фішингові кампанії характеризуються високим рівнем персоналізації та активно використовують методи соціальної інженерії, що суттєво ускладнює їх виявлення традиційними засобами захисту.

Проведено аналіз сучасних методів виявлення фішингових повідомлень, серед яких сигнатурні, евристичні, репутаційні методи, методи машинного навчання та технології штучного інтелекту. Встановлено, що найбільш перспективним напрямом розвитку систем захисту електронної пошти є використання комплексного підходу, який поєднує декілька механізмів аналізу для підвищення точності виявлення загроз.

У роботі досліджено принципи побудови систем аналізу електронної пошти та визначено основні ознаки фішингових повідомлень. Встановлено, що для ефективного виявлення фішингу необхідно враховувати не лише технічні характеристики повідомлення, але й зміст листа, ознаки соціальної інженерії, результати перевірки вкладень та вебпосилань, а також репутаційні характеристики зовнішніх ресурсів.

Окрему увагу приділено дослідженню можливостей сервісу VirusTotal та великих мовних моделей. Визначено, що використання VirusTotal дозволяє отримувати достовірну інформацію щодо безпечності вкладень і вебресурсів, а

застосування великих мовних моделей забезпечує можливість аналізу змісту повідомлень на семантичному рівні та виявлення прихованих ознак фішингових атак.

На основі проведеного аналізу було спроєктовано архітектуру програмного засобу для виявлення фішингових електронних листів. Розроблена архітектура включає модуль отримання електронних листів через протокол IMAP, модуль перевірки вкладень і вебпосилань через сервіс VirusTotal, модуль інтелектуального аналізу на основі платформи OpenRouter та модуль формування протоколу перевірки.

У процесі практичної реалізації створено програмний засіб, який забезпечує автоматизоване отримання електронних листів із поштової скриньки користувача, виділення вкладень та URL-адрес, перевірку їхньої репутації, аналіз змісту повідомлень за допомогою методів штучного інтелекту та формування детального звіту щодо рівня потенційної небезпеки повідомлення.

Особливістю розробленого програмного засобу є використання гібридного підходу до аналізу електронних листів. На відміну від традиційних систем захисту, запропоноване рішення поєднує результати репутаційного аналізу із можливостями сучасних великих мовних моделей. Це дозволяє підвищити ефективність виявлення як відомих, так і нових фішингових атак.

Проведене тестування підтвердило працездатність усіх реалізованих модулів програмного засобу та коректність їх взаємодії. Розроблена система успішно виконувала аналіз електронних листів різних категорій, здійснювала перевірку вкладень і вебпосилань, взаємодіяла із зовнішніми сервісами та формувала обґрунтовані висновки щодо рівня потенційної небезпеки повідомлень.

Практичне значення отриманих результатів полягає у створенні програмного засобу, який може використовуватися як додатковий інструмент підвищення рівня кібербезпеки користувачів електронної пошти. Розроблене рішення дозволяє автоматизувати процес аналізу повідомлень, зменшити ризик успішної реалізації фішингових атак та підвищити обізнаність користувачів щодо потенційних кіберзагроз.

Таким чином, поставлену мету роботи досягнуто, а всі визначені завдання виконано в повному обсязі.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Корченко О. Г. Основи захисту інформації. Київ : Видавнича група BHV, 2020. 448 с.

2. Корченко О. Г., Гнатюк С. О. Системи виявлення атак у комп'ютерних мережах. Київ : НАУ, 2019. 320 с.

3. Дудикевич В. Б. Захист інформації в комп'ютерних системах і мережах. Львів : Львівська політехніка, 2021. 360 с.

4. Гнатюк С. О. Кібербезпека та захист критичної інформаційної інфраструктури. Київ : НАУ, 2021. 304 с.

5. Бурячок В. Л., Толубко В. Б. Основи формування державної системи кібернетичної безпеки. Київ : ДУТ, 2020. 432 с.

6. Закон України «Про основні засади забезпечення кібербезпеки України» від 05.10.2017 №2163-VIII.

7. Закон України «Про захист інформації в інформаційно-комунікаційних системах» від 31.05.2005 №2594-IV.

8. ДСТУ ISO/IEC 27001:2023. Інформаційна безпека, кібербезпека та захист конфіденційності. Системи управління інформаційною безпекою. Вимоги.

9. Stallings W. Network Security Essentials: Applications and Standards. 7th ed. Pearson, 2020. 784 p.

10. Stallings W. Cryptography and Network Security: Principles and Practice. 8th ed. Pearson, 2023. 880 p.

11. Easttom C. Computer Security Fundamentals. 5th ed. Pearson, 2022. 624 p.

12. Hadnagy C. Social Engineering: The Science of Human Hacking. 2nd ed. Wiley, 2018. 320 p.

13. Jakobsson M., Myers S. Phishing and Countermeasures: Understanding the Increasing Problem of Electronic Identity Theft. Wiley, 2017. 592 p.

14. Gupta B. B., Arachchilage N., Psannis K. Defending Against Phishing Attacks. Springer, 2021. 357 p.

15. Sahoo D., Liu C., Hoi S. H. Malicious URL Detection using Machine Learning. ACM Computing Surveys. 2019. Vol. 52. No. 1.

16. Fette I., Sadeh N., Tomasic A. Learning to Detect Phishing Emails. Proceedings of the International World Wide Web Conference. 2018.

17. Bergholz A., Chang J., Paass G. Improved Phishing Detection Using Model-Based Features. CEAS Conference Proceedings. 2019.

18. RFC 3501 – Internet Message Access Protocol (IMAP4rev1). URL: https://datatracker.ietf.org/doc/html/rfc3501

19. RFC 5322 – Internet Message Format. URL: https://datatracker.ietf.org/doc/html/rfc5322

20. RFC 2045 – MIME Part One: Format of Internet Message Bodies. URL: https://datatracker.ietf.org/doc/html/rfc2045

21. RFC 8446 – The Transport Layer Security (TLS) Protocol Version 1.3. URL: https://datatracker.ietf.org/doc/html/rfc8446

22. OpenRouter Documentation. URL: https://openrouter.ai/docs

23. OpenRouter API Reference. URL: https://openrouter.ai/docs/api-reference

24. VirusTotal Documentation. URL: https://docs.virustotal.com

25. VirusTotal API Reference. URL: https://docs.virustotal.com/reference/overview

26. Electron Documentation. URL: https://www.electronjs.org/docs/latest

27. Electron IPC Communication Guide. URL: https://www.electronjs.org/docs/latest/tutorial/ipc

28. Node.js Documentation. URL: https://nodejs.org/docs/latest/api

29. MailParser Documentation. URL: https://nodemailer.com/extras/mailparser

30. OWASP Phishing Prevention Cheat Sheet. URL: https://cheatsheetseries.owasp.org

ДОДАТОК А

ЛІСТИНГ ПРОГРАМНОГО КОДУ ОСНОВНОГО МОДУЛЯ СИСТЕМИ

У даному додатку наведено програмний код основного модуля розробленого програмного засобу для виявлення фішингових електронних листів із використанням методів штучного інтелекту.

Основний модуль реалізовано мовою програмування JavaScript із використанням середовища Node.js та платформи Electron. Наведений код забезпечує виконання таких функцій:

– створення та налаштування головного вікна програмного засобу;

– організацію взаємодії між процесами Electron;

– підключення до поштового сервера через протокол IMAP;

– отримання та обробку електронних листів;

– аналіз вкладень та URL-адрес через сервіс VirusTotal;

– взаємодію з платформою OpenRouter для виконання інтелектуального аналізу повідомлень;

– формування та повернення результатів перевірки користувачу.

Лістинг програмного коду основного модуля системи наведено нижче.

```
const { app, BrowserWindow, ipcMain } = require('electron')
const path   = require('path')
const fs     = require('fs')
const https  = require('https')
const crypto = require('crypto')
const Imap   = require('imap')
const { simpleParser } = require('mailparser')
const { SYSTEM_PROMPT } = require('./prompt')

// ── Сховище ──────────────────────────────────────────────────────────
const STORE_FILE = path.join(app.getPath('userData'), 'store.json')

function storeRead() {
  try { return JSON.parse(fs.readFileSync(STORE_FILE, 'utf8')) } catch { return {} }
}
function storeWrite(data) {
  fs.writeFileSync(STORE_FILE, JSON.stringify(data, null, 2))
}
```

```
ipcMain.handle('store:get', (_, key) => storeRead()[key])
ipcMain.handle('store:set', (_, key, value) => {
  const d = storeRead(); d[key] = value; storeWrite(d)
})

// ── Вікно налаштувань ──────────────────────────────────────────
let settingsWin = null

function openSettingsWindow() {
  if (settingsWin && !settingsWin.isDestroyed()) {
    settingsWin.focus()
    return
  }
  settingsWin = new BrowserWindow({
    width: 480, height: 560,
    minWidth: 420, minHeight: 480,
    resizable: true,
    backgroundColor: '#13102b',
    frame: false,
    parent: win,
    modal: false,
    webPreferences: {
      preload: path.join(__dirname, 'settings-preload.js'),
      contextIsolation: true,
      nodeIntegration: false,
    },
  })
  settingsWin.loadFile(path.join(__dirname, 'settings.html'))
  settingsWin.on('closed', () => { settingsWin = null })
}

ipcMain.on('settings:open', openSettingsWindow)
ipcMain.on('settings:close', () => settingsWin?.close())

// Settings IPC handlers
ipcMain.handle('settings:get', () => {
  const d = storeRead()
  return {
    connected:    connected,
    connection:   d.phishing_state?.connection   || null,
    pollInterval: d.phishing_state?.settings?.pollInterval ?? 2,
    emailLimit:   d.phishing_state?.settings?.emailLimit   ?? 50,
    autoCheck:    d.phishing_state?.settings?.autoCheck    ?? true,
    aiEnabled:    d.phishing_state?.settings?.aiEnabled    ?? false,
    aiKey:        d.phishing_state?.settings?.aiKey        || '',
    aiModel:      d.phishing_state?.settings?.aiModel      || 'openai/gpt-oss-
120b:free',
    vtKey:        d.phishing_state?.settings?.vtKey        || '',
  }
})
```

```
ipcMain.handle('settings:save', (_, settings) => {
  const d = storeRead()
  if (!d.phishing_state) d.phishing_state = {}
  if (!d.phishing_state.settings) d.phishing_state.settings = {}
  Object.assign(d.phishing_state.settings, settings)
  storeWrite(d)
  // Notify main window to reload settings
  if (win && !win.isDestroyed()) win.webContents.send('settings:updated', settings)
})

ipcMain.handle('settings:saveConnection', (_, connection) => {
  const d = storeRead()
  if (!d.phishing_state) d.phishing_state = {}
  d.phishing_state.connection = connection
  if (!connection) d.phishing_state.connected = false
  storeWrite(d)
  if (win && !win.isDestroyed()) win.webContents.send('connection:updated', connection)
})

ipcMain.handle('settings:clearDeleted', () => {
  const d = storeRead()
  if (d.phishing_state) d.phishing_state.deletedIds = []
  storeWrite(d)
  if (win && !win.isDestroyed()) win.webContents.send('deleted:cleared')
})

// Керування вікном (для кастомного titlebar)
ipcMain.on('win:minimize', () => win?.minimize())
ipcMain.on('win:maximize', () => win?.isMaximized() ? win.unmaximize() : win.maximize())
ipcMain.on('win:close',    () => win?.close())

// ── мрація ──────────────────────────────────────────────
const VT_BASE = 'https://www.virustotal.com/api/v3'

function vtRequest(method, path, apiKey, body, contentType) {
  return new Promise((resolve, reject) => {
    const data = body ? (typeof body === 'string' ? body : JSON.stringify(body)) : null
    const req = https.request({
      hostname: 'www.virustotal.com',
      path:     '/api/v3' + path,
      method,
      headers: {
        'x-apikey':     apiKey,
        'Accept':       'application/json',
        ...(data ? { 'Content-Type': contentType || 'application/json', 'Content-
Length': Buffer.byteLength(data) } : {}),
      },
    }, (res) => {
      let raw = ''
```

```
      res.on('data', c => (raw += c))
      res.on('end', () => {
        try { resolve({ status: res.statusCode, data: JSON.parse(raw) }) }
        catch { resolve({ status: res.statusCode, data: { error: raw } }) }
      })
    })
    req.on('error', reject)
    if (data) req.write(data)
    req.end()
  })
}

// Чекаємо завершення аналізу (polling)
async function vtWaitAnalysis(analysisId, apiKey, maxWait = 60000) {
  const start = Date.now()
  while (Date.now() - start < maxWait) {
    await new Promise(r => setTimeout(r, 5000))
    const { status, data } = await vtRequest('GET', '/analyses/' + analysisId, apiKey)
    if (status === 200 && data.data?.attributes?.status === 'completed') {
      return data.data.attributes
    }
  }
  return null
}

// Форматуємо результат VT
function vtFormatResult(stats, meta) {
  if (!stats) return { verdict: 'unknown', malicious: 0, suspicious: 0, harmless: 0,
total: 0 }
  const malicious  = stats.malicious  || 0
  const suspicious = stats.suspicious || 0
  const harmless   = stats.harmless   || 0
  const undetected = stats.undetected || 0
  const total      = malicious + suspicious + harmless + undetected

  let verdict = 'clean'
  if (malicious >= 3)                    verdict = 'malicious'
  else if (malicious >= 1 || suspicious >= 3) verdict = 'suspicious'
  else if (suspicious >= 1)              verdict = 'possibly_suspicious'

  return { verdict, malicious, suspicious, harmless, undetected, total, meta: meta ||
null }
}

// Перевірка URL через VirusTotal
async function vtCheckUrl(url, apiKey) {
  try {
    const urlId = Buffer.from(url).toString('base64').replace(/=/g, '')
    const { status, data } = await vtRequest('GET', '/urls/' + urlId, apiKey)
```

```
    if (status === 200 && data.data?.attributes?.last_analysis_stats) {
      const stats = data.data.attributes.last_analysis_stats
      const meta  = {
        lastAnalysis: data.data.attributes.last_analysis_date,
        reputation:   data.data.attributes.reputation,
        categories:   data.data.attributes.categories,
      }
      return { url, ...vtFormatResult(stats, meta) }
    }

    const formBody = 'url=' + encodeURIComponent(url)
    const scanRes  = await vtRequest('POST', '/urls', apiKey, formBody, 'application/x-
www-form-urlencoded')

    if (scanRes.status !== 200) {
      return { url, verdict: 'error', error: 'Не вдалося відправити на сканування' }
    }

    const analysisId = scanRes.data.data?.id
    if (!analysisId) return { url, verdict: 'error', error: 'Немає ID аналізу' }

    const attrs = await vtWaitAnalysis(analysisId, apiKey, 30000)
    if (!attrs) return { url, verdict: 'pending', error: 'Аналіз ще не завершено' }

    return { url, ...vtFormatResult(attrs.stats) }
  } catch (err) {
    return { url, verdict: 'error', error: err.message }
  }
}


// Перевірка файлу через VirusTotal (за SHA256 хешем)
async function vtCheckFile(filename, fileData, apiKey) {
  try {
    const hash = crypto.createHash('sha256').update(fileData).digest('hex')

    const { status, data } = await vtRequest('GET', '/files/' + hash, apiKey)

    if (status === 200 && data.data?.attributes?.last_analysis_stats) {
      const stats = data.data.attributes.last_analysis_stats
      const meta  = {
        name:         data.data.attributes.meaningful_name || filename,
        type:         data.data.attributes.type_description,
        size:         data.data.attributes.size,
        sha256:       hash,
        lastAnalysis: data.data.attributes.last_analysis_date,
      }
      return { filename, hash, ...vtFormatResult(stats, meta) }
    }
    const boundary = '----VTBoundary' + Date.now()
```

```
    const parts = [
      '--' + boundary + '\r\n',
      'Content-Disposition: form-data; name="file"; filename="' + filename + '"\r\n',
      'Content-Type: application/octet-stream\r\n\r\n',
    ]
    const header = Buffer.from(parts.join(''))
    const footer = Buffer.from('\r\n--' + boundary + '--\r\n')
    const body   = Buffer.concat([header, fileData, footer])

    const uploadRes = await vtRequest('POST', '/files', apiKey, body.toString('binary'),
'multipart/form-data; boundary=' + boundary)

    if (uploadRes.status !== 200) {
      return { filename, hash, verdict: 'error', error: 'Не вдалося завантажити файл' }
    }

    const analysisId = uploadRes.data.data?.id
    if (!analysisId) return { filename, hash, verdict: 'error', error: 'Немає ID
аналізу' }

    const attrs = await vtWaitAnalysis(analysisId, apiKey, 60000)
    if (!attrs) return { filename, hash, verdict: 'pending', error: 'Аналіз ще не
завершено' }

    return { filename, hash, ...vtFormatResult(attrs.stats) }
  } catch (err) {
    return { filename, verdict: 'error', error: err.message }
  }
}

// IPC: перевірка URL через VT
ipcMain.handle('vt:checkUrls', async (_, { urls, apiKey }) => {
  if (!apiKey) return { ok: false, error: 'VT API ключ не вказано' }
  if (!urls?.length) return { ok: true, results: [] }

  const results = []
  for (const url of urls.slice(0, 10)) {  // максимум 10 URL
    const r = await vtCheckUrl(url, apiKey)
    results.push(r)
    await new Promise(res => setTimeout(res, 15000))  // 4 запити/хв = 15с між запитами
  }
  return { ok: true, results }
})

// IPC: перевірка вкладень через VT
ipcMain.handle('vt:checkFiles', async (_, { attachments, apiKey }) => {
  if (!apiKey) return { ok: false, error: 'VT API ключ не вказано' }
  if (!attachments?.length) return { ok: true, results: [] }

  const results = []
```

```
  for (const att of attachments.slice(0, 5)) {  // максимум 5 файлів
    const fileData = Buffer.from(att.content, 'base64')
    const r = await vtCheckFile(att.filename, fileData, apiKey)
    results.push(r)
    await new Promise(res => setTimeout(res, 15000))
  }
  return { ok: true, results }
})

// ── AI перевірка через OpenRouter ──────────────────────────────
const AI_URL           = 'https://openrouter.ai/api/v1/chat/completions'
const AI_MODEL_DEFAULT = 'openai/gpt-oss-120b:free'

function httpsPost(url, headers, body) {
  return new Promise((resolve, reject) => {
    const parsed = new URL(url)
    const data   = JSON.stringify(body)
    const req = https.request({
      hostname: parsed.hostname,
      path:     parsed.pathname,
      method:   'POST',
      headers:  { ...headers, 'Content-Length': Buffer.byteLength(data) },
    }, (res) => {
      let raw = ''
      res.on('data', c => (raw += c))
      res.on('end', () => {
        try { resolve({ status: res.statusCode, data: JSON.parse(raw) }) }
        catch { resolve({ status: res.statusCode, data: { error: raw } }) }
      })
    })
    req.on('error', reject)
    req.write(data)
    req.end()
  })
}
ipcMain.handle('ai:check', async (_, { email, apiKey, aiModel, vtResult }) => {
  if (!apiKey) return { ok: false, error: 'API ключ не вказано' }
  const MODEL = (aiModel || AI_MODEL_DEFAULT).trim()
  const toStr = v => {
    if (!v) return ''
    if (typeof v === 'string') return v
    if (typeof v === 'object') return v.text || v.value || ''
    return String(v)
  }

  // Витягуємо всі URL з листа для окремого аналізу
  const bodyText = toStr(email.body || email.html)
  const urlMatches = bodyText.match(/https?:\/\/[^\s"'<>)]+/gi) || []
  const urlList = urlMatches.length
    ? 'Знайдені посилання:\n' + urlMatches.slice(0, 10).map(u => '  - ' + u).join('\n')
```

```
    : 'Посилань не знайдено'

  // Вкладення
  const attachList = (email.attachments || []).length
    ? 'Вкладення: ' + email.attachments.map(a => toStr(a.filename || a.name ||
'невідомо')).join(', ')
    : 'Вкладень немає'

  // Формуємо блок з результатами VirusTotal якщо є
  let vtBlock = ''
  if (vtResult) {
    const lines = ['=== РЕЗУЛЬТАТИ VIRUSTOTAL ===']
    if (vtResult.urls?.length) {
      lines.push('Перевірені посилання:')
      vtResult.urls.forEach(u => {
        lines.push('  URL: ' + u.url)
        lines.push('  Вердикт: ' + u.verdict + ' (' + (u.malicious || 0) + '/' +
(u.total || 0) + ' детекцій)')
      })
    }
    if (vtResult.files?.length) {
      lines.push('Перевірені файли:')
      vtResult.files.forEach(f => {
        lines.push('  Файл: ' + f.filename)
        lines.push('  Вердикт: ' + f.verdict + ' (' + (f.malicious || 0) + '/' +
(f.total || 0) + ' детекцій)')
        if (f.hash) lines.push('  SHA256: ' + f.hash)
      })
    }
    vtBlock = lines.join('\n')
  }

  const systemPrompt = SYSTEM_PROMPT

  const emailData = [
    'ВІД: '          + toStr(email.from),
    'КОМУ: '         + toStr(email.to),
    'ТЕМА: '         + toStr(email.subject),
    'REPLY-TO: '     + toStr(email.headers?.replyTo),
    'RETURN-PATH: '  + toStr(email.headers?.returnPath),
    'ДАТА: '         + toStr(email.date),
    '',
    urlList,
    attachList,
    '',
    vtBlock ? vtBlock + '\n' : '',
    'ТІЛО ЛИСТА:',
    bodyText.slice(0, 2000),
  ].join('\n')
```

```
  try {
    const { status, data } = await httpsPost(AI_URL, {
      'Content-Type':  'application/json',
      'Authorization': 'Bearer ' + apiKey,
      'HTTP-Referer':  'http://localhost',
      'X-Title':       'Phishing Detector',
    }, {
      model: MODEL,
      messages: [
        { role: 'system', content: systemPrompt },
        { role: 'user',   content: emailData },
      ],
      temperature: 0.1,
      max_tokens:  700,
    })

    if (status !== 200) {
      const msg = data.error?.message || data.error || ('HTTP ' + status)
      return { ok: false, error: msg }
    }

    const raw   = data.choices?.[0]?.message?.content?.trim() || ''
    const clean = raw.replace(/```json\s*/gi, '').replace(/```\s*/g, '').trim()

    let parsed
    try {
      parsed = JSON.parse(clean)
    } catch {
      const rMatch = clean.match(/"result"\s*:\s*"(safe|suspicious|dangerous)"/)
      const sMatch = clean.match(/"score"\s*:\s*([\d.]+)/)
      if (!rMatch) return { ok: false, error: 'Невалідна відповідь від AI' }
      parsed = { result: rMatch[1], score: sMatch ? parseFloat(sMatch[1]) : 0.5,
reasons: [], analysis: '' }
    }

    const score = Math.min(Math.max(parseFloat(parsed.score) || 0, 0), 1)
    return {
      ok: true,
      result: {
        result:             ['safe','suspicious','dangerous'].includes(parsed.result) ?
parsed.result : 'suspicious',
        score:              Math.round(score * 100) / 100,
        reasons:            Array.isArray(parsed.reasons) ? parsed.reasons : [],
        urlAnalysis:        parsed.url_analysis  || '',
        attachmentAnalysis: parsed.attachment_analysis || '',
        aiAnalysis:         parsed.analysis      || '',
        source:             'ai',
        model:              MODEL,
      }
    }
```

```
  } catch (err) {
    return { ok: false, error: err.message }
  }
})

// ── IMAP стан ──────────────────────────────────────────────────────────
let imap      = null
let connected = false
let lastUID   = 0
let pollTimer = null
let win       = null

// ── Вікно ──────────────────────────────────────────────────────────────
app.whenReady().then(() => {
  win = new BrowserWindow({
    width: 1000, height: 660,
    minWidth: 800, minHeight: 560,
    backgroundColor: '#0d0d1a',
    frame: false,          // прибираємо стандартний заголовок і меню
    titleBarStyle: 'hidden',
    webPreferences: {
      preload: path.join(__dirname, 'preload.js'),
      contextIsolation: true,
      nodeIntegration: false,
    },
  })
  win.loadFile(path.join(__dirname, '../src/index.html'))
  win.on('closed', () => { stopPoll(); disconnectImap(); win = null })
})

app.on('window-all-closed', () => {
  stopPoll(); disconnectImap()
  if (process.platform !== 'darwin') app.quit()
})

// ── IMAP ───────────────────────────────────────────────────────────────
function disconnectImap() {
  stopPoll()
  if (imap) { try { imap.end() } catch (_) {} ; imap = null }
  connected = false; lastUID = 0
}

function stopPoll() {
  if (pollTimer) { clearInterval(pollTimer); pollTimer = null }
}

function startPoll(ms = 30000) {
  stopPoll()
  pollTimer = setInterval(async () => {
    if (!connected || !imap || !win) return
```

```
    try {
      await openBox()
      imap.search([['UID', `${lastUID + 1}:*`]], async (err, uids) => {
        if (err || !uids?.length) return
        const fresh = uids.filter(u => u > lastUID)
        if (!fresh.length) return
        const emails = await fetchByUIDs(fresh)
        for (const e of emails) {
          if (e.uid > lastUID) lastUID = e.uid
          if (win && !win.isDestroyed()) win.webContents.send('imap:new', e)
        }
      })
    } catch (_) {}
  }, ms)
}

ipcMain.handle('imap:connect', async (_, cfg) => {
  disconnectImap()
  return new Promise((resolve) => {
    imap = new Imap({
      user: cfg.email, password: cfg.password,
      host: cfg.host || 'imap.ukr.net', port: cfg.port || 993,
      tls: true, tlsOptions: { rejectUnauthorized: false },
      authTimeout: 10000, connTimeout: 15000,
    })
    imap.once('ready', () => {
      connected = true
      startPoll(30000)
      resolve({ ok: true })
    })
    imap.once('error', (e) => resolve({ ok: false, error: e.message }))
    imap.once('end',   () => { connected = false })
    imap.connect()
  })
})

ipcMain.handle('imap:disconnect', () => { disconnectImap(); return { ok: true } })

// ── Завантаження листів ──────────────────────────────────────
function openBox() {
  return new Promise((res, rej) =>
    imap.openBox('INBOX', false, (e, b) => e ? rej(e) : res(b))
  )
}

function fetchByUIDs(uids) {
  return new Promise((resolve, reject) => {
    if (!uids.length) return resolve([])
    const msgs = new Map()
    const f = imap.fetch(uids, { bodies: '', struct: true })
```

```
    f.on('message', (msg) => {
      let uid = null; const chunks = []
      msg.on('attributes', (a) => { uid = a.uid; msgs.set(uid, chunks) })
      msg.on('body', (s) => s.on('data', (c) => chunks.push(c)))
    })
    f.once('error', reject)
    f.once('end', async () => {
      const results = []
      for (const [uid, chunks] of msgs) {
        try {
          const p = await simpleParser(Buffer.concat(chunks))
          results.push({
            id: String(uid), uid,
            from:    p.from?.text    || '',
            to:      p.to?.text      || '',
            subject: p.subject       || '(без теми)',
            date:    p.date ? new Date(p.date).toLocaleString('uk-UA') : '',
            body:    p.text || '',
            html:    p.html || '',
            headers: {
              replyTo:    p.replyTo?.text               || '',
              returnPath: p.headers?.get('return-path') || '',
            },
            // Вкладення з base64 контентом для VT перевірки
            attachments: (p.attachments || []).map(a => ({
              filename: a.filename || 'unknown',
              size:     a.size     || 0,
              type:     a.contentType || '',
              content:  a.content ? a.content.toString('base64') : '',
            })),
            phishing: null,
          })
        } catch (_) {}
      }
      resolve(results)
    })
  })
}

ipcMain.handle('imap:fetch', async (_, limit = 50) => {
  if (!connected) return { ok: false, error: 'Не підключено' }
  try {
    await openBox()
    return new Promise((resolve, reject) => {
      imap.search(['ALL'], async (err, uids) => {
        if (err) return reject(err)
        if (!uids?.length) return resolve({ ok: true, emails: [] })
        const slice = [...uids].sort((a, b) => a - b).slice(-limit)
        lastUID = Math.max(...slice)
        const emails = await fetchByUIDs(slice)
```

```
        emails.sort((a, b) => b.uid - a.uid)
        resolve({ ok: true, emails })
      })
    })
  } catch (e) { return { ok: false, error: e.message } }
})
```

Метадані

ДОКУМЕНТ

Заголовок

БКР_Стасюк В.В.

Автор	Науковий керівник / Експерт	ІД документу
Стасюк В.В.	**Кондратюк**	**334270356**

ОРГАНІЗАЦІЯ

Назва організації	підрозділ
Kyiv National Economic University named after Vadym Hetman KNEU	**кафедра системного аналізу та кібербезпеки**

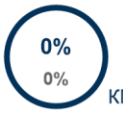
ЗВІТ

Дата звіту	Дата редагування
6/11/2026	**---**

Обсяг знайдених подібностей

Коефіцієнт подібності визначає, який відсоток тексту по відношенню до загального обсягу тексту було знайдено в різних джерелах. Зверніть увагу, що високі значення коефіцієнта не автоматично означають плагіат. Звіт має аналізувати компетентна / уповноважена особа.

КП 1



КП 2

КЦ

25	8889	75582
Довжина фрази для коефіцієнта подібності 2	Кількість слів	Кількість символів

Тривога

У цьому розділі ви знайдете інформацію щодо текстових спотворень. Ці спотворення в тексті можуть говорити про МОЖЛИВІ маніпуляції в тексті. Спотворення в тексті можуть мати навмисний характер, але частіше характер технічних помилок при конвертації документа та його збереженні, тому ми рекомендуємо вам підходити до аналізу цього модуля відповідально. У разі виникнення запитань, просимо звертатися до нашої служби підтримки.

Заміна букв	ß	0
Інтервали	A→	0
Мікропробіли	·	0
Білі знаки	ß	0
Парафрази (SmartMarks)	a	2

Джерела

Нижче наведений список джерел. В цьому списку є джерела із різних баз даних. Колір тексту означає в якому джерелі він був знайдений. Ці джерела і значення Коефіцієнту Подібності не відображають прямого плагіату. Необхідно відкрити кожне джерело і проаналізувати зміст і правильність оформлення джерела.

10 найдовших фраз — Колір тексту

#	НАЗВА ТА АДРЕСА ДЖЕРЕЛА URL (НАЗВА БАЗИ)	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)

1	https://www-sciencedirect-com.ezp.ukma.edu.ua/browse/author?value=%D0%A8%D1%83%D0%BC%D1%81%D1%8C%D0%BA%D0%B0,%20%D0%A1%D0%B2%D1%96%D1%82%D0%BB%D0%B0%D0%BD%D0%B0	16 (0.18 %)
2	ФКНТ_2025_126_Задорожна_О_В [redacted] 12/22/2025 Ukrainian national aviation university (ФКНТ Кафедра інтелектуальних кібернетичних систем)	12 (0.13 %)
3	Розробка застосунку для виявлення фішингових повідомлень електронної пошти з використанням штучного інтелекту [redacted] 5/18/2026 Odessa National Polytechnic University (ІІБРТ, Каф. кібербезпеки та програмного забезпечення)	10 (0.11 %)
4	https://magistr.ua/works/91/833388/	10 (0.11 %)
5	Розробка застосунку для виявлення фішингових повідомлень електронної пошти з використанням штучного інтелекту [redacted] 5/18/2026 Odessa National Polytechnic University (ІІБРТ, Каф. кібербезпеки та програмного забезпечення)	10 (0.11 %)
6	Розробка застосунку для виявлення фішингових повідомлень електронної пошти з використанням штучного інтелекту [redacted] 5/18/2026 Odessa National Polytechnic University (ІІБРТ, Каф. кібербезпеки та програмного забезпечення)	10 (0.11 %)
7	Розробка застосунку для виявлення фішингових повідомлень електронної пошти з використанням штучного інтелекту [redacted] 5/18/2026 Odessa National Polytechnic University (ІІБРТ, Каф. кібербезпеки та програмного забезпечення)	10 (0.11 %)
8	Розробка застосунку для виявлення фішингових повідомлень електронної пошти з використанням штучного інтелекту [redacted] 5/18/2026 Odessa National Polytechnic University (ІІБРТ, Каф. кібербезпеки та програмного забезпечення)	10 (0.11 %)
9	Розробка застосунку для виявлення фішингових повідомлень електронної пошти з використанням штучного інтелекту [redacted] 5/18/2026 Odessa National Polytechnic University (ІІБРТ, Каф. кібербезпеки та програмного забезпечення)	10 (0.11 %)
10	Розробка застосунку для виявлення фішингових повідомлень електронної пошти з використанням штучного інтелекту [redacted] 5/18/2026 Odessa National Polytechnic University (ІІБРТ, Каф. кібербезпеки та програмного забезпечення)	9 (0.1 %)

База даних RefBooks

#	ЗАГОЛОВОК	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)

Домашня база даних

#	ЗАГОЛОВОК	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)

Програма обміну базами даних (1 %)

#	ЗАГОЛОВОК	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)
1	Розробка застосунку для виявлення фішингових повідомлень електронної пошти з використанням штучного інтелекту [redacted] 5/18/2026	69 (7) (0.78 %)

[redacted] 5/18/2026
Odessa National Polytechnic University (ІІБРТ, Каф. кібербезпеки та програмного забезпечення)

2	ФКНТ_2025_126_Задорожна_О_В [redacted] 12/22/2025 Ukrainian national aviation university (ФКНТ Кафедра інтелектуальних кібернетичних систем)	12 (1) (0.13 %)
3	ФКНТ_2026_СП_СТРІК ДЮ [redacted] 5/25/2026 Ukrainian national aviation university (ФКНТ Кафедра інтелектуальних кібернетичних систем)	8 (1) (0.09 %)

Інтернет (0.29 %)

#	ДЖЕРЕЛО URL	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)
4	https://www-sciencedirect-com.ezp.ukma.edu.ua/browse/author?value=%D0%A8%D1%83%D0%BC%D1%81%D1%8C%D0%BA%D0%B0,%20%D0%A1%D0%B2%D1%96%D1%82%D0%BB%D0%B0%D0%BD%D0%B0	16 (1) (0.18 %)
5	https://magistr.ua/works/91/833388/	10 (1) (0.11 %)

Список прийнятих фрагментів

#	ЗМІСТ	КІЛЬКІСТЬ ОДНАКОВИХ СЛІВ (ФРАГМЕНТІВ)