НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
НАВЧАЛЬНО-НАУКОВИЙ ФІЗИКО-ТЕХНІЧНИЙ ІНСТИТУТ
Кафедра інформаційної безпеки

«До захисту допущено»
Завідувач кафедри

_____Дмитро ЛАНДЕ
(підпис)
“___”_____2024 р.

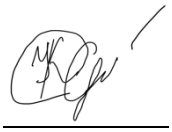
Дипломна робота

на здобуття ступеня бакалавра

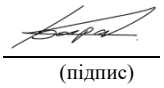
зі спеціальності 125 «Кібербезпека»

на тему: Програмний засіб виявлення фішингових атак за допомогою великих мовних моделей

Виконав: студент 4 курсу, групи ФБ-з01
(шифр групи)



_____Кисіль Денис Сергійович_____ _____
(прізвище, ім’я, по батькові) (підпис)

Керівник __к.т.н., доцент Барановський Олексій Миколайович_____ 
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Консультант _____ _____ _____
(назва розділу) (посада, вчене звання, науковий ступінь, прізвище, ініціали) (підпис)

Рецензент_____ _____
(посада, науковий ступінь, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.

Студент _____
(підпис)

Київ – 2024 року

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Навчально-науковий фізико-технічний інститут
Кафедра інформаційної безпеки

Рівень вищої освіти – перший (бакалаврський)
Спеціальність 125 Кібербезпека
Освітня програма Системи, технології та математичні методи кібербезпеки

ЗАТВЕРДЖУЮ
Завідувач
кафедри

_____ Дмитро ЛАНДЕ
(підпис)

«___»_____2024 р.

ЗАВДАННЯ

на дипломну роботу студенту

Кисілю Денису Сергійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Програмний засіб виявлення фішингових атак за допомогою великих мовних моделей, керівник роботи Барановський Олексій Миколайович, кандидат технічних наук, доцент кафедри інформаційної безпеки,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом по університету від «27» травня 2024 року № 2093-с
2. Термін подання студентом роботи 6 червня 2024 року.
3. Вихідні дані до роботи Програмний засіб для виявлення фішингових атак за допомогою великих мовних моделей.
4. Зміст роботи Створення програмного засобу для виявлення фішингових атак за допомогою великим мовних моделей.

5. Перелік ілюстративного матеріалу (із зазначенням плакатів, презентацій тощо)

6. Консультанти розділів роботи*

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

7. Дата видачі завдання: 14.04.2024

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітка
1	Отримання завдання	14.04.2024	Виконано
2	Формулювання теми дипломної роботи, визначення мети, постановка задач.	15.04.2024	Виконано
3	Огляд та опрацювання інформаційних джерел	15.04.2024	Виконано
4	Визначення структури дипломної роботи	15.04.2024	Виконано
5	Проходження переддипломної практики	15.05.2024	Виконано
6	Робота над першим розділом дипломної роботи	20.05.2024	Виконано
7	Робота над другим розділом дипломної роботи	25.05.2024	Виконано
8	Робота над третім розділом дипломної роботи	05.06.2024	Виконано
9	Тестування створеного програмного застосунку	04.06.2024	Виконано
10	Передзахист дипломної роботи	06.06.2024	Виконано
11	Доопрацювання відповідно до зауважень комісії, підготовка презентації до захисту.	15.06.2024	Виконано
12	Захист дипломної роботи	22.06.2024	

Студент _____ Денис КИСІЛЬ
(підпис) (ім’я, ПРІЗВИЩЕ)

Керівник роботи _____ Олексій БАРАНОВСЬКИЙ
(підпис) (ім’я, ПРІЗВИЩЕ)

---

* Консультантом не може бути зазначено керівника дипломної роботи.

Реферат

Дипломна робота складається з 4 розділів, містить 30 ілюстрацій, 2 таблиці, та 9 літературних посилань, загальний обсяг роботи – 57 сторінок.

Мета роботи – дослідження та розробка програмного засобу розпізнавання фішингових атак за допомогою великих мовних моделей для виявлення індикаторів фішингу в електронних листах.

Метод дослідження – аналіз найпоширеніших підходів для виявлення фішингових листів, створення програмного засобу системи виявлення фішингу, який включає функціонал обробки вхідних електронних листів для отримання даних придатних для подальшого аналізу, функціонал взаємодії з великою мовною моделлю з метою подальшої обробки, функціонал прийняття рішення на основі отриманих результатів.

Наукова новизна дослідження полягає в тому, що був створений і реалізований програмний засіб для виявлення фішингових електронних листів із застосуванням великих мовних моделей.

Прикладна програма була розроблена з використанням мови програмування Python версії 3.12.

Ключові слова: фішинг, великі мовні моделі, фішингові листи, індикатори фішингу.

Abstract

The thesis consists of 4 chapters, contains 30 illustrations, 2 tables, 9 literary references and the total volume of the work is 57 pages.

The purpose of the work is to research and develop a software tool for recognizing phishing attacks using large-scale language models to detect phishing indicators in emails.

Research method - analysis of the most common approaches to detect phishing emails, creation of a software tool for phishing detection system, which includes the functionality of processing incoming emails to obtain data suitable for further analysis, the functionality of interaction with a large language model for further processing, the functionality of making decisions based on the results obtained.

The scientific novelty of the study is that a software tool for detecting phishing emails using large language models was created and implemented.

The application program was developed using the Python programming language version 3.12.

Keywords: phishing, large language models, phishing emails, phishing indicators.

Зміст

Перелік умовних позначень, символів, одиниць, скорочень і термінів

Фішинг — це тип хакерської атаки із застосуванням соціальної інженерії, під час якої зловмисник, часто видаючи себе за законну особу або сутність, з метою отримати конфіденційну інформацію від потенційної жертви або заразити її пристрій шкідливим програмним забезпеченням.

Фішинговий лист — це електронний лист що містить індикатори фішингу, який використовують зловмисники для проведення фішингової атаки.

Фішингове посилання — це URL-адреса, яка створена зловмисником з метою обманом перевести потенційну жертву фішингової атаки на зловмисний сайт, що часто виглядає як легітимний сайт відомого інтернет ресурсу.

Індикатор фішингу – це конкретні технічні параметри або дані, які можуть вказувати на фішингову атаку.

Зловмисник — це особа або організація, яка прагне отримати несанкціонований доступ до інформаційних систем, ресурсів або даних, щоб їх пошкодити, викрасти або використати в особистих цілях.

Макроси — це невеликі програми, які можуть автоматизувати завдання, але можуть бути використані щоб виконати шкідливий код.

Обфускована URL адреса — це URL, який було навмисно змінено або заплутано для того, щоб приховати справжнє призначення або зміст адреси.

Вступ

Актуальність роботи

Сьогодні складно уявити життя сучасної людини без смартфону, соціальних мереж та онлайн сервісів. Сучасні інформаційні технології забезпечують користувачів онлайн інструментами здатними підвищити продуктивність, пришвидшити спілкування, полегшити доступ до інформації.

Цифрові каналами зв'язку дозволяють працювати над складними проектами, не перебуваючи в одному місці. Роботодавці пропонують робітникам працювати з дому, за для підвищення комфорту та продуктивності співробітників.

Цифровізація банківських послуг дозволяє клієнтам здійснювати транзакції, переглядати баланси рахунків, відкривати рахунки і навіть займатися інвестиціями через мобільні додатки та веб-портали.

Цифровізація послуг державних органів дозволяє зменшити кількість державних службовців, та знизити ризики корупції. Яскравим прикладом може стати єдиний портал державних послуг Дія.

Однак у такого високого рівня цифровізації є й недоліки. Одним із найбільших є той факт, що високий рівень цифровізації відкрив для кіберзлочинців безліч нових можливостей для здійснення фішингових атак.

Прикладом, успішної фішингової атаки є атака на американську нафтопровідну компанію Colonial Pipeline у 2021 році. Внаслідок даної атаки було зупинено роботу трубопроводів на кілька днів, а переривання постачання палива призвело до зростання цін на нього.

Саме завдяки фішингового електронного листа зловмисники змогли отримати дані автентифікації співробітника компанії та встановити первинний доступ до внутрішньої мережі компанії.

Цілями для фішингових атак зловмисників стають не тільки співробітники великих компаній або урядових установ, а і звичайні користувачі.

Щодня відбуваються десятки тисяч фішингових атак, під час яких зловмисники застосовують хитрі техніки, щоб змусити людину розкрити особисту інформацію або встановити шкідливе програмне забезпечення на її пристрої.

Виявлення та запобігання фішингових атак залишається надзвичайно важливими завданнями інформаційної безпеки. Розпізнавання фішингових листів за допомогою технологій великих мовних моделей (Large Language Models) є перспективним підходом до боротьби із загрозами типу фішинг.

Мета і завдання дослідження

Метою моєї дипломної роботи є дослідження та розробка програмного засобу розпізнавання фішингових веб-сторінок із застосуванням технології великих мовних моделей.

Основною метою є покращення безпеки користувачів електронних поштових скриньок шляхом виявлення та блокування фішингових електронних листів, що дозволить запобігти крадіжкам особистих даних, фінансовим шахрайствам та реалізації більш складних хакерських атак.

Об’єкт дослідження

Об'єктом дослідження у дипломній роботі є процес виявлення та ідентифікації фішингових атак в цифровому середовищі за допомогою методів машинного навчання, зокрема великих мовних моделей (LLM).

Предмет дослідження

Предметом дослідження у дипломній роботі є методи та алгоритми виявлення фішингових атак за допомогою великих мовних моделей (LLM). Це включає використання технологій обробки природної мови (NLP), машинного навчання та аналізу даних для створення і вдосконалення програмного засобу, здатного ефективно ідентифікувати фішингові повідомлення.

Наукова новизна одержаних результатів

Відкриття нових можливостей в області розпізнавання фішингових листів із застосуванням великих мовних моделей.

Практичне значення одержаних результатів

Створений програмний засіб може бути використаний для інтеграції в поштові сервіси для підвищення ефективності виявлення фішингових атак.

Розділ 1 Дослідження елементів електронного листа, технік створення фішингових листів, визначення індикаторів фішингу, що можуть бути використані для виявлення фішингової атаки.

Фішингові атаки стали однією з найпоширеніших і найшкідливіших форм кіберзагроз, націлених як на приватних осіб, так і на організації. Ці атаки передбачають використання фішингових електронних для того, щоб обманом змусити одержувачів розкрити конфіденційну інформацію, наприклад облікові дані для входу в систему, фінансові дані або дані, що посвідчують особу або встановити шкідливе програмне забезпечення на її (жертви фішингової атаки) комп'ютер з метою отримання віддаленого доступу, або шифрування даних жертви для вимагання викупу. Витонченість і поширеність фішингових атак зумовили необхідність розробки надійних механізмів виявлення для захисту від цих загроз.

Основна мета цього розділу - дослідити структуру типових електронних листів, методи та техніки фішингових атак, що використовуються кіберзлочинцями та їх основі визначити індикатори фішингу що можуть міститись в ключових елементах електронного листа та бути використані для виявлення фішингових атак. Розуміння цих елементів та індикаторів має вирішальне значення для розробки ефективного програмного засобу, що використовує великі мовні моделі для ефективного виявлення фішингових листів.

1.1 Елементи електронного листа

Вміст типового повідомлення електронної пошти включає в собі різні елементи які можна розділити на два типи: мета дані та корисні дані.

Мета дані – інформація про електронний лист, яка не є частиною фактичного вмісту повідомлення, але надає важливі відомості про нього. Мета дані включають в себе такі елементи:

- Time – дані про дату і час. Електронні листи, котрі надіслані в нестандартні години або дати, які не відповідають типовій діяльності відправника, можуть бути вторинним попередженнями про фішингову атаку;
- From – дані про відправника повідомлення. Є одним із ключових елементів, що може містити індикатори фішингу. Фішингові електронні листи часто використовують підроблені або оманливі адреси відправників, щоб виглядати так, ніби вони походять із надійного джерела;
- To – дані про отримувача повідомлення. Адреса одержувача може виявити ознаки спроби фішингу, якщо вона є частиною списку масової розсилки або містить кількох одержувачів без очевидного зв’язку.
- CC – дані про отримувача копії повідомлення.
- BCC – дані про отримувача секретної копії повідомлення та видна лише йому, зазвичай не міститься електронному листі.
- Message-ID – ідентифікатор повідомлення.
- Subject – тему листа.
- Headers – дані заголовків (тип даних та версію MIME).

Корисні дані – дані що містять фактичний зміст повідомлення. Корисні дані повідомлення включають в себе:

- Text body – основний текст, написаний відправником, який може бути у форматі звичайного тексту або HTML. Є ключовим елементом у фішинговому листі. Застосовуючи соціальну інженерію в основному тексті повідомлення має на меті викликати довіру та переконати одержувача виконати певну дію, потрібну зловмиснику, наприклад перейти за фішинговим посиланням, завантажити інфікований файл, або надати конфіденційну інформацію.
- URL (Universal Resource Locator) – унікальні ідентифікатори, що використовуються для пошуку ресурсів в мережі інтернет та можуть бути включені в текст повідомлення. У фішинговому листі текст, який

відображається для посилання, може бути оманливим, відображаючись як законна URL-адреса під час спрямування на шкідливий сайт. Фішингові URL-адреси можуть використовувати незначні варіації законних доменних імен, додаткових субдоменів або незнайомих доменів верхнього рівня, а також використовувати служби скорочення URL-адрес, щоб приховати кінцевий пункт призначення.

- Attached files (вкладені файли) – файли які додаються до електронного листа, наприклад, документи, зображення, архіви та інші типи файлів. Фішингові листи часто містять виконувані файли, документи з макросами або стиснені файли, які можуть запускати шкідливий код під час відкриття, хоча на перший виглядають зовсім безпечними.
- Форматування тексту – інформація про форматування тексту: шрифт, розмір тексту, курсив, підкреслення тощо (застосовно якщо електронний лист відправлено у форматі HTML або Rich Text).

Серед розглянутих елементів типових електронних листів було обрані такі, що найчастіше використовуються у фішингових листах, та можуть бути проаналізовані для ефективного виявлення та попередження фішингових атак:

- From – дані про відправника повідомлення
- Text body – основний текст електронного повідомлення
- URL посилання
- Attached files (Вкладені файли)

1.2 Фішингова адреса відправника, техніки створення

Мета фішинговий адрес відправника повідомлення за допомогою візуальної схожості видати себе легітимного суб’єкта для створення довіри у потенційної жертви атаки, а також обходити спам-фільтри та системи безпеки електронної пошти.

Поширені прийоми та методи, які використовують зловмисники для створення фішингових адрес електронної пошти та підвищення довіри до своїх атак:

1. Підробка відображуваного імені: техніка передбачає зміну відображуваного імені відправника електронної пошти, щоб воно нагадувало надійне джерело. Хоча фактична адреса електронної пошти може відрізнятися, відображуване ім’я (наприклад, «Служба підтримки клієнтів НАЗВА ОРГАНІЗАЦІЇ» або «ІТ-підтримка НАЗВА ОРГАНІЗАЦІЇ») змінюється, щоб одержувач виглядав легітимним. Цей метод використовує тенденцію користувачів зосереджуватися на відображеному імені, а не на повній адресі електронної пошти.
2. Підробка домену: зловмисники створюють адреси електронної пошти, які імітують законні домени, використовуючи незначні варіації. Це може включати подвоєння, заміну схожих символів (наприклад, використання «rn» замість «m») або додавання додаткових слів, літер, символів (наприклад, «@secure-privatbank.ua»), або допущення помилок легітимних доменів (наприклад «goolge.com»). Зловмисники розраховують що невеликі зміни залишаться непоміченими одержувачами.
3. Використання скомпрометованих легітимних облікових записів. Фішингові листи, надіслані зі скомпрометованих облікових записів, можуть бути дуже ефективними, оскільки вони надходять із надійних джерел і з меншою ймовірністю будуть позначені фільтрами безпеки.
4. Внутрішній спуфінг: у деяких випадках зловмисники отримують доступ до внутрішньої системи електронної пошти організації та проводять фішингову атаку зсередини організації використовуючи поштову адресу цієї організації.
5. Використання безкоштовних, одноразові, анонімних сервісів (наприклад, Gmail, Yahoo, Yopmail). Наприклад, електронний лист від фінансової установи з використанням адреси Gmail.

1.3 Фішингові вкладені URL посилання, техніки створення

Розуміння методів, які використовуються для створення фішингових URL-адрес і супутніх посилань, має вирішальне значення для визначення індикаторів фішингу та їх ефективного виявлення. Поширені прийоми та методи, які використовують зловмисники для створення вкладених фішингових URL адрес:

1. Атака типу IDN Homograph: використання особливостей багатомовних комп'ютерних систем, де різні логічні символи можуть мати ідентичний вигляд. Наприклад, xn--pple-43d.com може відображатися як apple.com у певних браузерах.
2. Скорочення URL-адреси: використання сервісів скорочення URL-адрес (наприклад, bit.ly, tinyurl.com), щоб приховати явно фішингові URL-адреси та ускладнити для одержувачів розпізнавання загрози, доки вони не натиснуть посилання. До даної техніки можна додати розширену: ланцюгове скорочування URL-адреси, що характерне послідовним використанням декількох сервісів скорочення URL-адрес, щоб ще краще приховати кінцевий пункт призначення.
3. Законні субдомени: використання субдомени, які виглядають законними, викликати у потенційної жертви атаки довіру до URL-адреси. Наприклад, http://privat-bank.login.com. До даної техніки можна додати розширену: використання надмірної кількісті субдоменів, що робить URL-адресу заплутаною або надто складною, наприклад http://secure-update-login.account.verify.bonk.com.
4. Оновлення мета тегів або перенаправлення JavaScript: використання можливостей Meta Tags та JavaScript щоб перенаправити користувача із початкової, здавалося б, безпечної сторінки на шкідливий сайт. Початкова URL-адреса може виглядати нешкідливою, але переспрямування відбувається після завантаження сторінки. Ця техніка може обійти первинну перевірку посилання.

5. Оманливі гіперпосилання: використання гіперпосилань зображення або текст посилання відрізнятися від фактичної URL-адреси, на яку воно вказує. Наприклад, «Натисніть тут, щоб оновити свій обліковий запис» може відображати законну URL-адресу, як-от http://www.privatbank.ua, але фактично посилатися на http://fakesite.ru.
6. Орфографічні помилки : використання доменів з поширеними орфографічними помилками легітимних ресурсів, приклад privetbank.com .
7. Коди відстеження: використання кодів відстеження маркетингових кампаній для оцінки ефективності фішингової атаки і подальшого вдосконалення застосованих тактик.
8. Відсутність HTTPS протоколу: використання фішинговими сайтами незахищеного протоколу HTTP. Однак використання HTTPS протоколу не є остаточним показником легітимності, оскільки фішингові сайти можуть також використовувати HTTPS із дійсними сертифікатами SSL, щоб виглядати безпечно.
9. Підроблені сертифікати: використання підроблених або викрадених SSL сертифікатів, щоб створити хибне відчуття безпеки.

1.4 Фішинговий текст електронного листа, техніки створення

Фішинговий текст основою фішингового електронного листа, оскільки збирає на собі увагу потенційної жертви, та використовуючи психологічні прийоми реалізує мету викликати довіру та змусити до дії, необхідної зловмиснику. Поширені прийоми та методи, які використовують зловмисники для створення текстового наповнення фішингових листів:

1. Термінова мова: у фішингових електронних листах часто використовується термінова мова, щоб створити відчуття негайної необхідності. Поширені прийоми та методи, які використовують зловмисники для створення текстового наповнення фішингового листа: «Потрібні негайні дії», «Ваш обліковий запис буде призупинено» або

«Платіж прострочено», тощо, щоб створити відчуття негайної необхідності та змусити одержувачів діяти швидко без ретельної перевірки.

2. Загрози та наслідки: неправдива інформація про компрометацію облікового запису та необхідності вжити негайних дій або погрози наслідками у разі невиконання вимог, наприклад призупиненням облікового запису, судовим позовом або фінансовими втратами, з метою викликати страх щоб знизити обережність одержувача.
3. Видача себе за довірених осіб: видавання за авторитетних осіб або авторитетні організацій, наприклад окремих керівників, банків, державні установи. Характерне використання офіційних назв, логотипів, офіційної мови та підписів в кінці повідомлення.
4. Цікавість та азарт: використання природної людської цікавості та азарту, пропонуючи привабливі на перший погляд можливості, такі як виграш призу, повернення коштів або доступ до ексклюзивного контенту.
5. Персоналізація: Фішингові електронні листи часто містять персоналізовані елементи, такі як ім'я одержувача, посада або інші конкретні дані, щоб надати електронному листу легітимного вигляду. Цю інформацію можна отримати через попередні витоки даних або соціальні мережі.
6. Недоречні запити: створення правдоподібного сценарію або приводу для виправдання запиту недоречної інформації. Наприклад, у повідомленні може стверджуватися, що воно від ІТ-відділу, який проводить перевірку безпеки, або від постачальника, який підтверджує транзакцію.

1.5 Вкладені файли фішингових листів

У фішингові електронні листах часто використовуються вкладені файли, які виглядають як справжні документи чи файли. Ці вкладені файли є

вектором для доставки зловмисного програмного забезпечення або для того, щоб обманом одержувачів надати конфіденційну інформацію.

Популярні формати вкладених файлів що використовуються зловмисниками під час фішингових атак:

- Виконувані файли (.exe, .bat, .cmd): щоб інсталювати в систему зловмисне програмне забезпечення, наприклад віруси, трояни або програми-вимагачі.
- Файли документів з вбудованими макросами (.docm, .xlsm, .pptm)
- Стиснуті файли (.zip, .rar)
- PDF-файли (.pdf) з вбудованими сценаріями або посилання на шкідливі веб-сайти
- Файли зображень (.jpg, .png, .gif)

1.6 Індикатори фішингових листів

Індикатор фішингу – це конкретні технічні параметри або дані, які можуть вказувати на фішингову атаку.

Саме наявність індикаторів фішингу будуть підставою для оцінки електронного листа на предмет фішингу.

1. Індикатори фішинговий адрес відправника електронної пошти:
    1.1. Невідповідні або подібні домени.
    1.2. Домені адреси одноразових, анонімних сервісів електронної пошти.
    1.3. Розбіжності між іменем і адресою електронної пошти.
    1.4. Загальні адреси електронної пошти.
2. Індикатори фішингових URL посилань:
    2.1. Домени з орфографічними помилками, додатковими символами, знаками або словами.
    2.2. Нетипові домени вищого рівня (Top-level domains).
    2.3. Надмірна кількість субдоменів.
    2.4. Легітимні субдомени з нетиповими доменами.

2.5. Відсутність HTTPS протоколу.

2.6. Обфуксовані URL адреси.

2.7. Використання сервісів скорочення посилань.

3. Індикатори основного тексту фішингових повідомлень:

3.1. Створення відчуття невідкладності та терміновості.

3.2. Виклик емоційної реакції, наприклад страху, тривоги, тощо.

3.3. Виклик зацікавленості, демонстрація ексклюзивності, можливостей.

3.4. Використання легітимних назв та логотипів.

3.5. Недоречні запити (особистої, конфіденційної інформації).

3.6. Звернення до соціальних, політичних, подій, доказів, авторитетів.

4. Індикатори вкладений файлів що містять шкідливе програмне забезпечення:

4.1. Недоречні вкладення.

4.2. Виконувані файли.

4.3. Файли що містять макроси.

4.4. Стиснені або архівовані файли.

4.5. Файли з подвійним форматом.

Висновок до першого розділу

В даному розділі було розглянуто елементи типового електронного листа, обрано елементи що будуть підлягати перевірці, описані техніки що використовуються зловмисниками створення фішингових листів, а також визначено індикатори фішингу, що можуть бути використані для під час виявлення фішингових атак у розробленому програмному засобі.

Розділ 2 Великі мовні моделі, потенціал у виявленні індикаторів фішингу, порівняльний аналіз рішень, проектування антифішингової системи

2.1 Великі мовні моделі та їх потенціал у виявленні індикаторів фішингу

Проривний та швидкий розвиток штучного інтелекту та машинного навчання демонструє величезний потенціал у широкому застосуванні. Серед найбільш публічно відомих розробок — великі мовні моделі (Large language models), які продемонстрували надзвичайні можливості розуміння та створення тексту на природній мові, та ефективні від завдань обробки природної мови (NLP) до вирішення складних проблем.

LLM (або Large Language Models) - це потужні моделі машинного навчання, засновані на передових архітектурах нейронних мереж, зокрема трансформаторах. Ці моделі призначені для обробки, розуміння та генерування людської мови шляхом навчання на великих обсягах текстових даних. Таке навчання передбачає доступ моделі до надвеликих і різноманітних наборів даних, що дозволяє їй вивчати лінгвістичні моделі, контекстні значення і тонкощі граматики та синтаксису. Аналізуючи текст великі мовні моделі можуть інтерпретувати та класифікувати зміст, визначати теми та настрої і навіть виявляти аномалії чи шаблони, які можуть вислизнути від алгоритмів виявлення шаблонів. Уваги також заслуговує здатність розуміти велику кількість природній мов. Великі мовні моделі можуть розуміти та перекладати текст з однієї мови на іншу з високим ступенем точності, передаючи не лише буквальне значення, а й культурні нюанси та ідіоматичні вирази.

У контексті кібербезпеки великі мовні моделі є перспективними для виявлення фішингових атак. Оскільки фішингові атаки здебільшого базуються на маніпулятивному спілкуванні природною мовою з використанням соціальної інженерії, ці моделі можуть аналізувати та виявляти підозрілі шаблони, формулювання та контекст в електронних листах. Використовуючи можливості обробки природної мови, великі мовні

моделі можуть виявляти індикатори фішингу, тим самим підвищуючи точність і швидкість виявлення загроз. Крім того, великі мовні моделі можна донавчати та вдосконалювати з часом.

2.2 Обрання великої мовної моделі для аналізу індикаторів фішингу

Для обрання великої мовної моделі котра буде використана для побудови програмного засобу виявлення фішингу було сформовано порівняльну таблицю з наявних передових рішень великих мовних моделей.

Порівняльна таблиця містить різні характеристики, зокрема такі що є важливими для вибору підходящого варіанту:

- Країна походження – в умовах геополітичної напруженості є важливим критерієм.
- Кількість параметрів – характеристика що демонструє потенційну здатність до узагальнення, складність завдань що можуть бути оброблені, та масштаб даних що були використані для навчання цієї моделі. Чим показник вищий, тим ефективніше модель буде виконувати необхідні задачі.
- Доступність – критерій що демонструє можливості розгортання моделі локально, або із використанням вже розгорнутої на технічних потужностях розробника за допомогою Web інтерфейсу, застосунку, або API.
- Розгортання LLM моделі локально потребує великої кількості обчислювальних ресурсів та специфічної архітектури відеокарт оптимізованих для задач машинного навчання, тому було віддано перевагу використанню моделям що пропонують сервіс API.
- Кількість підтримуваних мов – важливий критерій що демонструє лінгвістичну навченість моделі та є прямою характеристикою ефективності роботи з природними мовами.

- Формат доступу – є важливим критерієм, оскільки характеризує драйвери розвиту моделі. Великі мовні моделі що є комерційними продуктами мають більше можливостей та стимулів до швидкого оновлення, виправлення помилок, розширення функціоналу, донавчання моделі та є конкурентнішими.

Модель	Вендор	Країна походження	Кількість параметрів	Доступність	Формат доступу	Кількість підтримуваних мов
GPT-4	OpenAI	США	1,76 трильйонів	API, Веб інтерфейс, застосунок	Платний	95
Gemini Ultra	Google	США	175 мільярдів	API, Застосунок	Платний	38
Llama 3	Meta AI	США	70 мільярдів	Локальне розгортання	Політика відкритого коду	33
Glaude 3 Opus	Anthropic	США	~60 мільярдів	API, застосунок	Платний	25
Falcon 180B	Falcon LLM	ОАЄ	180 мільярдів	Хмарне, локальне розгортання		11

Таблиця 1. Порівняння великих мовних моделей.

Виконавши порівняльний аналіз найбільш прогресивних публічних рішень в області великих мовних моделей було обрано модель OpenAI GPT-4, оскільки вона має порівняно високу кількість параметрів та підтримувальних мов, а також можливість доступу до моделі за допомогою API сервісу.

2.3 Використання API OpenAI GPT-4

Для обміну даними з обраною великою мовною моделлю OpenAI GPT-4 буде використано API інтерфейс. Щоб використовувати GPT-4, необхідно спочатку отримати доступ до API від OpenAI. Це передбачає створення облікового запису, та поповнення рахунку.

Після реєстрації надається доступ до панелі управління ключами на сайті platform.openai.com/api-keys. Ключі API необхідний для автентифікації ваших запитів до API GPT-4, і мусить бути збережений в безпеці та конфіденційності.

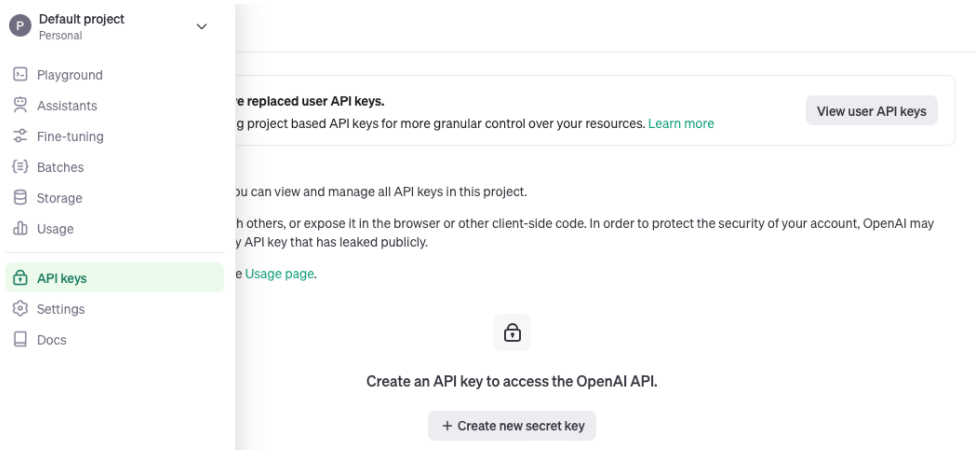


Рисунок 1. Панель управління API ключами OpenAI

Оскільки інтерфейс API GPT-4 заснований на HTTP запитах, це робить його сумісним з будь-якою мовою, яка може відправляти HTTP запити.

Python одна з найпопулярніших мов програмування, особливо у сфері машинного навчання і штучного інтелекту. Вендор OpenAI надає офіційну бібліотеку Python, яка спрощує взаємодію з API GPT-4. Для створення модулей взаємодії з LLM моделлю GPT-4 буде використано Python версії 3.12.

Для доступу до моделі GPT-4 через API інтерфейс необхідно мати ключ.

2.4 Налаштування великої мовної моделі для виконання задач виявлення індикаторів фішингу

Ефективність використання GPT-4 для виявлення індикаторів фішингу значною мірою залежить від створення добре продуманих підказок. Підказка або промпт (з англійської “prompt”) - це ретельно розроблений фрагмент тексту або інших вхідних даних що використовується як директиви або інструкції, що надаються мовній моделі, які спрямовують її на генерування змістовних і релевантних результатів.

Ретельно продумана підказка, її компоненти дозволяють оптимізувати велику мовну модель для ефективного виконання специфічних завдань. Добре сформована підказка містить чітке визначення завдання, відповідний контекст і конкретні запитання чи інструкції.

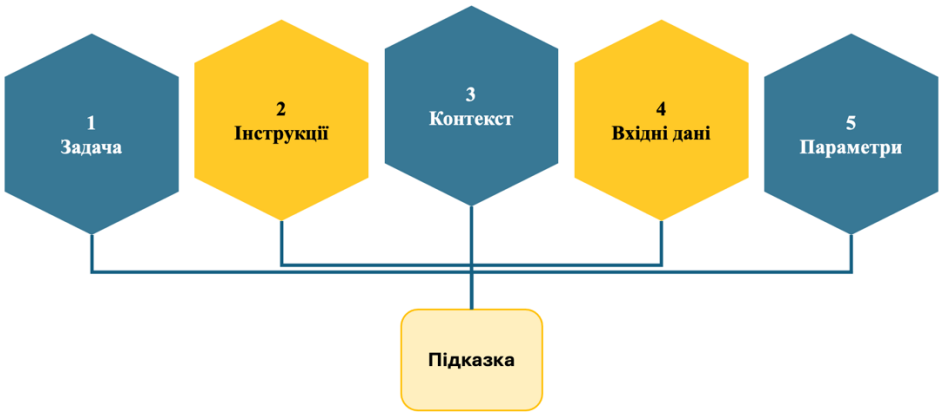


Рисунок 2. Структура підказок, компоненти

Компонент «1. Задача» може містити такі елементи, як:

- Роль - образ, яку ШІ повинен прийняти
- Команда – дієслово, що керує моделлю
- Тема - предметна область фокусування
- Запит - конкретне запитання, на яке потрібно відповісти

До складу компонента «2. Інструкції можуть» входити такі елементи:

- Вихід - Очікувані атрибути згенерованого контенту
- Структура - формат організації, розділи, потік
- Дозволено - Прийнятні якості та зміст
- Заборонено - Неприйнятні якості та зміст
- Тези/Ідеї - Конкретні концепції, які слід включити
- Приклади - Приклади для ілюстрації бажаного результату

Компонент «3. Контекст» включає в себе такі елементи:

- Цільова аудиторія - передбачуваний споживач контенту
- Перспектива - Точка зору, яку потрібно прийняти
- Мета - цілі та мотивації
- Додаткова інформація - Додаткова довідкова інформація

Компонент «4. Вхідні дані» може визначати типи можливих вхідних даних.

Компонент «5. Параметри» може містити такі налаштування:

- Температура - Рівень креативності/непередбачуваності
- Довжина - розмір згенерованого контенту
- Top-p - Ймовірність малоймовірних результатів
- Штраф - Попередження небажаних результатів
- Модель - Використовувана система ШІ

Підказка для налаштування моделі для виявлення індикатора фішингу, наприклад, фішингової адреси відправника може бути такою:

```
Role: Model for detecting phishing indicators
Purpose: To analyze an email element for phishing indicators
Task: Check the input data for phishing indicators
Input: The FROM element of an email in the form of text: First Name Surname <nickname@email.com>
Output: A binary response in the form of the text “true” or “false”

Instructions:
- Output: Provide a binary response as "true" if phishing indicators are present, otherwise "false"
- Structure: Single word output
- Do's:
  - Check for discrepancies between the display name and email address
  - Identify suspicious domain names
  - Highlight unusual or generic sender names
- Don'ts:
  - Do not analyze content outside the FROM element
  - Do not provide explanations or details, only binary response
- Points/Ideas:
  - Recognize common phishing tactics such as: Mismatched or similar domains; Disposable, anonymous email service addresses;
Name and email address discrepancies; Shared email addresses; and other indicators and combinations of indicators.

Context:
- Target Audience: Security analysts and automated phishing detection systems
- Perspective: Objective and analytical
- Purpose: To evaluate the legitimacy of the email sender's address and detect potential phishing indicators
- Supplementary Info: Use known phishing indicators and patterns to inform analysis

Parameters:
- Temperature: 0.0
- Length: 1 token
- Top-p: 1.0
- Penalty: No penalty

Input: [tmp_m4_from.txt]
```

Рисунок 3. Підказка для виявлення індикаторів фішингової адреси відправника

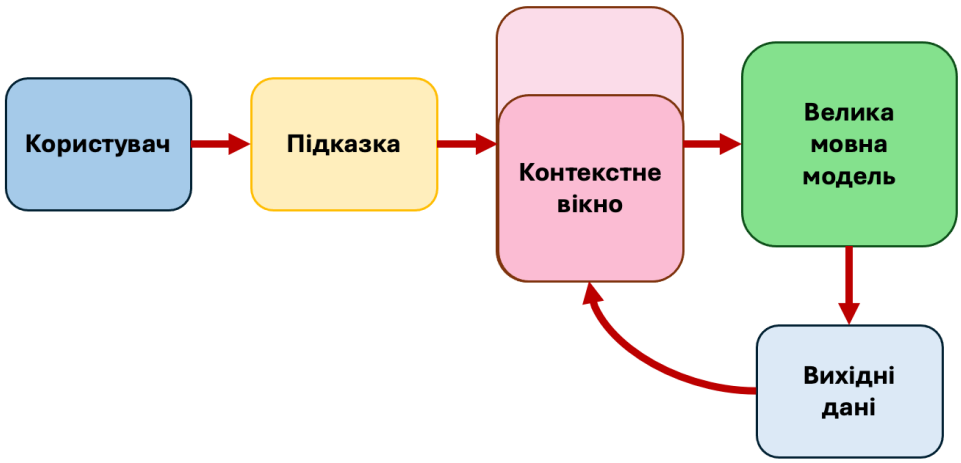


Рисунок 4. Цикл підказок

Ітеративний цикл підказок є ключовим для сприяння розвитку взаємодії між системою засобом фішингу та великою мовною моделлю:

1. Встановлена підказка надає початкову підказку, щоб розпочати взаємодію та спрямувати модель до конкретного завдання.
2. Створення ретельно продуманої підказки забезпечує модель оптимальними інструкціями та контекстом. Такі фактори, як

контекст, точка зору та формат відповіді, допомагають керувати відповіддю LLM.

3. Велика мовна модель обробляє задану підказку в поточному контекстному вікні, щоб сформувати відповідну текстову відповідь. Відповідь базується на умовах завданих підказкою, що відображають інженерію підказки.
4. Згенерована відповідь моделі рекурсивно додається до контекстного вікна. Також модель генерує відповіді послідовно, кожного разу використовуючи свою попередню відповідь як частину вхідних даних для наступного кроку. Це дозволяє моделі спиратися на власні попередні відповіді, створюючи таким чином зв'язний та контекстуально обґрунтований текст.
5. Цикл зворотного зв'язку з користувачем: Користувач надає подальші підказки у відповідь на результати роботи LLM. Цей зворотний зв'язок спрямовує розмову на подальші ітерації циклу.
6. З кожним циклом контекстне вікно розширюється, що дозволяє великій мовній моделі накопичувати знання і краще розуміти цілі взаємодії користувачем.
7. Протягом багатьох циклів модель покращує механізм знаходження рішення, розкриває глибші сенс і підтримує фокус теми в взаємодії.

2.5 Формат вхідних файлів, обробка вхідних даних

Стандартним форматом для електронних листів, який зберігає електроне повідомлення як окремий файл з усіма його елементами текстом листа, його форматуванням, заголовками, а також прикріпленими файлами є EML формат. Цей формат широко підтримується багатьма поштовими клієнтами, наприклад, Google Gmail, Novell GroupWise, Microsoft Outlook Express, Lotus notes, Windows Mail, Mozilla Thunderbird, та Postbox. У випадку Google Gmail електроні листи в даному форматі можна легко зберегти за допомогою функції «Завантажити лист».

Обрана велика мовна модель OpenAI GPT-4 не вміє працювати з файлами у EML форматі. Сильна сторона цих моделей аналізувати безпосередньо текст і працювати з ним. Отже вхідні дані у вигляді EML формату потребують попередньої обробки.

Для роботи з файлами EML формату буде використано мову Python. є одна з найбільш популярних мов для обробки даних, включаючи EML файли. Python має бібліотеку email, яка дозволяє зручно обробляти електронні листи. Використання однієї мови програмування для обробки вхідних даних та взаємодії з великою мовною моделлю дозволить легко поєднати код.

Отриману інформацію електронного листа в EML форматі буде збережено в текстовому форматі .txt . Даний формат дуже поширений, підтримується величезною кількістю систем. Формат .txt є зручним для редагування та має невеликий розмір, що є важливим від час проміжних тестувань модулей системи виявлення фішингових листів.

Для розбору вхідного листа у EML форматі необхідно створити окремий модуль на котрий буде називатись «Модуль підготовки даних». Мета «Модуля підготовки даних» розібрати EML файл та дістати з нього елементи електронного листа, що були визначені у першому розділі, такими що найчастіше використовуються у фішингових листах, та можуть бути проаналізовані для ефективного виявлення та попередження фішингових атак:

- From – дані про відправника повідомлення будуть збережені у txt форматі із збереженням ім’я відправника та його поштовою адресою, із назвою «tmp_m4_from.txt», де tmp – префікс що свідчить про тимчасовість даних, m4 – модуль що використовуватиме ці дані в подальшому.
- Text body – основний текст електронного повідомлення буде збережений у вигляді тексту, попередньо очищений від зайвих пробілів, відступів та URL адрес, у файлі із назвою «tmp_m6_m7_body.txt» – префікс що свідчить про тимчасовість даних, m6 та m7 – модулі що використовуватимуть ці дані в подальшому.

- URL посилання – усі вкладені посилання будуть збережені в окремому txt файлі, із назвою «tmp_m5_urls», де tmp – префікс що свідчить про тимчасовість даних, m5 – модуль що використовуватиме ці дані в подальшому
- Attached files (Вкладені файли) – будуть збережені із збереженням оригінальних форматів для подальшої перевірки антивірусним сервісом, але зі зміненими назвами у форматі «tmp_m7_testfile№.format», де tmp – префікс що свідчить про тимчасовість даних, m7 – модуль що використовуватиме ці дані в подальшому, testfile№ порядковий номер файлу, format оригінальний формат файлу.

Висновок другого розділу

Наразі наявний широкий вибір великих мовних моделей від різних вендорів, що спеціалізуються на різних задачах, мають різні потужності та способи розгортання та використання.

Постійна підтримка, донавчання великих мовних моделей забезпечують ефективне виявлення індикаторів фішингу від нових методів, технік та шаблонів що використовуватимуться в фішингових електронних листів.

Розгортання великої мовної моделі локально потребує великої потужності, тому перевага надана великими мовним моделям що підтримують API для використання в програмному засобі виявлення фішингових листів.

Ефективність використання великих мовних моделей, зокрема GPT-4 для виявлення індикаторів фішингу значною мірою залежить від створення добре продуманих підказок.

Форматом вхідних даних обрано EML формат, який широко підтримується багатьма поштовими клієнтами, наприклад, Google Gmail,

Novell GroupWise, Microsoft Outlook Express, Lotus notes, Windows Mail. Вхідні дані потребують попередньої обробки для представлення їх у вигляді, з яким здатна взаємодіяти велика мовна модель. Попередньої обробка вхідних даних буде виконуватись модулем обробки даних.

Для забезпечення швидкості розробки, легкості тестування було обрано архітектуру модульний моноліт. Мову програмування Python версії

Розділ 3 Розробка засобу виявлення фішингових листів

У розділі 3 представлено розробку та ретельне тестування інструменту виявлення фішингової електронної пошти.

Основні функції інструменту включають розбір електронного листа на компоненти, використання підказок для налаштування великої мовної моделі для виконання задач з виявлення визначених індикаторів ризику, оцінка та обробка отриманих результатів аналізу та прийняття рішення на поставі отриманих результатів та порогового значення.

Процес розробки передбачав інтенсивну розробку релевантних підказок та тестування з використанням різноманітного набору даних, що включає як законні, так і фішингові електронні листи. Це забезпечило високу точність інструменту та мінімізувало помилкові спрацьовування.

3.1 Проектування засобу виявлення фішингу

Для забезпечення швидкості розробки, легкості тестування було обрано архітектуру модульний моноліт. Основною ідеєю є побудова системи як єдиного застосунку (моноліту), але при цьому з чітким розподілом на модулі, кожен з яких має свою власну відповідальність.

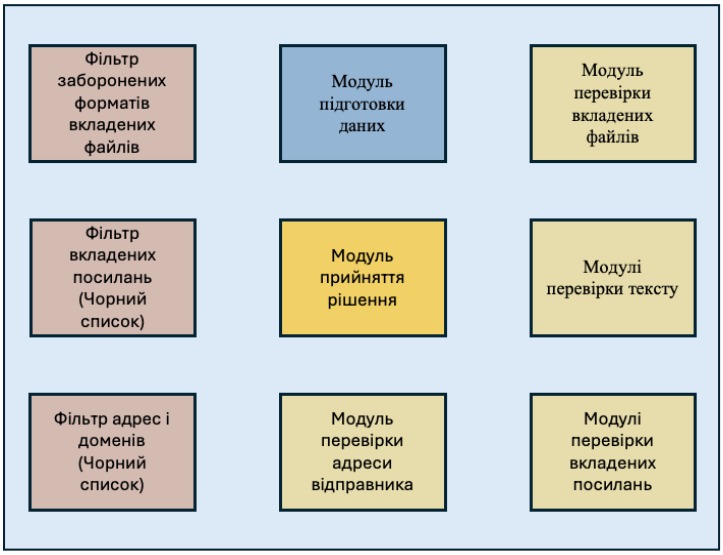


Рисунок 5. Діаграма архітектури програмного рішення

Для розширення функціоналу системи та оптимізації ресурсів для визначення індикаторів фішингу, слід додати первісну перевірку адресів електронної пошти відправника електронного листа, вкладених URL посилань, та заборонених форматів вкладених файлів.

Це дозволить визначити критерії адресів та доменів, які є забороненими, і автоматично визначені які фішингові. Це можуть бути домени одноразових анонімних поштових сервісів, наприклад Yopmail, та небажані поштових сервіси наприклад, “mail.ru” та домени верхнього рівня, наприклад, « .рф».

Для заборонених форматів вкладених файлів це будуть виконувані файли та файли з подвійним форматом.

Запропонована діаграма варіантів використання (англійською Use case diagram) програмного засобу виявлення фішингу.

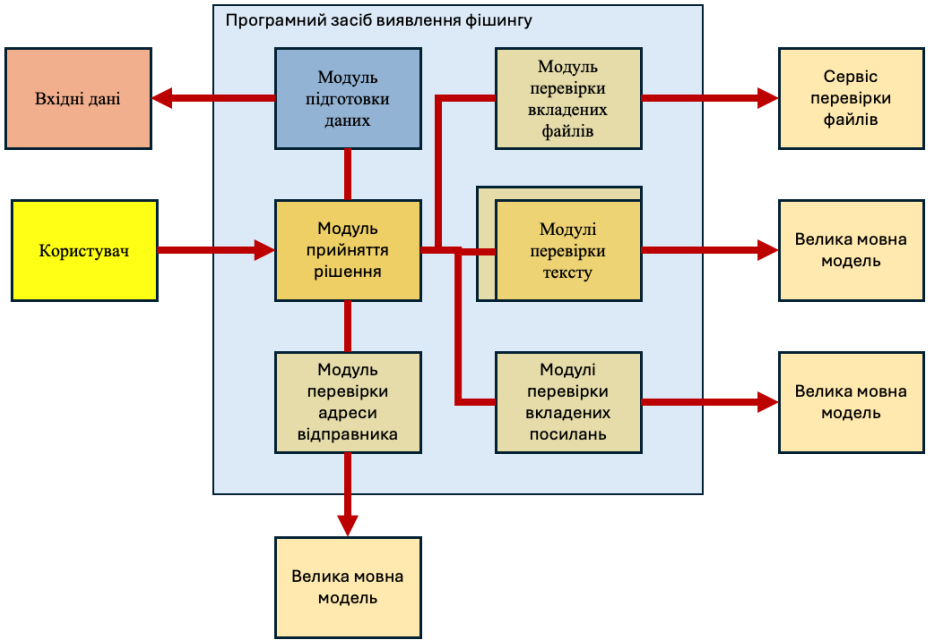


Рисунок 6. Діаграма прецедентів засобу виявлення фішингових листів

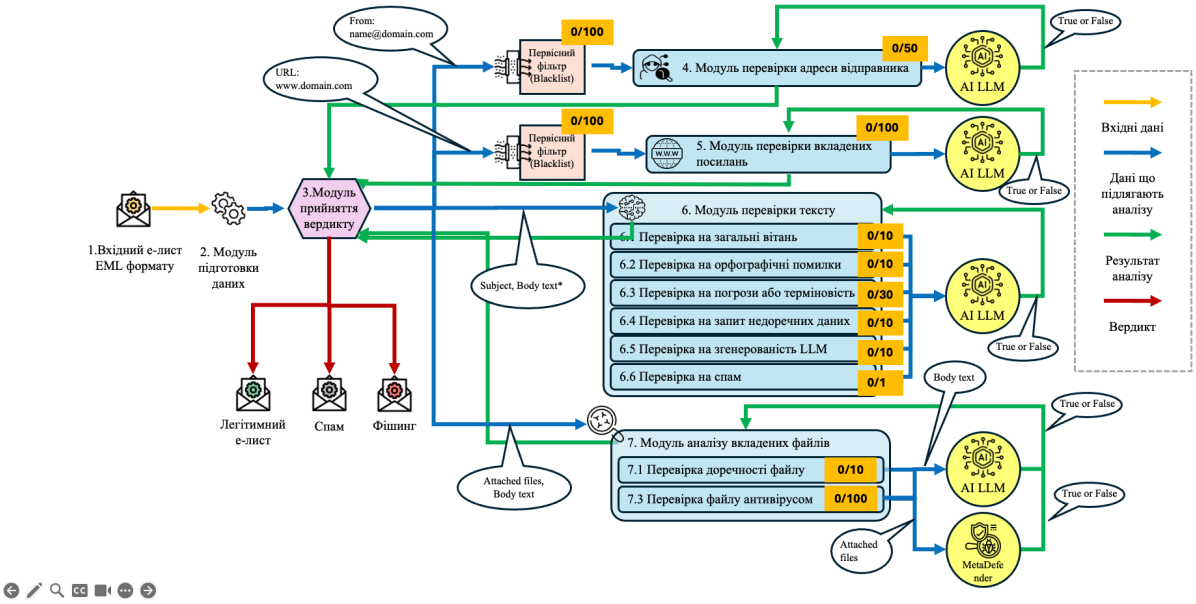


Рисунок 7. Загальна діаграма потоків даних

3.2 Система оцінювання результатів аналізу

Система оцінювання результатів перевірки елементів електронного листа на наявність індикаторів фішингу базується на бальній системі, де кожний модуль, відповідальний за визначення індикаторів фішингу має певну вагу або кількість балів. Цей метод можна характеризувати як ваговий бальній метод оцінювання ризиків. Кожен модуль фішингу має призначену вагу або кількість балів. Ці бали призначаються на основі результатів оцінки елементів електронного листа LLM моделлю. LLM модель повертає бінарний результат оцінки (true/false).

Система використовує порогове значення в 100 балів, щоб визначити, чи є лист фішинговим. Якщо загальний бал за всіма індикаторами досягає або перевищує це порогове значення, лист класифікується як фішинг.

Модуль	**Можлива оцінка (false/true)**
Первісний фільтр поштових адрес	0 / 100
Модуль перевірки адреси відправника	0 / 50

Первісний фільтр вкладених URL посилань	0 / 100
Модуль перевірки на загальні вітань	0 / 10
Модуль перевірки на орфографічні помилки	0 / 10
Модуль перевірки на погрози або терміновість	0 / 30
Модуль перевірки на запит недоречних даних	0 / 10
Модуль перевірки на згенерованість LLM	0 / 10
Модуль перевірки на спам	0 / 1
Модуль перевірки на доречність файлу	0 / 10
Перевірка вкладеного файлу антивірусом	0 / 100

Таблиця 2. Вагові значення оцінювання ризиків

3.3 Модуль розбору електронних листів

Модуль підготовки даних (m2_1_parcer_email.py) – здійснює розбір (парсинг) EML файлу електронної пошти, вилучає з нього основні дані (адресу відправника, тему листа, текст, URL-посилання, вкладені файли) і виконує деякі дії з для підготовки цих даних до аналізу.

Використані бібліотеки:

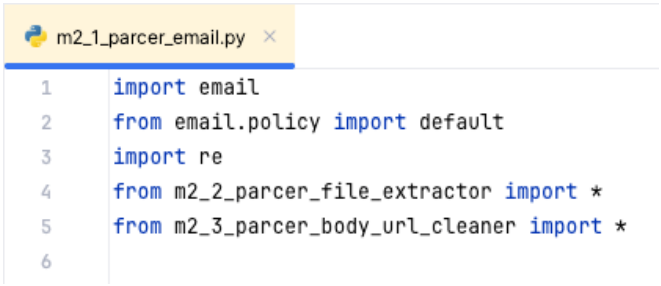


Рисунок 8. Бібліотеки модуля підготовки даних

- email – бібліотека для робити з поштовими повідомленнями. Використовується для читання EML файлу: *email.message_from_file(file, policy=default)*

- re – бібліотека для роботи з регулярними виразами. Використовується для пошуку URL у тексті повідомлення: re.findall('http[s]?://...', body)
- from email.policy import default - є частиною стандартного модуля email у Python. Використовується у функції email.message_from_file для встановлення політики: email.message_from_file(file, policy=default). Аргумент policy=default визначає політику обробки повідомлень. Політика default є рекомендованою для більшості випадків, вона забезпечує коректне декодування і обробку різних частин повідомлення.
- from m2_2_parcer_file_extractor import * - власноруч написаний підмодуль для розширення функціоналу Модуля підготовки даних(m2_1_parcer_email.py). Використовується для вивантаження вкладених файлів (Attached files) з EML файлу, та збереження їх для подальшої перевірки. Виконано окремим підмодулем з метою забезпечення масштабованості та сегментації функціоналу материнського модуля.
- from m2_3_parcer_body_url_cleaner import * - власноруч написаний підмодуль-розширення функціоналу Модуля підготовки даних (m2_1_parcer_email.py). Використовується для очищення тексту електронного листа від зайвих пробілів та рядків, що дозволяє оптимізувати подальшому обробку.

Модуль підготовки даних має блок *if __name__ == "__main__":* необхідний для самостійного виконання модуля, що полегшує тестування та розробку.

Основна функція модуля підготовки даних def parcer_email(file_path): складається з 3 логічних частин:

1. Вилучення даних з елементів електронного листа представленого у вигляді EML файлу

m2_1_parcer_email.py

```python

def parcer_email(file_path):
    # Читаємо EML файл
    print("Status: Підготовка EML файлу до розбору ...")
    with open(file_path, 'r', encoding='utf-8') as file:
        msg = email.message_from_file(file, policy=default)

    # Вилучаємо інформацію
    print("Status: Вилучення данних з EML файлу ...")
    from_address = msg.get('From')
    subject = msg.get('Subject')
    body = ""
    urls = []

    if msg.is_multipart():
        for part in msg.walk():
            content_type = part.get_content_type()
            content_disposition = part.get( name: 'Content-Disposition', failobj: '')  # Встановлюємо порожній рядок як значення за замовчуванням
            if content_type == 'text/plain' and 'attachment' not in content_disposition:
                body += part.get_payload(decode=True).decode()
    else:
        body = msg.get_payload(decode=True).decode()

    # Знаходимо усі URL
    urls = re.findall( pattern: 'http[s]?://(?:[a-zA-Z]|[0-9]|[$-_@.&+]|[!*\\(\\),]|(?:%[0-9a-fA-F][0-9a-fA-F]))+', body)

```

Рисунок 9. Вилучення елементів електронного листа з EML файлу

2. Збереження вилучених даних у .txt форматі із визначеними таксономією назвами.

```python

    # Зберігаємо результати у файлах
    #base_path = os.path.splitext(file_path)[0]
    with open(f"tmp_m4_from.txt", 'w', encoding='utf-8') as file: #Якщо потрібен шлях до файлу f"{base_path}_from.txt"
        file.write(from_address if from_address else "")
    print("     Дані відправника файлу вилучені та готові до аналізу")

    with open(f"tmp_m6_subject.txt", 'w', encoding='utf-8') as file: #Якщо потрібен шлях до файлу f"{base_path}_subject.txt"
        file.write(subject if subject else "")
    print("     Тема електронного листа вилучена та готова до аналізу")

    with open(f"tmp_m6_m7_body.txt", 'w', encoding='utf-8') as file: #Якщо потрібен шлях до файлу f"{base_path}_body.txt"
        file.write(body if body else "")
    print("     Основний текст електронного листа вилучений")

```

Рисунок 10. Збереження вилучених елементів електронного листа

Кожен текстовий елемент електронного листа зберігається окремо, із відповідною назвою файлу у текстовому форматі .txt .

3. Вилучення та збереження вкладених.

```python

def save_attachments_from_eml(eml_path):
    print("Status: Вилучення вкладених файлів ...")
    # Відкриваємо EML файл
    with open(eml_path, 'rb') as file:
        # Парсимо EML файл
        msg = BytesParser(policy=policy.default).parse(file)

    # Лічильник для іменування файлів
    attachment_number = 1

    # Ітеруємо по частинах повідомлення
    for part in msg.walk():
        # Якщо частина є вкладеним файлом, зберігаємо його
        if part.get_content_maintype() == 'multipart':
            continue
        if part.get('Content-Disposition') is None:
            continue

        # Отримуємо ім'я файла та його розширення
        filename = part.get_filename()
        if not filename:
            file_extension = ''
        else:
            file_extension = os.path.splitext(filename)[1]

        # Нове ім'я файла із збереженням розширення
        new_filename = f"tmp_m7_testfile{attachment_number}{file_extension}"
        save_path = os.path.join(os.getcwd(), new_filename)

        # Зберігаємо файл
        with open(save_path, 'wb') as save_file:
            save_file.write(part.get_payload(decode=True))
            print(f"    Saved attachment to {save_path}")
        attachment_number += 1
    print("Status: Вкладені файли вилучені та готові до аналізу")
```

Рисунок 11. Вилучення та збереження вкладених

Після збереження даних у викликається підмодуль m2_3_parcer_body_url_cleaner очистки вилученого елементу Text body від зайвих пробілів, відступів та URL адрес, що потрібно подальшого для якісного аналізу великою мовною моделлю моделі.

m2_3_parcer_body_url_cleaner.py

```python
import re

def remove_urls_and_flatten_text(input_file_path, output_file_path):
    # Регулярний вираз для пошуку URL
    url_pattern = r'https?://\S+|www\.\S+'

    try:
        # Відкриття вхідного файлу для читання
        with open(input_file_path, 'r', encoding='utf-8') as file:
            content = file.read()

        # Видалення URL
        content_without_urls = re.sub(url_pattern, repl: '', content)

        # Видалення переносів рядків і зайвих пробілів
        single_line_content = ' '.join(content_without_urls.split())

        # Запис обробленого тексту в вихідний файл
        with open(output_file_path, 'w', encoding='utf-8') as file:
            file.write(single_line_content)

        print("    Основний текст електронного листа оброблений і готовий до аналізу")
    except FileNotFoundError:
        print("    Помилка: вказаний файл не знайдено.")
    except Exception as e:
        print(f"    Сталася помилка: {e}")
```

Рисунок 12. Очистка елементу основного тексту електронного повідомлення

Тестовий електронний лист містить основний текст, в якому відбувається відправка файлів, містить 4 файли. Формати вкладених файлів: excel, pfd, png, .exe), 1 з яких kitty.exe є шкідливим програмним забезпеченням та URL посилання.

```
/Users/denyskysil/PycharmProjects/pythonProject1/.venv/bin/python /Users/denyskysil/PycharmProjects/pythonProject1/.ve
Status: Підготовка EML файлу до розбору ...
Status: Вилучення данних з EML файлу ...
     Дані відправника файлу вилучені та готові до аналізу
     Тема електронного листа вилучена та готова до аналізу
     Основний текст електронного листа вилучений
     Основний текст електронного листа оброблений і готовий до аналізу
     Вкладені посилання вилучені та готові до аналізу
Status: Вилучення вкладених файлів ...
    Saved attachment to /Users/denyskysil/PycharmProjects/pythonProject1/.venv/scr/tmp_m7_testfile1.pdf
    Saved attachment to /Users/denyskysil/PycharmProjects/pythonProject1/.venv/scr/tmp_m7_testfile2.png
    Saved attachment to /Users/denyskysil/PycharmProjects/pythonProject1/.venv/scr/tmp_m7_testfile3.xlsx
    Saved attachment to /Users/denyskysil/PycharmProjects/pythonProject1/.venv/scr/tmp_m7_testfile4.exe
Status: Вкладені файли вилучені та готові до аналізу
Status: EML файл розібрано. Перевірка електронного листа на фішинг ...

Process finished with exit code 0
```

Рисунок 13. Робота модуля обробки вхідних даних

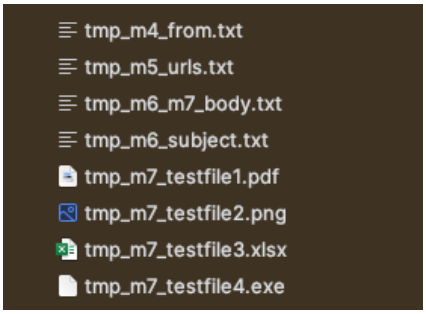


Рисунок 14. Результат роботи модуля обробки вхідних даних

3.4 Первісний фільтр доменів адресів електронної пошти

Фільтр доменів (m4_0_filter_address.py) адреси електронної пошти відправника відіграє важливу роль у забезпеченні ефективності та точності системи. Визначення переліку заборонених доменів дозволяє збільшити швидкість відповіді системи та зменшує витрати на обробку, оскільки можна уникнути перевірки великою мовною моделлю.

Для ведення чорного списку доменів адресів електронної пошти використано файл формату CSV, що є стандартом для подібних задач.

Використані бібліотеки:

- re – бібліотека для роботи з регулярними виразами. Використовується для пошуку домену у чорному списку: match = re.search(r'<(.+?)@(.+?)>', email)
- csv – бібліотека для роботи з файлами у форматі CSV (Comma-Separated Values).

m4_0_filter_address.py

```python
# Функція для завантаження чорного списку доменів з CSV файлу
def load_blacklist(filename):
    with open(filename, newline='') as csvfile:
        reader = csv.reader(csvfile)
        blacklist = [row[0] for row in reader]
    return blacklist

# Функція для витягування домену з поштової адреси
def email_domain(email):
    match = re.search( pattern: r'<(.+?)@(.+?)>', email)
    if match:
        return match.group(2)
    return None

# Функція для перевірки, чи домен знаходиться в чорному списку
def is_blacklisted(domain, blacklist):
    for pattern in blacklist:
        if pattern.startswith('*.'):  # Перевірка для шаблонів типу *.ru
            if domain.endswith(pattern[1:]):
                return True
        else:  # Перевірка для точних відповідностей домену
            if domain == pattern:
                return True
    return False

# Функція для перевірки, чи заблокований домен поштової адреси
def check_email(email, blacklist):
    domain = email_domain(email.strip())
    return domain is not None and is_blacklisted(domain, blacklist)
```

Рисунок 15. Вилучення елементів електронного листа з EML файлу

3.3 Модулі взаємодії з великою мовною моделлю

Архітектура модулів взаємодії з LLM моделлю є подібною для всі модулей взаємодії з великою мовною моделлю. Для обміном інформації використовується API GPT-4. Розглянемо на прикладі модуля перевірки адреси відправника повідомлення. Використані бібліотеки в модулі:

m4_1_verificator_email_address.py

```python
import openai
import time
from config import openai_api_key
```

Рисунок 16. Бібліотеки модуля взаємодії з великою мовною моделлю

- openai - використовується для взаємодії з API OpenAI, зокрема для отримання відповідей від моделі.
- time - бібліотека що використовується для управління часовими інтервалами у програмі, для створення затримки між спробами повторного відправлення запиту в разі неуспішної спроби
- config - модуль використовується для збереження та імпорту ключів API

Код модуля взаємодії з моделлю GPT-4 має 2 основні функції:

```
39
40  def verificator_email_address():
41      # Відкриття файлу з текстом
42      with open('tmp_m4_from.txt', 'r') as file:
43          # Читання тексту з файлу
44          text = file.read()
45
46      # Присвоєння зчитаного тексту змінній email_text
47      email_text = text
48
49      # Викликаємо функцію та повертаємо результат
50      testresult_address_check = verification_email_address(email_text)
51
52      return testresult_address_check
53
```

Рисунок 17. Функція def verificator_email_address()

- def verificator_email_address() - функція вищого рівня і використовується для організації процесу перевірки електронної адреси. Безпосередньо ця функція викликається модулем прийняття рішення, за умови, що домен відправника електронного повідомлення не є в чорному списку. Данна функція виконує такі дії
  - Відкриває файл що містить адресу відправника повідомлення що перевіряється, зчитує з адресу електронної пошти у вигляді тексту
  - Викликає наступну функцію def verification_email_address(email_text) та передає їй зміну що містить елемент електронного листа, основний текст.

```python
def verification_email_address(email_text):
    openai.api_key = openai_api_key

    messages = [
        {"role": "system", "content": {promt_email_address}},
        {"role": "user", "content": f"Analyze the following email address and determine if it fishing address "
                                    f"or not. The answer should be 'true' if it is fishing address, 'false' if it is not:\n\n{email_text}"}
    ]

    attempt = 0
    response = None

    while response is None and attempt < 5:  # Спробуємо до 5 разів
        try:
            response = openai.ChatCompletion.create(
                model="gpt-4o",
                messages=messages,
                max_tokens=256
            )
            result = response.choices[0].message['content'].strip().lower()
            if result == 'true':
                return True
            elif result == 'false':
                return False
            else:
                raise ValueError("Unexpected response")

        except Exception as e:
            print(f"Attempt {attempt + 1} failed: {e}")
            attempt += 1
            time.sleep(1)  # Чекаємо секунду перед наступною спробою

    if response is None:
        return "Failed to get a response after several attempts."

```

Рисунок 18. Основна функція взаємодії з великою мовною моделлю

- def verification_email_address(email_text) – функція взаємодії з LLM моделлю приймає текст електронної адреси як вхідний параметр і виконує
    - Налаштовує API ключ для взаємодії з великими мовними моделями OpenAI
    - Визначає велику мовну модель до якої звертатиметься, тип та розмір запиту
    - Створює запит до моделі з використанням тексту електронної адреси та підказкою
    - Обробляє відповідь моделі та повертає True, якщо адреса визнана фішинговою, і False, якщо адреса безпечна. Якщо модель повертає невизначену відповідь або виникає помилка, код намагається повторити запит до п'яти разів з інтервалом в одну секунду.

3.5 Перевірка вкладених файлів

Фішингові атаки часто використовують вкладені файли, що містять шкідливі програми або посилання, які можуть обійти прості текстові фільтри. Сканер вкладених файлів здатен аналізувати вміст таких файлів на наявність підозрілих ознак і шкідливого коду, тим самим значно знижуючи ризик успішного фішингу.

Для сканування вкладених файлів буде використана безкоштовна платформа виявлення шкідливого програмного забезпечення MetaDefender.

MetaDefender має низку переваг, які роблять його ефективним інструментом для забезпечення безпеки даних і захисту від кіберзагроз:

- Використання до 30 антивірусних рушіїв для виявлення шкідливого ПЗ, що значно підвищує точність і швидкість виявлення загроз.
- Легка інтеграція з іншими системами безпеки та програмними рішеннями, такими як шлюзи електронної пошти, мережеві екрани та SIEM-системи.
- Часткова безкоштовність використання інтерфейсу API MetaDefender.

Використані бібліотеки:

```
1  import requests
2  import time
3  import os
4  import re
5  from config import metadefender_api_key
```

Рисунок 19. Бібліотеки модуля перевірки вкладених файлів

- re – Використовується для роботи з регулярними виразами для фільтрації вкладених файлів за шаблоном
- requests - використовується для надсилання HTTP-запитів до API MetaDefender

- time – використовується для паузи між запитами під час перевірки статусу сканування файлу.
- os - використовується для роботи з файловою системою, зокрема для переліку файлів у директорії.
- config - використовується для імпорту metadefender_api_key.

Перед виконанням передвіки кладених файлів за допомогою сервісу API MetaDefender, відбувається перевірка на індикатор фішингу «Доречність вкладеного файлу» модулем взаємодії з великою мовною моделлю OpenAI GPT-4 «m7_1_verificator_file_relevance».

Після цього виконується перевірка на шкідливе програмне забезпечення вкладених файлів.

Модуль перевірки вкладених файлів складається з 3х важливих функцій:

1. Функція send_file_for_scanning(api_key, file_path) – приймає API-ключ та шлях до файлу, відкриває файл для читання в бінарному режимі і надсилає POST-запит до API MetaDefender з цим файлом. Вона отримує відповідь у форматі JSON та повертає data_id, який використовується для подальшої перевірки статусу сканування

```
6
7   # Надсилання файлу для сканування
8   def send_file_for_scanning(api_key, file_path):
9       url = 'https://api.metadefender.com/v4/file'
10      headers = {'apikey': api_key}
11      with open(file_path, 'rb') as file:
12          response = requests.post(url, headers=headers, files={'file': file})
13          json_response = response.json()
14          return json_response['data_id']
15
```

Рисунок 20. Відправка вкладених файлів на перевірку сканером

2. Функція def check_scan_status(api_key, data_id) – функція приймає API-ключ та data_id файлу, що був надісланий на сканування. Вона надсилає GET-запити до API MetaDefender для отримання статусу сканування, очікуючи на завершення процесу сканування. Якщо прогрес сканування досягає 100%, функція повертає результат сканування (Allowed або Blocked). У разі, якщо сканування триває,

вона робить паузу на 60 секунд перед наступним запитом (60 – є обмеженням безкоштовного плану сервісу).

```python
# Перевірка статусу сканування файлу
def check_scan_status(api_key, data_id):
    url = f'https://api.metadefender.com/v4/file/{data_id}'
    headers = {'apikey': api_key}
    while True:
        response = requests.get(url, headers=headers)
        json_response = response.json()
        progress_percentage = json_response.get('process_info', {}).get('progress_percentage', 0)
        if progress_percentage == 100:
            return json_response['process_info']['result']
        else:
            print(f"    Scan in progress... {progress_percentage}% completed.")
            time.sleep(60)  # Чекати 60 секунд перед наступним запитом (обеження ресурсу)

```

Рисунок 21. Перевірка статусу сканування файлів

3. def verificator_antivirus_metadefender() – основна функція модуля, визначає директорію для сканування та шаблон файлів, які потрібно сканувати за допомогою регулярного виразу. Вона проходить по всіх файлах у заданій директорії, відфільтровує файли за шаблоном і запускає процес сканування для кожного відповідного файлу, використовуючи функції send_file_for_scanning та check_scan_status. Після завершення сканування всіх файлів вона аналізує результати сканування і повертає True, якщо хоча б один файл був заблокований, і False, якщо всі файли дозволені. Якщо є файли з невизначеними результатами, функція повідомляє про необхідність оновлення коду.

```python

# Основний процес скрипту
def verificator_antivirus_metadefender():
    # Перевірка всіх файлів у директорії
    api_key = metadefender_api_key
    directory_path = '.'  # Змініть на шлях до вашої директорії, якщо потрібно
    file_pattern = re.compile(r'tmp_m7_testfile\d+')  # Регулярний вираз для ідентифікації файлів

    results = []

    for filename in os.listdir(directory_path):
        if file_pattern.match(filename):
            file_path = os.path.join(directory_path, filename)
            print(f"    Scanning {filename}...")
            data_id = send_file_for_scanning(api_key, file_path)
            result = check_scan_status(api_key, data_id)
            print(f"    Scan result for {filename}: {result}")
            results.append(result)

        # Перевірка результатів після сканування всіх файлів
    if any(result == "Blocked" for result in results):
        return True
    if all(result == "Allowed" for result in results):
        return False
        # Якщо є файли з іншими результатами, теж повертаємо True
    return print('Невідомий результат. Необхідно оновити код')
```

Рисунок 22. Основний скрипт модуля перевірки вкладених файлів

3.6 Модуль прийняття рішення

Модуль прийняття рішення (m3_1_phishing_detector.py) – є ключовою частиною системи виявлення фішингу, що ініціює поступове виконання усіх інших модулів, збирає результати тестування та оцінює їх, виносить вердикт.

У Модуль прийняття рішення було імпортовано усі залежні модулі, що викликаються поступово.

Використані бібліотеки:

m3_1_phishing_detector.py

```
from m2_1_parcer_email import *
from m4_0_filter_address import *
from m4_1_verificator_email_address import *
from m5_1_verificator_url_address import *
from m6_1_verificator_general_greeting import *
from m6_2_verificator_spelling_mistakes import *
from m6_3_verificator_threat_urgency import *
from m6_4_verificator_inappropriate_request import *
from m6_5_verificator_ai_generation import *
from m6_6_verificator_spam import *
from m7_1_verificator_file_relevance import *
from m7_3_verificator_antivirus_metadefender import *

```

Рисунок 23. Бібліотеки модуля прийняття рішення

Частина коду що відповідає за виклик модуля перевірки адреси електронної пошти відправника. Та очікує результатів перевірки у вигляді бінарної змінної true або false. Залежно від результату, код визначає кількість балів, або викликає модуль взаємодії з великою мовною моделлю.

```
    #Перевірка чорного списку m4_0_filter_address
    domain_blacklisted = filter_address(sender_address_file, blacklist_domains_file)
    print(f'Status: Результат перевірки чорного списку TLD: {domain_blacklisted}')
    if domain_blacklisted:
        print("!!!Warning!!!: Домен відправника заблокований адміністратором!")

    else:
        print("Status: Перевірка адреси відправника LLM моделлю ...")
        testresult_address_check = verificator_email_address()
        print(f"Status: Результат перевірки адреси відправника: {testresult_address_check}")
        if testresult_address_check:
            print("!!!Warning!!!: Адреса відправника повідомлення є фішинговою")
            testresult_address_check = 50
        else:
            print(f"Status: Адреса відправника повідомлення є легитимною")
            testresult_address_check = 0
```

Рисунок 24. Перевірка чорного списку адрес та взаємодія з великою мовною моделлю

Зібравши результати виконання всі модулі, та надавши бали згідно з таблицею вагових значень оцінювання ризиків.

```
# Перевірка терміновість або погрози m6_4_verificator_inappropriate_request
print("Status: Перевірка тексту на запит недоречної інформації LLM моделлю...")
testresult_inappropriate_request_check = verificator_inappropriate_request()
print(f"Status: Результат превірки тексту на запит недоречної інформації: {testresult_inappropriate_request_check}")
if testresult_inappropriate_request_check:
    print("!!!Warning!!!: Повідомлення містить запит недоречної інформації")
    testresult_inappropriate_request_check = 50
else:
    print(f"Status: Повідомлення не містить запитів недоречної інформації")
    testresult_inappropriate_request_check = 0
```

Рисунок 25. Призначення вагового значення результату аналізу великої мовної моделі

Частина коду що відповідає за сумування вагових значень та прийняття вердикту.

```
160
161     # Система прийняття рішення
162     print('Status: Обробка отриманих результатів')
163
164     gen_result = testresult_address_check + testresult_urls_check + testresult_general_greeting_check +
165
166     print(f"Status: Загальна кількість набраних балів = {gen_result}")
167
168     if gen_result >= 100:
169         print('Status: Результат перевірки - Перевіряємий лист є фішинговим')
170         gen_result = True
171     elif gen_result < 100:
172         print('Status: Результат перевірки - Перевіряємий лист не є фішинговим')
173         gen_result = False
174
175     print(f"Вердикт: {gen_result}")
176
177
```

Рисунок 26. Прийняття вердикту

Висновки до третього розділу

У розділі 3 детально описано розробку інструменту виявлення фішингових листів, що складається з 6 важливих модулів та первинних фільтрів.

Під час проектування програмного засобу виявлення фішингових листів було створено діаграму архітектури програмного рішення, запропоновано діаграму варіантів використання та загальна діаграма потоків даних з вказаними елементам електронного листа що аналізуються та ваговими коефіцієнтами що можуть бути призначеними кожному модулю перевірки.

Також було визначено вагові коефіцієнти ризику для кожного модуля виявлення індикаторів ризику.

Система оцінювання результатів перевірки елементів електронного листа на наявність індикаторів фішингу базується на бальній системі, де кожний модуль, відповідальний за визначення індикаторів фішингу має певну вагу або кількість балів.

Розділ 4 Тестування розробленого засобу виявлення фішингових листів

У цьому розділі детально описано процес тестування розробленого програмного засобу для виявлення фішингових листів. Метою тестування було оцінити ефективність та надійність засобу у виявленні фішингових атак за допомогою великих мовних моделей.

4.1 Тестова популяція

Для тестування було зібрано популяцію електронних листів у розмірі 20 серед якої було представлені легітимні, фішингові, та штучні фішингові електроні листи в EML форматах.

Популяція складалась з 40% легітимних, 40% фішингових та 20% з штучних фішингових листів.

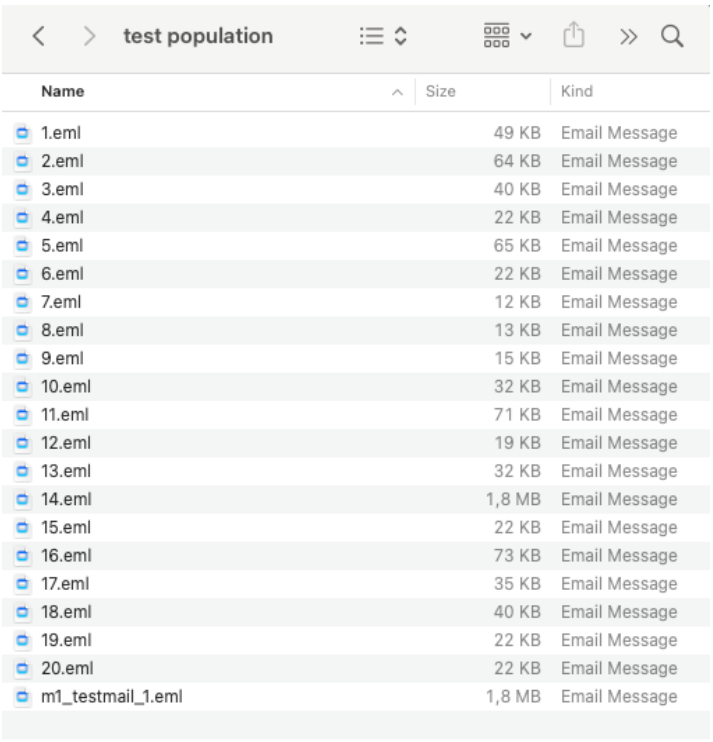


Рисунок 27. Тестова популяція

4.2 Тестування

Демонстрація роботи програмного засобу із виявленням основних індикаторів фішингу в обраних для перевірки елементах електронного листа.

Рисунок 28. Перевірка фішингового листа

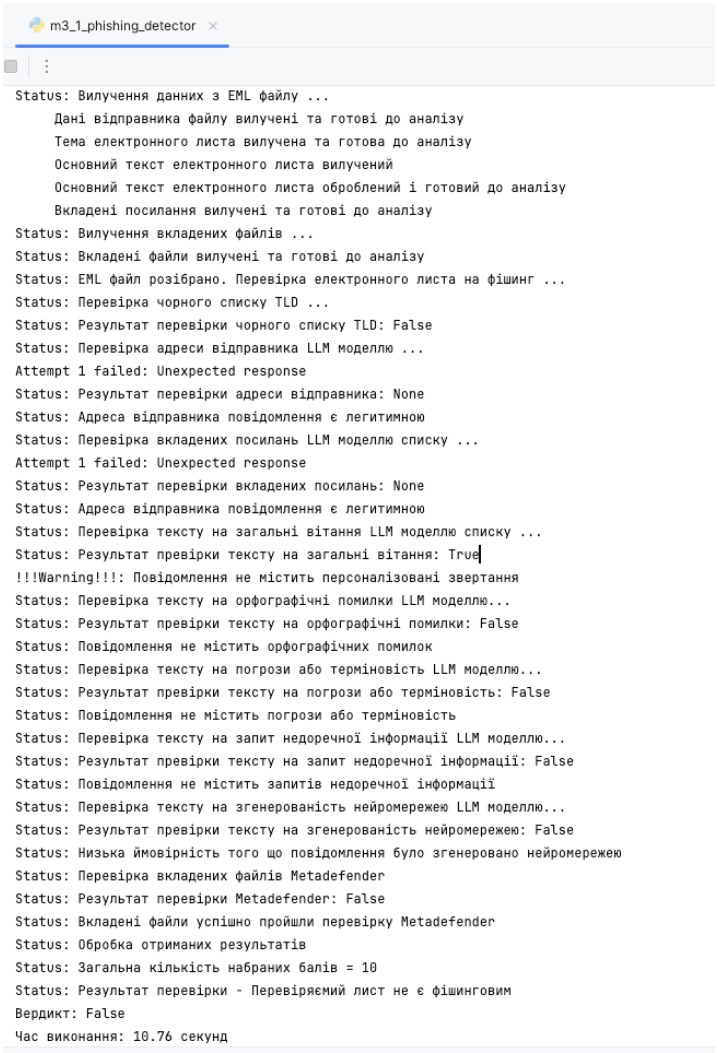


Рисунок 29. Перевірка легітимного електронного листа

4.3 Результати тестування

Тестова популяція складалась 50 листів, серед з яких 27 природних фішингових листів, 13 синтетичних фішингових листів та 10 легітимних листів.

Загальна ефективність програмного засобу, а саме правильна оцінка електронних листів склала 92%. З 50 сканувань тестових електронних листів для 46 листів було правильно визначено оцінку.

Успішність виявлення фішингових листів (природних та синтетичних) складає 97,5%, та було виявлено 39 фішингових листів з 40 протестованих.

Модель оцінила фішинговий лист як легітимний через те що він не мав жодних URL посилань, чи вкладених файлів, а також був відправлений зі скомпрометованої легітимної адреси учбового закладу. Цей лист включав

лише текст, який закликав відповісти на нього, для обговорення умов отримання «виграшу».

Мета даної техніки фішингу розрахована для того щоб обійти спам фільтр організації. Проте, у разі продовження спілкування із зловмисником, та подальшим отриманням від нього фішингових URL посилань або вкладених файлів, з високою ймовірністю буде лист буде позначено як фішинговий.

Під час перевірки легітимних листів, 3 легітимних листів було позначено неправильно. Один з випадків коли легітимний лист був невірно оцінено за критерієм «фішингова URL адреса», виявився рекламною розсилкою популярної мережі магазинів техніки, що використовував скорочені посилання що ведуть на соціальні мережі компанії. Використаний сервіс скорочення адрес є досить популярним серед зловмисників.

В інших випадках легітимні листи були позначені як фішингові, оскільки вони були мали посилання на маловідомі легітимні сторінки.

Для підвищення точності у оцінки вхідних листів слід додати фільтр білих посилань, які за замовченням є легітимними.

	A	B	C	D	E	F	G	H	I	J	K	L
1	test id	Date	Type	Class	result	Attached files	Efficiency	time, секунди	Опис			
2	1	04.05.2024	Nature	FALSE	FALSE	no	Effective	4,07				
3	2	04.05.2024	Nature	FALSE	FALSE	no	Effective	10,76				
4	3	04.05.2024	Nature	FALSE	FALSE	no	Effective	5,44				
5	4	04.05.2024	Nature	FALSE	FALSE	no	Effective	5,47				
6	5	04.05.2024	Nature	TRUE	TRUE	yes	Effective	56,17	під час розбору отриманий фантомний файл			
7	6	04.05.2024	Sintetic	TRUE	TRUE	no	Effective	4,56	Штучно встроєні елементи листа			
8	7	04.05.2024	Nature	TRUE	FALSE	no	Ineffective	4,67	Фішинговий лист, без посила або вкладень			
9	8	04.05.2024	Nature	TRUE	TRUE	yes	Effective	66,37	під час розбору отриманий фантомний файл			
10	9	04.05.2024	Nature	TRUE	TRUE	no	Effective	4,99				
11	10	04.05.2024	Nature	TRUE	TRUE	no	Effective	4,09				
12	11	04.05.2024	Nature	TRUE	TRUE	no	Effective	5,92				
13	12	04.05.2024	Nature	TRUE	TRUE	no	Effective	5,16				
14	13	04.05.2024	Nature	TRUE	TRUE	no	Effective	6,62				
15	14	04.05.2024	Sintetic	TRUE	TRUE	yes	Effective	254,45	4 вкладених файлів			
16	15	04.05.2024	Sintetic	TRUE	TRUE	no	Effective	5,07	Штучно встроєні елементи листа			
17	16	04.05.2024	Nature	FALSE	TRUE	no	Ineffective	5,05	Посилання Allo позначено як фішинг			
18	17	04.05.2024	Nature	FALSE	FALSE	no	Effective	5,36				
19	18	04.05.2024	Nature	FALSE	FALSE	no	Effective	5,4				
20	19	04.05.2024	Nature	FALSE	FALSE	no	Effective	6,58				
21	20	07.05.2024	Sintetic	TRUE	TRUE	no	Effective	4,59				
22	21	07.05.2024	Sintetic	TRUE	TRUE	yes	Effective	4,34				
23	22	07.05.2024	Nature	TRUE	TRUE	no	Effective	4,57				
24	23	07.05.2024	Nature	TRUE	TRUE	no	Effective	4,5				
25	24	07.05.2024	Nature	TRUE	TRUE	yes	Effective	5,02				
26	25	07.05.2024	Nature	FALSE	TRUE	no	Ineffective	4,33	Посилання Steam позначено як фішинг			
27	26	07.05.2024	Nature	TRUE	TRUE	no	Effective	4,27				
28	27	07.05.2024	Nature	TRUE	TRUE	no	Effective	5,12				
29	28	07.05.2024	Nature	TRUE	TRUE	no	Effective	5,02				
30	29	07.05.2024	Sintetic	TRUE	TRUE	no	Effective	5,28				
31	30	07.05.2024	Sintetic	TRUE	TRUE	yes	Effective	5,46				
32	31	07.05.2024	Sintetic	TRUE	TRUE	no	Effective	6,52				
33	32	07.05.2024	Sintetic	TRUE	TRUE	no	Effective	4,3				
34	33	07.05.2024	Sintetic	TRUE	TRUE	no	Effective	4,13				
35	34	07.05.2024	Nature	FALSE	TRUE	no	Ineffective	4,17	Посилання Yasno позначено як фішинг			
36	35	08.05.2024	Nature	TRUE	TRUE	no	Effective	4,2				
37	36	08.05.2024	Nature	TRUE	TRUE	yes	Effective	5,22				
38	37	08.05.2024	Nature	TRUE	TRUE	no	Effective	4,31				
39	38	08.05.2024	Nature	TRUE	TRUE	no	Effective	4,21				
40	39	08.05.2024	Nature	TRUE	TRUE	yes	Effective	5,03				
41	40	08.05.2024	Nature	TRUE	TRUE	no	Effective	5,01				
42	41	08.05.2024	Sintetic	TRUE	TRUE	no	Effective	5,4				
43	42	08.05.2024	Sintetic	TRUE	TRUE	no	Effective	5,12				
44	43	08.05.2024	Sintetic	TRUE	TRUE	no	Effective	6,13				
45	44	08.05.2024	Nature	TRUE	TRUE	no	Effective	4,31				
46	45	08.05.2024	Nature	TRUE	TRUE	yes	Effective	4,13				
47	46	08.05.2024	Nature	TRUE	TRUE	no	Effective	4,24				
48	47	08.05.2024	Nature	TRUE	TRUE	no	Effective	4,28				
49	48	08.05.2024	Nature	TRUE	TRUE	no	Effective	5,21				
50	49	08.05.2024	Nature	TRUE	TRUE	no	Effective	4,16				
51	50	08.05.2024	Nature	TRUE	TRUE	no	Effective	4,11				

Рисунок 30. Результати тестування системи

Висновок четвертого розділу

Результати тестування показали високу ефективність розробленого засобу у виявленні фішингових листів, що підтверджується високими значеннями метрик точності та продуктивності. Усі тести були проведені на різноманітних наборах даних, включаючи як реальні, так і штучно створені фішингові атаки, що забезпечило всебічну перевірку можливостей системи

Інтеграція великих мовних моделей відіграло ключову роль в процесі виявлення індикаторів фішингу, забезпечивши високий рівень ефективності.

Модуль прийняття рішень виконав задачу з оцінки отриманих результатів та прийняття остаточного рішення про фішинговий статус електронних листів.

Такий багатогранний підхід забезпечив адаптацію інструменту до широкого спектру фішингових стратегій, постійне навчання та вдосконалення алгоритмів виявлення.

Технологія великих мовних моделей є ефективною для виявлення фішингових атак та має великий потенціал застосування.

Висновки дипломної роботи

У ході виконання дипломної роботи було розроблено програмний засіб для виявлення фішингових атак за допомогою великих мовних моделей. Дослідження охоплювало кілька етапів, кожен з яких забезпечив вклад у загальний успіх роботи.

У першому розділі було проведено детальний аналіз типових елементів електронного листа та визначено ключові індикатори, які можуть вказувати на фішингову атаку. Було розглянуто різні техніки, що використовуються зловмисниками для створення фішингових листів, що дозволило сформулювати чіткі критерії для виявлення таких атак.

Другий розділ зосередився на виборі відповідних великих мовних моделей для реалізації завдання. Було проаналізовано різні моделі, їхні потужності та можливості розгортання. Враховуючи необхідність обробки великих обсягів даних та забезпечення високої продуктивності, перевагу було надано моделям з підтримкою API. Також було описано архітектуру програмного засобу, розроблено діаграми варіантів використання та потоків даних. Основним форматом вхідних даних було обрано EML, що забезпечує сумісність із більшістю поштових клієнтів.

У третьому розділі було детально описано процес розробки Було створено та інтегровано 6 модулів, таких як модуль підготовки даних, модуль перевірки адреси відправника, модуль перевірки вкладених URL адрес, модуль перевірки основного тексту повідомлення та модуль перевірки вкладених файлів на наявність шкідливого програмного забезпечення та модуль прийняття рішень. Кожен модуль відігравав свою роль у загальній структурі засобу, забезпечуючи ефективне виявлення фішингових атак.

У четвертому розділі було проведено тестування розробленого програмного засобу виявлення фішингових атак із застосуванням великих мовних мереж. Результати тестування показали високу ефективність засобу, що підтверджується високими значеннями метрик точності та продуктивності.

Таким чином, розроблений програмний засіб виявлення фішингових атак показав високу ефективність у тестуванні на реальних та штучно створених наборах даних. Він здатний адаптуватися до нових фішингових технік завдяки інтеграції з великими мовними моделями, що забезпечує постійне вдосконалення алгоритмів виявлення. Це робить розроблений інструмент надійним рішенням для виявлення та запобігання фішинговим атакам у сучасних інформаційних системах. Великі мовні моделі мають великий потенціал для виявлення фішингових атак.

Перелік джерел посилань

1. Phishing For Dummies®, Cisco Special Edition by Gabrielle Bridgers, Christina Hausman, Adam Tomeo, and Ganesh Vellala Umapathy
Anti-Phishing Working Group (2020). Phishing activity trends report. https://docs.apwg.org/reports/ apwg_trends_report_q2_2020.pdf.
2. Detecting Scams Using Large Language Models, Liming Jiang https://arxiv.org/html/2402.03147v1#:~:text=Phishing%20Detection%3A%20LLMs%20can%20assist,phishing%20patterns%20and%20emerging%20threats.
3. URL Analysis: How to Determine Maliciousness https://hustlelead.medium.com/url-analysis-how-to-determine-maliciousness-f630b4e51b9e
4. What Are Large Language Model (LLM) Agents and Autonomous Agents https://promptengineering.org/what-are-large-language-model-llm-agents/
5. OpenAI Libraries: https://platform.openai.com/docs/libraries
6. Getting started with LLM prompt engineering https://learn.microsoft.com/en-us/ai/playbook/technology-guidance/generative-ai/working-with-llms/prompt-engineering
7. MetaDefender Cloud API v4 https://docs.opswat.com/mdcloud/metadefender-cloud-api-v4
8. Jain, A. K., & Gupta, B. B. (2018). Phish-safe: Url features-based phishing detection system using machine learning. In M. U. Bokhari, N. Agrawal, and D. Saini (Eds.), Cyber Security. Singapore: Springer Singapore.
9. El Aassal, A., Baki, S., Das, A., & Verma, R. M. (2020). An in-depth benchmarking and evaluation of phishing detection research for security needs. IEEE Access, 8, 22170–22192.