

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

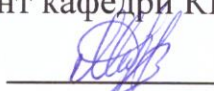
ПОЯСНЮВАЛЬНА ЗАПИСКА

до випускної кваліфікаційної роботи бакалавра
на тему: «Проект інтелектуальної підсистеми захисту
від фішингових атак»

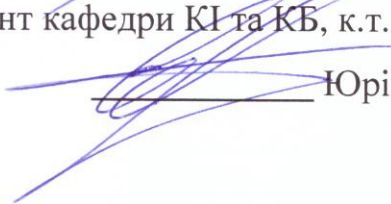
Виконав: студент 4-го курсу групи КБ-21-2
спеціальності 125 «Кіберберзпека»

 Максим ПАРАЩУК

Керівник: Доцент кафедри КІ та КБ, к.т.н

 Олександр МАЄВСЬКИЙ

Рецензент: Доцент кафедри КІ та КБ, к.т.н.

 Юрій БРОДСЬКИЙ

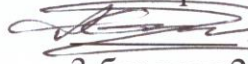
Житомир – 2025

Державний університет «Житомирська політехніка»
Факультет інформаційно-комп'ютерних технологій
Кафедра комп'ютерної інженерії та кібербезпеки
Спеціальність 125 «Кібербезпека»
Освітня програма «Кібербезпека»

ЗАТВЕРДЖУЮ

Завідувач

кафедри комп'ютерної
інженерії та кібербезпеки

 Андрій Єфіменко
3 березня 2025 р.

ЗАВДАННЯ

на випускню кваліфікаційну роботу бакалавра

Студент: Паращук Максим Анатолійович

Тема роботи: «Проект інтелектуальної підсистеми захисту від фішингових атак»

затверджена Наказом університету від 3 березня 2025 р. № 80/с

Термін здачі студентом закінченої роботи 27 червня 2025 р.

Вихідні дані роботи (зазначається предмет і об'єкт дослідження):

Предметом дослідження є методи виявлення фішингових атак на основі аналізу URL-адрес та вмісту вебсторінок.

Об'єктом дослідження є інтелектуальна система, що здійснює автоматичну перевірку сайтів на предмет фішингових загроз із використанням алгоритмів машинного навчання, вбудованого HTML-парсингу, а також перевірок через зовнішні сервіси безпеки.

Обладнання та програмне забезпечення:

1. Visual Studio Code
2. Python

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

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв

Студент  Максим ПАРАЩУК
Керівник  Олександр МАЄВСЬКИЙ

Календарний план

№ з/п	Назви етапів випускної кваліфікаційної роботи	Термін виконання етапів роботи	Примітка
1	Постановка завдання	03 березня	
2	Збір та аналіз наукових джерел за тематикою фішингу, ШІ, методів виявлення	05 березня	
3	Аналіз наявних методів та інструментів захисту від фішингу	10 березня	
4	Формування вимог до програмної системи. Проектування архітектури рішення	01 квітня	
5	Підготовка та аналіз датасетів, векторизація ознак URL та HTML	20 квітня	
6	Навчання моделі з використанням Keras	02 травня	
7	Реалізація системи у вебінтерфейсі Flask	20 травня	
8	Тестування, оцінка якості	2 червня	
9	Написання пояснювальної записки	17 червня	
10	Підготовка до захисту	22 червня	
11	Захист	27 червня	

Студент Сто Максим ПАРАЩУК

Керівник Маєвський Олександр МАЄВСЬКИЙ

АНОТАЦІЯ

Метою кваліфікаційної роботи є розробка інтелектуальної підсистеми захисту від фішингових атак.

У роботі проведено огляд актуальних кіберзагроз, пов'язаних із фішинговими атаками, а також розглянуто сучасні підходи до побудови систем виявлення шкідливих вебресурсів. Проаналізовано структуру URL-адрес, методи обробки текстової та структурної інформації з HTML-коду, а також ефективність моделей штучного інтелекту у виявленні фішингу.

Дипломна робота має обсяг 78 сторінок, містить 52 рисунка, 9 таблиць, 18 літературних джерел та 3 додатка.

У процесі виконання роботи визначено основні завдання для побудови системи виявлення фішингових сайтів, зокрема: автоматичний збір ознак з URL-адрес, аналіз HTML-контенту та перевірка сайтів за допомогою зовнішніх сервісів. Проаналізовано методи класифікації, зокрема нейронні мережі та побудовано модель із використання фреймворку Keras. Розглянуто HTML-парсинг як додатковий рівень аналізу. Показано розробку повнофункціональної системи з вебінтерфейсом на Flask. Наведено результати тестування, які підтвердили високу точність моделі та практичну ефективність розробленої системи в умовах реального інтернет-середовища.

Ключові слова: ФІШИНГ, ІНТЕЛЕКТУАЛЬНА СИСТЕМА, АНАЛІЗ URL, МАШИННЕ НАВЧАННЯ, НЕЙРОННІ МЕРЕЖІ.

					ККІКБ.КР.Б - 125 - 15 - ПЗ				
Змн.	Арк.	№ докум.	Підпис	Дата	Проект інтелектуальної підсистеми захисту від фішингових атак	Літ.	Арк.	Аркушів	
Розроб.		Паращук М.А.		24.06					
Керівник		Маєвський О.В.		25.06			4	78	
Реценз.		Бродський Ю.Б.		25.06		ФІКТ, Гр. КБ-21-2			
Н. контр.		Маєвський О.В.		25.06					
Затверд.		Єфіменко А.А.		25.06					

ANNOTATION

The purpose of the qualification work is to develop an intelligent subsystem for protection against phishing attacks.

The paper reviews current cyber threats related to phishing attacks, and also considers modern approaches to building systems for detecting malicious web resources. The structure of URL addresses, methods for processing text and structural information from HTML code, and the effectiveness of artificial intelligence models in detecting phishing are analyzed.

The thesis has a volume of 78 pages, contains 52 figures, 9 tables, 18 literary sources and 3 appendices.

In the process of performing the work, the main tasks for building a system for detecting phishing sites were identified, in particular: automatic collection of features from URL addresses, analysis of HTML content and verification of sites using external services. Classification methods were analyzed, in particular neural networks, and a model was built using the Keras framework. HTML parsing was considered as an additional level of analysis. The development of a fully functional system with a web interface on Flask was shown. Testing results were presented, which confirmed the high accuracy of the model and the practical effectiveness of the developed system in a real Internet environment.

Keywords: PHISHING, INTELLIGENT SYSTEM, URL ANALYSIS, MACHINE LEARNING, NEURAL NETWORKS.

		Паращук М.А			ККІКБ.КР.Б – 125 – 15 – ПЗ	Арк.
		Маєвський О.В				5
Змн.	Арк.	№ докум.	Підпис	Дата		

Зміст

ВСТУП.....	8
РОЗДІЛ 1 АНАЛІЗ ПРОБЛЕМ ФІШИНГУ ТА МЕТОДІВ ЙОГО ВИЯВЛЕННЯ	10
1.1 Фішинг як загроза кібербезпеці	10
1.2 Методи реалізації фішингових атак.....	13
1.3 Огляд сучасних підходів до виявлення фішингових сайтів	14
1.4 Порівняння та недоліки існуючих рішень	17
РОЗДІЛ 2 ВИКОРИСТАННЯ МЕТОДІВ ШТУЧНОГО ІНТЕЛЕКТУ	22
2.1 Теоретичні основи машинного навчання для задач класифікації	22
2.2 Формування ознак на основі URL-адрес	24
2.3 Аналіз HTML-контенту вебсторінок.....	28
2.4 Побудова та навчання моделі на основі фреймворку Keras	32
2.5 Загальна архітектура системи	36
РОЗДІЛ 3 РЕАЛІЗАЦІЯ ТА ДОСЛІДЖЕННЯ ІНТЕЛЕКТУАЛЬНОЇ ПІДСИСТЕМИ ВИЯВЛЕННЯ ФІШИНГУ	41
3.1 Реалізація системи виявлення фішингових сайтів	41
3.2 Розробка вебінтерфейсу користувача	48
3.3 Приклад роботи системи.....	51
3.4 Аналіз ефективності моделі та порівняння з аналогами	53
ВИСНОВКИ	66
СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ	68
ДОДАТКИ	70

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

AI- Artificial Intelligence

ML- Machine Learning

NN – Neural Network

Keras-Python-бібліотека для побудови нейронних мереж

URL- Uniform Resource Locator

HTML- HyperText Markup Language

HTTPS- HyperText Transfer Protocol Secure

FPR- False Positive Rate

TPR- True Positive Rate

DNS- Domain Name System

API- Application Programming Interface

IP-Internet Protocol

CSV- Comma-Separated Values

		Паращук М.А			ККІКБ.КР.Б – 125 – 15 – ПЗ	Арк.
		Маєвський О.В				7
Змн.	Арк.	№ докум.	Підпис	Дата		

ВСТУП

Актуальність теми. Сучасний розвиток інформаційних технологій спричиняє зростання кількості кібератак, серед яких особливо небезпечними є фішингові атаки. Фішинг є один із найпоширеніших методів шахрайства, спрямованих на викрадення конфіденційних даних користувачів через імітацію легітимних вебресурсів. Через швидке розповсюдження інтернету та зростання обсягів онлайн-транзакцій проблема забезпечення безпеки вебсайтів та захисту користувачів набуває все більшої уваги.

Виявлення фішингових сайтів є складним завданням через постійну еволюцію методів атакуючих та необхідність оперативного реагування. Тому розробка ефективних інтелектуальних систем для автоматичного розпізнавання фішингових ресурсів є актуальним напрямом досліджень у сфері кібербезпеки.

Метою кваліфікаційної роботи є розробка інтелектуальної підсистеми захисту від фішингових атак, яка базується на аналізі URL-адрес та HTML-контенту вебресурсів із застосуванням методів машинного навчання.

Визначена мета обумовлює наступні завдання:

- Проаналізувати актуальні кіберзагрози, пов'язані з фішинговими атаками;
- Вивчити існуючі підходи до виявлення фішингових сайтів;
- Розробити методику формування ознак з URL та HTML-коду;
- Створити модель на основі нейронних мереж з використанням фреймворку Keras;
- Реалізувати вебінтерфейс для взаємодії користувача з системою;
- Провести тестування та оцінку ефективності розробленої системи;

Об'єктом дослідження є процес виявлення та класифікації фішингових вебресурсів на основі аналізу їхніх URL-адрес та HTML-контенту.

Предметом дослідження є методи та алгоритми машинного навчання для автоматичного розпізнавання фішингових сайтів за допомогою аналізу URL та HTML-структури вебсторінок.

		Паращук М.А			ККІКБ.КР.Б – 125 – 15 – ПЗ	Арк.
		Маєвський О.В				8
Змн.	Арк.	№ докум.	Підпис	Дата		

У процесі використання кваліфікаційної роботи було розроблено проєкт інтелектуальної підсистеми виявлення фішингових сайтів з вебінтерфейсом, яка поєднує аналіз URL, HTML-контенту та зовнішніх перевірок.

		Паращук М.А			ККІКБ.КР.Б – 125 – 15 – ПЗ	Арк.
		Маєвський О.В				9
Змн.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 1 АНАЛІЗ ПРОБЛЕМ ФІШИНГУ ТА МЕТОДІВ ЙОГО ВИЯВЛЕННЯ

1.1 Фішинг як загроза кібербезпеці

У сучасному світі кібербезпека є однією з складових інформаційного середовища як для окремих користувачів, так і для організацій, бізнесу, державних структур. Активне впровадження цифрових технологій у повсякденне життя призвело до того, що значна частина дій – фінансових, соціальних, адміністративних – перемістилась в онлайн простір. Разом із тим створило зручне поле для зловмисників, які розробляють нові способи шахрайства. Одним із найпоширеніших та водночас небезпечних методів є фішинг.[5]

Фішинг(від англ. Phishing, співзвучно з fishing-риболовля)- це вид кіберзлочинності, суть якого полягає у виманюванні особистої інформації у користувачів шляхом створення підроблених (фальшивих) вебсайтів, електронних листів або повідомлень у месенджерів, що візуально і функціонально імітують справжні ресурси.

Ці атаки зазвичай спрямовані на отримання паролів, логінів, даних банківських рахунків, номерів соціального страхування, тощо. Зловмисники можуть використовувати отриману інформацію для крадіжки грошей, зламу облікових записів, а іноді – й для більш серйозних атак, наприклад, шпигунства або саботажу.

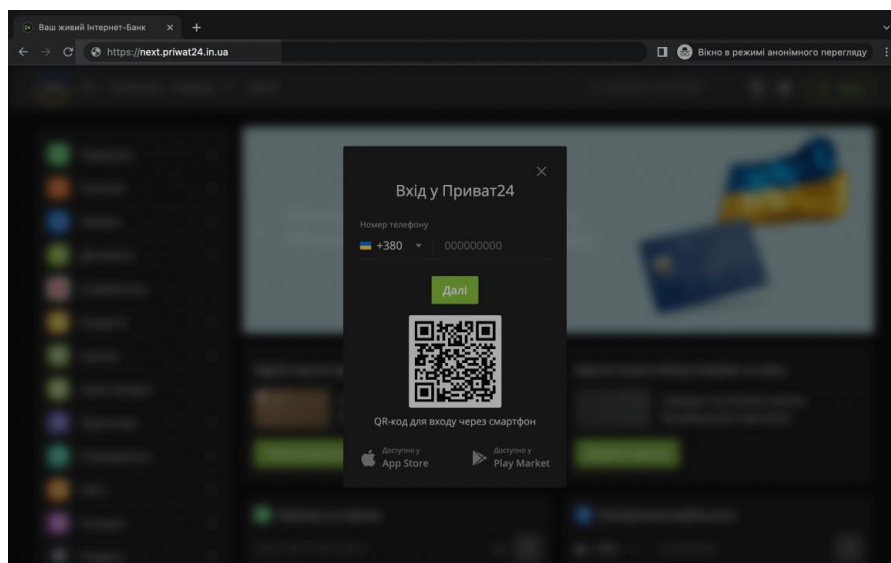


Рисунок 1.1- Приклад фішингово сайту, що імітує стрінку входу в банківській сервіс

		Паращук М.А			ККІКБ.КР.Б – 125 – 15 – ПЗ	Арк.
		Маєвський О.В				10
Змн.	Арк.	№ докум.	Підпис	Дата		

Поширення фішингу.

За останні роки кількість фішингових атак зростає. Це пояснюється не лише зростанням кількості користувачів інтернету, а й підвищеною ефективністю цього методу шахрайства. У звітах таких організацій, як Google Safe Browsing, PhishLabs, Cisco, Anti-Phishing Working Group(APWG) регулярно фіксується збільшення зафіксованих фішингових доменів, особливо у період фінансової нестабільності або соціальної напруги.[13,15]

Окрім того, фішинг сьогодні використовується не лише для масових атак, а й у таргетованих кампаніях, які спрямовані на конкретній організації або посадових осіб. Такий фішинг називається spear phishing, і він є ще більш небезпечним, адже ретельно підготовлений і має ймовірність успіху.[3]

Причини поширеності фішингу.

Фішинг має кілька переваг для зловмисників:

Низька вартість реалізації: достатньо створити підроблений сайт або розіслати масову фішингову розсилку.

Мінімальні технічні вимоги: базові навчки веброзробки та копіювання шаблонів сторінки.

Соціальна інженерія: багато користувачів схильні вірити 'офіційним' повідомленням, особливо якщо ті нагадують відомі бренди(банки, сервіси, пошта,соцмережі).

Види фішингу.

Існує кілька типів фішингових атак:

1. Класичний фішинг- підроблені сайти, що імітують зовнішній вигляд справжній.
2. Email-фішинг – фальшиві електронні листи від імені банків, служб безпеки тощо.
3. Spear phishing- індивідуальні листи або повідомлення з конкретною інформацією про жертву.

		Паращук М.А			ККІКБ.КР.Б – 125 – 15 – ПЗ	Арк.
		Маєвський О.В				11
Змн.	Арк.	№ докум.	Підпис	Дата		

4. Vishing(voice phishing)- телефонні дзвінки, під час яких зловмисник прикидається співробітником служби безпеки.
5. Smishing(SMS phishing)- надсилання SMS із шкідливим посиланням або закликом передати особисту інформацію.
6. Clone phishing- атаки, в яких легітимний лист копіюється, змінюється посилання і відправляється знову.

У таблиці 1.1 наведено класифікацію основних типів фішингових атак залежно від каналу атаки, способу реалізації та цільової аудиторії.

Таблиця 1.1

Класифікація видів фішингу

Тип фішингу	Канал атаки	Ціль	Опис
Класичний фішинг	Вебсайти, email	Широке коло користувачів	Підроблений сайт, що імітує справжній сервіс
Email-фішинг	Електронна пошта	Масові користувачі	Розсилка листів з фейковими посиланням
Spear phishing	Email, соцмережі	Конкретна особа або організація	Цілеспрямована атака з персоналізованим змістом
Smishing	SMS	Користувачі мобільних пристроїв	Повідомлення з посиланням
Vishing	Телефонні зв'язки	Будь-який користувач	Імітація розмови з банком/службою безпеки
Clone phishing	Email	Користувач який вже отримав легітимний лист	Повторне надсилання копії справжнього листа з підміною URL

Фішинг є однією з причин: масового злону облікових записів, фінансові втрати, витоку особистих даних, втрати репутації організації, компрометації державних структур.

1.2 Методи реалізації фішингових атак

Фішингові атаки можуть реалізовуватись різними способами, від простих до високотехнологічних, залежно від рівня підготовки зловмисника, обраної цілі, платформи доставки та механізмів маскування. У своїй основі всі ці методи мають спільну мету – змусити жертву добровільно передати конфіденційну інформацію зловмиснику.

Одним із ключових аспектів фішингових атак є використання соціальної інженерії, тобто психологічного впливу на жертву. Зловмисники часто апелюють до емоції: страху(наприклад повідомлення про блокування картки), довіри(лист від банку), терміновість або ж цікавість (ви виграли приз).[3]

Основні етапи фішингових атак.

Більшість фішингових кампаній проходять через такі етапи:

1. Підготовка інфраструктури: зловмисник створює фальшивий сайт, реєструє схожий домен (іноді з використанням гомогліфів або схожих символів), або використовує компроментований сервер.

2. Маскування атаки: підроблений сайт маскується під справжній – копіює дизайн, логотипи, URL, навіть сертифікати HTTPS(через безкоштовні сервіси).

3. Розсилка: найчастіше через email, але також можливі варіанти через соцмережі, SMS, месенджери чи навіть рекламу.

4. Взаємодія з жертвою: користувач переходить по посиланню, потрапляє на фейкову сторінку де вводить конфіденційні дані.

5. Передача даних зловмиснику: введена інформація миттєво зберігається на сервері атакуючого або передається у реальному часі.

6. Завершення атаки: у деяких випадках користувача автоматично перенаправляють на справжній сайт, щоб уникнути підозр.[4-5]

Технічні методи реалізації.

		Паращук М.А			ККІКБ.КР.Б – 125 – 15 – ПЗ	Арк.
		Маєвський О.В				
Змн.	Арк.	№ докум.	Підпис	Дата		13

Фішингові атаки можуть реалізовуватись за допомогою такої технічних прийомів:

1. Клонування сайтів: використовується програма(наприклад, HTTrack або SocialFish) яка копіює HTML, CSS, та JS-код справжніх сайтів.

2. Реєстрація схожих доменів: часто з використанням гомогліфів(наприклад google.com де дві літери 'о' замінені на крилицю. Схожих символів(наприклад googl3.com, go0gle.com). Додавання зайвих слів (secure-google-login.com).

3. Застосування редиректів: перенаправлення користувача через серію доменів для ускладнення виявлення джерела.

4. HTTPS-сертифікати: використання безкоштовних сертифікатів щоб переконати користувачів у 'безпеці' сайту.

5. JavaScript-обфускація: приховання коду, що краде дані, за допомогою складного або шифрованого JavaScript.[7-8]

Методи реалізації фішингових атак постійно вдосконалюються, поєднуючи технічні хитрощі з психологічним тиском на жертву. Ці атаки є відносно дешевими для зловмисників, але водночас надзвичайно небезпечними, оскільки здатні призвести до серйозних фінансових та репутаційних втрат.

Для ефективного протистояння таким загрозам важливо не лише вдосконалювати технологічні засоби виявлення фішингу, а й підвищувати обізнаність користувачів щодо типових ознак шахрайських дій.

1.3 Огляд сучасних підходів до виявлення фішингових сайтів

З кожним роком фішингові атаки стають дедалі виточенішими, що зумовлює необхідність удосконалення методів їх виявлення. Традиційні методи, які раніше дозволяли успішно фільтрувати шкідливі ресурси, сьогодні часто є недостатніми через швидку зміну доменів, складність маскування та використання HTTPS-сертифікатів навіть на фейкових сторінках. У цьому підрозділі розглянемо основні сучасні підходи до виявлення фішингових сайтів, їх особливості, переваги і недоліки.

Методи на основі чорних списків (Blacklist-based).

		Паращук М.А			ККІКБ.КР.Б – 125 – 15 – ПЗ	Арк.
		Маєвський О.В				
Змн.	Арк.	№ докум.	Підпис	Дата		14

Найпростішим і найстарішим підходом є використання чорних списків доменів або IP-адрес, які вже були виявлені як фішингові. Такі списки ведуться спеціальними сервісами такими як Google Safe Browsing, PhishTank, OpenPhish, які постійно оновлюють свої бази даних.

Переваги: Висока швидкість перевірки, простота інтеграції у браузері, антивіруси тощою

Недоліки: Неможливість виявлення нових, ще не зареєстрованих сайтів, Вразливість до частих змін доменів зловмисниками.

Аналіз URL-адрес(Heuristic and Feature-based URL analysis).

Цей підхід базується на перевірці структури URL-адреси: довжини, кількість піддоменів, наявність підозрілих символів, використання IP-адрес замість доменів, схожість з легітимними брендами тощо. У такому випадку застосовується аналіз за набором ознак (features), які можуть бути передані в алгоритм класифікації.

Приклади ознак: кількість крапок у домені, наявність слова 'login', 'bank', 'secure', гомогліфи або обфускація символів.

Переваги: можливість виявлення загроз без попереднього внесення в базу, широкі можливості для автоматизації.

Недоліки: висока чутливість до зміни шаблонів, можливість помилкових спрацювань (false positive).

Порівняльний аналіз основних методів подано в таблиці 1.2

Таблиця 1.2

Порівняння основних методів виявлення фішингових сайтів за ознаками URL

Метод	Принцип роботи	Переваги	Недоліки
Чорні списки	Перевірка URL у відомих базах фішингових сайтів	Висока швидкість, простота	Не виявляє нові/змінені домени
Гештальт-аналіз URL	Перевірка довжини, доменів, піддоменів, символів	Легкий у реалізації, незалежний від бази	Часті false-positive, легко обходиться

Пошук брендних слів	Виявлення слів типу login, bank, secure у URL	Дозволяє фокусуватись на цільових атаках	Не завжди присутні у фішингових адресах
---------------------	---	--	---

Таблиця 1.2(продовження)

Аналіз гомогліфів	Виявлення підміни символів (наприклад, g00gle vs google)	Актуально для маскування відомих брендів	Потребує словників і складних перевірок
Машинне навчання (ML)	Класифікація на основі набору ознак з URL	Автоматичне навчання, виявляє нові шаблони	Вимагає датасетів, налаштування, обчислень
Гібридний підхід	Поєднання кількох методів	Висока точність, універсальність	Складність реалізації, більше ресурсів

Аналізу HTML-контенту сторінки.

Окрім URL, сучасні системи аналізують сам HTML- код сторінки. Тут можуть враховуватися: наявність форм для введення паролів, підозрілі JavaScript-скрипти, використання фреймів(iframe), дублювання дизайну відомих ресурсів.[17]

Для цього використовують інструменти на кштал BeautifulSoup, Selenium або спеціальні парсери. У комплексі з іншими підходами HTML-аналіз значно підвищує точність.

Гібридні системи.

Комбінування кількох підходів – це найефективніший спосіб виявлення фішингу. Система може спочатку перевірити домен по базі чорного списку, потім проаналізувати URL та HTML, а в разі підозри використати модель машинного навчання для остаточного рішення.

Переваги: висока точність, зниження кількості помилкових спрацювань, гнучкість адаптації до нових атак.

Недоліки: складність реалізування, вища обчислювальна вартість.

Методи машинного навчання.

		Паращук М.А			ККІКБ.КР.Б – 125 – 15 – ПЗ	Арк.
		Маєвський О.В				
Змн.	Арк.	№ докум.	Підпис	Дата		16

Широкого поширення набули методи машинного навчання(ML), які навчаються на великих наборах URL (фішингові та легітимні) для подальшого прогнозування. Найпопулярніші моделі:Random Forest, Support Vector Machine(SVM), Multilayer Perceptron(MLP), Convolutional Neural Networks(CNN), Keras-моделі які дозволяють гнучко багаторівневі нейронні мережі.[9,11]

Інші підходи.

До додаткових методів можна віднести: DNS-аналіз(перевірка домені, WHOIS-даних, часу життя), аналіз поведінки користувача, графовий аналіз.

Ці методи менш популярні, але активно розвиваються зокрема у сфері корпоративної кібербезпеки.

Сучасні системи виявлення фішингових сайтів базуються на комбінації декількох підходів – від простих списків до складних нейромереж. Кожен метод має свої сильні й слабкі сторони, тому найбільш ефективним є багаторівневі гібридні рішення. Вибір оптимального підходу залежить від задачі, масштабів системи та доступність ресурсів.

1.4 Порівняння та недоліки існуючих рішень

У сучасному цифровому середовищі, де зловмисники постійно вдосконалюють свої інструменти, системи виявлення фішингу повинні бути не лише точними а й адаптованими. Відтак важливо не лише знати методи виявлення фішингу, а й розуміти їх сильні і слабкі сторони, рівень ефективності та практичну реалізованість.

Огляд основних типів рішень.

На практиці використовуються такі основні підходи до виявлення фішингових сайтів:[1,5]

1. Засновані на бази даних
2. Евристичні методи
3. Модель машиного навчання
4. Гібридні системи, що поєднують кілька підходів

Ці підходи реалізуються як окремими компаніями так і в рамках відкритих проєктів.

		Паращук М.А			ККІКБ.КР.Б – 125 – 15 – ПЗ	Арк.
		Маєвський О.В				17
Змн.	Арк.	№ докум.	Підпис	Дата		

На таблиці 1.3 можна побачити основні рішення для виявлення фішингу різними методами.

Таблиця 1.3

Платформа / Інструмент	Підхід до виявлення	Доступність	Актуальність оновлення	Недоліки
Google Safe Browsing	Чорні списки + евристика	Інтегровано в Chrome	Висока	Не розкриває внутрішню логіку
PhishTank	Чорні списки (краудсорсинг)	Відкритий доступ	Помірна	Потребує ручного підтвердження
VirusTotal	Багатоаналіз + API	Частково платний	Висока	Обмеження у безкоштовному режимі
OpenPhish	Евристика + сигнатури	Частково відкритий	Середня	Може давати false-negative
Custom ML Solutions	ML/NN	Власні рішення	Залежить від розробника	Потребує великих ресурсів
Cisco Umbrella	DNS + телеметрія	Платно	Висока	Не підходить для малих систем

Переваги та недоліки відомих рішень.

Google Safe Browsing.

Google Safe Browsing є, без сумніву, одним із найвідоміших захисних механізмів проти фішингу. Він вбудований у більшість сучасних браузерів.[13]

Переваги: швидкий та простий у використанні, постійно оновлюється база, інтеграція на рівні браузера.

Недоліки: чорний ящик- відсутність відкритого API-ключа для повного доступу, залежність від централізованої бази, затримка в оновленні нових фішингових доменів.

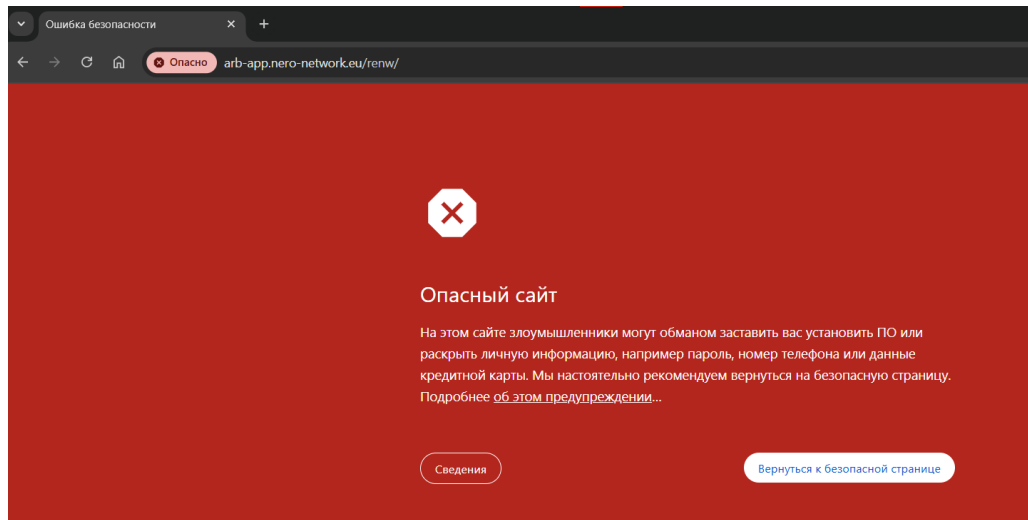


Рисунок 1.2 Механізм роботи Google Safe Browsing

PhishTank.

Це сервіс працює на основі краудсорсингу – користувачі самостійно надсилають підозрілі посилання, які згодом проходять перевірку. Це дозволяє швидко виявити нові загрози, проте є кілька обмежень.[12]

Переваги: відкрите API, активна спільнота, зрозуміла процедура верифікації.

Недоліки: людський фактор у перевірці, повільне оновлення для деяких ресурсів, може бути легко обійдено фішерами.

Порівняння за критеріями.

Щоб об'єктивно оцінити ефективність різних систем, потрібно враховувати такі показники:

1. True Positive Rate(TPR)- скільки реальних фішингів було виявлено
2. False Positive Rate(FPR)- скільки легітимних сайтів було помилково визначено як фішинг.
3. Latency – швидкість прийняття рішень.
- 4.Scalability – машабованість ознак.
5. Explainability- прозорість рішень.

Основні проблеми існуючих рішень.

		Паращук М.А			ККІКБ.КР.Б – 125 – 15 – ПЗ	Арк.
		Маєвський О.В				
Змн.	Арк.	№ докум.	Підпис	Дата		19

Незважаючи на значний прогрес, жоден підхід не є досконалим. Основні проблеми:

1. Застарілість даних. Фішингові сайти можуть існувати лише декілька годин, тому системи, що оновлюються рідко, стають неефективними.

2. False Positive та False Negative. Деякі сайти блокуються помилково, інші не виявляються вчасно.

3. Недостатній аналіз HTML-контенту. Багато систем не використовують повноцінний аналіз структури сторінки.

4. Висока вартість або закриті. Комерційні продукти часто мають обмеження або високу ціну.

5. Адаптивність атак. Фішинг постійно еволюціонує – від email – розсилок до SMS, месенджерів, QR-кодів та навіть AI- згенерованих сайтів.

Тенденції розвитку систем виявлення фішингу.

На основі аналізу можна зробити висновки щодо майбутнього:

1. Інтеграція AI/ML- дедалі більше рішень переходять до нейронних мереж, здатних адаптуватися до нових шаблонів.

2. Комплексний підхід – найкращі системи поєднують URL-аналіз, HTML-аналіз, DNS, поведінку користувача.

3. Автоматичне оновлення моделей – використання технологій автонавчання для зменшення ручної участі.

4. Збір телеметрії – використання даних браузера або операційної системи для швидкої реакції.

Отже, сучасні системи виявлення фішингових сайтів демонструють значну різноманітність як за методологією, так і за якістю реалізації. Прості рішення на основі чорних списків швидкі та дешеві, але вразливі до нових атак. ML – рішення забезпечують високу точність, однак вимагають багато ресурсів і даних. Найбільш перспективними вважаються гібридні системи з адаптивними механізмами оновлення та розширеним аналізом URL/HTML.

Власне розроблена система, яку буде розглянуто в наступних розділах, побудована саме на таких принципах.

		Паращук М.А			ККІКБ.КР.Б – 125 – 15 – ПЗ	Арк.
		Маєвський О.В				20
Змн.	Арк.	№ докум.	Підпис	Дата		

Висновок до першого розділу

У першому розділі було проведено ґрунтовний аналіз проблем фішингових атак, які на сьогодні є однією з найпоширеніших та найнебезпечніших форм кіберзагроз. З'ясовано, що фішинг не лише активно еволюціонує, використовуючи різноманітні канали доставки та соціальну інженерію, але й ускладнюється завдяки використанню гомогліфів, шифрування та динамічного HTML-коду.

Розглянуто основні методи реалізації фішингових атак, серед яких особливу небезпеку становлять ті, що імітують вже відомі сервіси та банки, маніпулюючи довірою користувачів. Це підтверджує необхідність побудови багаторівневої системи виявлення.

У межах огляду сучасних підходів до виявлення фішингових сайтів було досліджено різні рішення: від традиційних чорних списків до моделей, що базується на штучному інтелекті. Показано, що найвищу ефективність демонструють гібридні системи.

Проведено порівняння рішень, дозволило виявити основні недоліки існуючих методів – зокрема обмеження по швидкості оновлення, проблеми з хибними спрацюваннями та складність у масштабуванні. Ці висновки стали основою для обґрунтування потреби у створенні власної інтелектуальної підсистеми виявлення фішингових ресурсів.

		Паращук М.А			ККІКБ.КР.Б – 125 – 15 – ПЗ	Арк.
		Маєвський О.В				
Змн.	Арк.	№ докум.	Підпис	Дата		21

РОЗДІЛ 2 ВИКОРИСТАННЯ МЕТОДІВ ШТУЧНОГО ІНТЕЛЕКТУ

2.1 Теоретичні основи машинного навчання для задач класифікації

У сучасному цифровому середовищі методи машинного навчання (ML) стали основою числених рішень у сфері аналізу даних, виявлення закономірності та автоматичної класифікації. Зокрема, при побудові систем для виявлення фішингових сайтів класифікації відіграє ключову роль, оскільки дозволяє віднести вхідні дані до однієї з двох категорій: фішинговий або легітимний веб ресурс.

Основні поняття машиного навчання.

Машинне навчання – це розділ штучного інтелекту, який передбачає побудову моделей, здатних навчатися на основі даних, а не заздалегіть запрограмованих інструкцій. У межах машинного навчання виділяють кероване (supervised), не кероване (unsupervised), та підкріплювальне навчання (reinforcement learning).

У задачах виявлення фішингових сайтів найчастіше використовують кероване навчання, де моделі навчаються на основі мічених даних – тобто таких, де відома правильна відповідь, мітка фішинг чи легітимний.



Рисунок 2.1 Класифікація основних типів машинного навчання

Класифікація як задача ML.

Задача класифікації передбачає побудови функції $f(x)$, яка на основі вектора озна x прогнозує клас y . У контексті фішингу – це бінарна класифікація де: клас 0 – легітимний сайт, клас 1 – фішинговий клас.

Модель навчається на тренувальній вибірці, яка включає приклади URL-адрес з відповідною міткою. Після навчання модель має бути здатньо узагальнювати – тобто коректно класифікувати нові, раніше не бачені зразки.

Основні алгоритми класифікації.

Найпоширенішими алгоритмами для задач виявлення фішингу є:

1. Логістична регресія(Logistic Regression)- проста, але потужна модель для бінарної класифікації.
2. Дерева рішень(Decision Trees) та випадкові ліси(Random Forest) – добре працюють на структурованих даних.
3. Методи опорних векторів (SVM)- ефективні для складних розмежувальних гіперплощин
4. Нейронні мережі(NN) – гнучкі моделі, здатні обробляти великі масиви даних з високою точністю.

Таблиця 2.1

Алгоритм	Переваги	Недоліки	Застосування
Logistic Regression	Простота, швидкість	Обмежена складність	Простий аналіз URL
Decision Tree	Інтерпретованість, швидкість	Схильність до переобучення	Ознаки URL та HTML
Random Forest	Стабільність, точність	Важкий для інтерпретації	Змішані ознаки
SVM	Висока точність	Повільність на великих даних	Малі вибірки
Neural Networks	Гнучкість, висока точність	Вимогливі до ресурсів, «чорна скринька»	Великі датасети

Метрики якості класифікації.

Щоб оцінити, наскільки добре модель виконує свою функцію, використовують метрики класифікації:

1. Accuracy – загальна точність.
2. Precision – точність виявлення фішингових сайтів.
3. Recall(sensitivity) – здатність моделі знаходити всі фішингові сайти.
4. F1-score – середнє між Precision і Recall

5. ROC AUC – якість моделі при різних порогах.

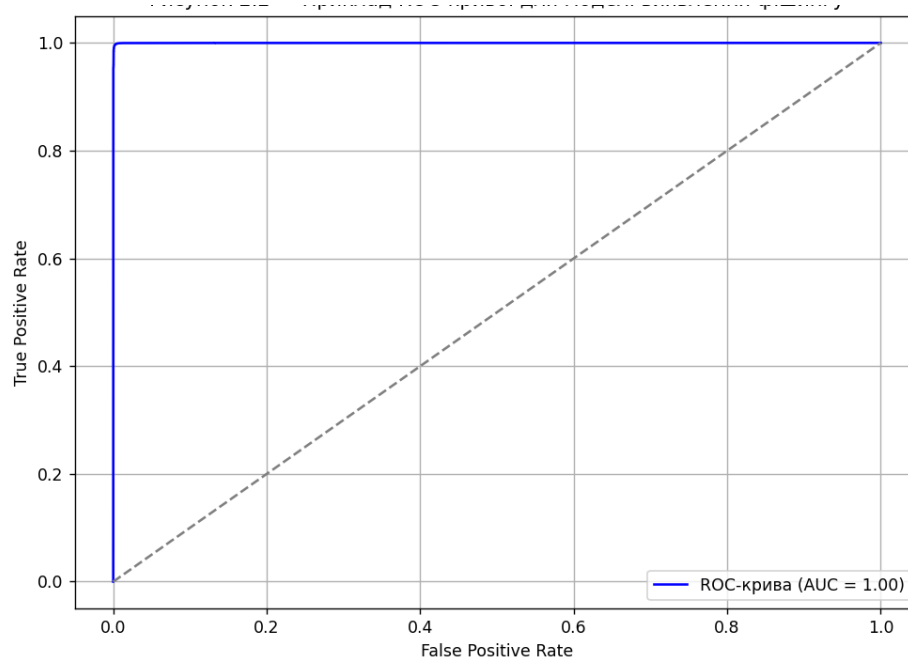


Рисунок 2.2 Приклад ROC-кривої для моделі виявлення фішингу

Особливості використання ML у задачах фішинг-детекції

Застосування машинного навчання виявляє ряд особливостей:

Нерівномірності даних – фішингових сайтів набагато менше ніж легітимних.

Швидкість оновлення – нові фішингові домени з'являються щодня.

Уникнення розпізнавання – зловмисники змінюють URL- адреси, використовують редиректи, маскування тощо.

2.2 Формування ознак на основі URL-адрес

Виявлення фішингових сайтів на основі аналізу URL-адрес є одним з найважливіших напрямків у галузі кібербезпеки. Це обумовлено тим, що адреса часто є єдиною доступною інформацією до фактичного завантаження вебсторінки. Саме тому попередній аналіз структури URL дозволяє ефективно виявляти потенційно шкідливі ресурси без необхідності звернення до вмісту сторінки. Розглянемо основні групи ознак, які можуть бути витягнуті з URL-адрес, а також методи їхнього формування та приклади реалізації.[4]

		Паращук М.А			ККІКБ.КР.Б – 125 – 15 – ПЗ	Арк.
		Маєвський О.В				
Змн.	Арк.	№ докум.	Підпис	Дата		24

Загальні характеристики URL

URL (Uniform Resource Locator) — це стандартизований спосіб вказування розташування ресурсу в Інтернеті. Типовий URL містить такі складові: протокол (http або https), доменне ім'я (наприклад, google.com), шлях (наприклад, /login/index.html), параметри запиту (?id=123), фрагменти (#section1).

Аналіз структури URL дозволяє виділити велику кількість характеристик, які можуть бути інформативними з точки зору фішингового ризику. В основному ознаки поділяють на такі групи:

1. Символічні та статистичні ознаки (довжина, кількість символів, піддомени тощо).
2. Логічні ознаки (наявність IP, специфічні символи, підозрілі слова).
3. Семантичні (схожість із брендами, наявність гомогліфів).
4. Ознаки скорочення адрес.

Символічні та статистичні ознаки

Ці ознаки є одними з найпростіших для обчислення та дуже часто зустрічаються в роботах виявлення фішингу. [7]

До них належать:

1. Довжина URL (кількість символів)
2. Кількість піддоменів.
3. Кількість спеціальних символів.
4. Кількість цифр, букв або їх співвідношення.
5. Глибина шляху.

Фішингові URL зазвичай є довшими, мають більше піддоменів, містять багато цифр і спеціальних символів, що використовується для імітації справжніх сайтів.

Логічні ознаки

До логічних ознак належать такі, що визначають наявність підозрілих елементів у структурі URL:

1. Наявність IP-адреси замість домену.
2. Наявність символу @, який дозволяє приховати справжній домен.

		Паращук М.А			ККІКБ.КР.Б – 125 – 15 – ПЗ	Арк.
		Маєвський О.В				
Змн.	Арк.	№ докум.	Підпис	Дата		25

3. Наявність скорочених URL(bit.ly, tinyurl.com тощо).
4. Початок з http(а не з https).
5. Підозрілі ключові слова login, secure, account, update.

Семантичні ознаки

Фішингові адреси часто імітують легітимні доменні імена. Для цього застосовуються такі підходи:

1. Використання слів, що схожі на бренди(раура1 замість раурал).
2. Використання гомогліфів (наприклад g00gle).
3. Візуально схожі символи з іншим алфавітів.

Для оцінки схожості використовують метрики: відстань Левенштейна, cosine, similarity.

Ознаки скорочених адрес

Скорочені посилання часто використовуються зловмисниками для приховування реального розміщення сайту. Тому виявлення таких сервісів у доменному імені є важливою ознакою.

У таблиці 2.2 було наведено приклад ознак витягнутих із URL-адрес.

Таблиця 2.2

Приклад ознак, витягнутих із URL-адрес

Ознака	Опис
URLLength	Загальна довжина URL
NoOfHyphenInURL	Кількість дефісів у URL
IsDomainIP	Чи є домен IP-адресою
ContainsBrandKeyword	Чи містить ключові слова
EntropyOfURL	Ентропія (міра випадковості символів)
MinLevenshteinToBrand	Мін. відстань Левенштейна до бренду

Приклад функції обробки ознак

На рисунку 2.3 зображено частину функції яка була використана для витягування ознак з URL-адрес для створення датасету, а також для використання в подальших діях.

```
def extract_features(url):
    features = {}

    parsed = urlparse(url)
    hostname = parsed.hostname or ""

    # ВІДПРАВНА ОБРОБКА WWW
    if hostname.startswith("www."):
        hostname = hostname[4:] # Видаляємо www.

    path = parsed.path or ""
    query = parsed.query or ""
    url_length = len(url)

    # Розбиття домену
    domain_parts = hostname.split('.') if hostname else []
    subdomains = domain_parts[:-2] if len(domain_parts) > 2 else []
    domain = domain_parts[-2] if len(domain_parts) > 1 else ""
    tld = domain_parts[-1] if len(domain_parts) > 0 else ""

    features['URLLength'] = url_length
    features['DomainLength'] = len(domain)
    features['TLDLength'] = len(tld)
    features['NoOfSubDomain'] = len(subdomains)
    features['IsHTTPS'] = int(parsed.scheme == 'https')
    features['IsDomainIP'] = int(bool(re.match(r"^\d{1,3}(\.\d{1,3}){3}$", hostname)))

    features['NoOfLettersInURL'] = sum(c.isalpha() for c in url)
    features['LetterRatioInURL'] = features['NoOfLettersInURL'] / url_length
    features['NoOfDigitsInURL'] = sum(c.isdigit() for c in url)
    features['DigitRatioInURL'] = features['NoOfDigitsInURL'] / url_length
    features['DigitLetterRatio'] = features['NoOfDigitsInURL'] / (features['NoOfLettersInURL'] + 1)

    features['NoOfHyphenInURL'] = url.count('-')
    features['NoOfQMarkInURL'] = url.count('?')
    features['NoOfAmpersandInURL'] = url.count('&')
    features['NoOfEqualsInURL'] = url.count('=')
    features['NoOfAtSymbol'] = url.count('@')
```

Рисунок 2.3 Функція витягування ознак з URL-адрес

Частотний аналіз ознак

Розглянемо умовну таблицю частоти появи ознак у фішингу і легітимний сайтах.

Таблиця 2.3

Ознака	Фішингові (%)	Безпечні (%)
IP у домені	31%	1%
Символ @	22%	0.3%
Довжина URL >75 символів	63%	5%

Додаткові ознаки для підвищення точності:

1. Ентропія: вираховується як міра випадкових символів. Використовується для виявлення автоматично згенерованих адрес.
2. Гомогліфи: виявлення символів з іншим алфавітів, які візуально схожі на латинські.
3. Схожість із брендами: використання відомих назв і піддоменах або шляхах. Обраховується за допомогою метрик схожості.

Ознаки, що витягуються з URL, є першим і найважливішим рівнем перевірки при виявленні фішингових сайтів. Вони мають низьку вартість обчислення, не потребують доступу до HTML-контенту та можуть бути використані для попередньої фільтрації.

2.3 Аналіз HTML-контенту вебсторінок

У той час як аналіз URL-адреси є швидким способом первинної перевірки вебресурсу, сам по собі він не завжди є достатнім для точного виявлення фішингових сайтів. Усе більше зломисників вдаються до реєстрації ‘нешкідливих’ доменів, структура яких не викликає підозр, але при цьому вміст сторінки є шкідливим або імітує відомі сайти. Саме тому на другому етапі роботи системи виявлення фішингу доцільно здійснювати аналіз HTML-контенту сторінки, який дозволяє виявити фішингові ознаки вже у візуальному та функціональному наповненні ресурсу.

HTML (HyperText Markup Language) є основною мовою розмітки вебсторінки, і завдяки аналізу цього коду можна зчитати структуру сайту, форми введення, підключені скрипти, використані стилі та інші елементи, що допомагають виявити потенційну небезпеку.[17]

Основні цілі HTML-аналізу:

1. Виявлення фішингових форм входу, які імітують логіни банків, соцмереж тощо.
2. Визначення кількості та типів скриптів.
3. Аналіз зовнішніх посилань та вставок.

		Паращук М.А			ККІКБ.КР.Б – 125 – 15 – ПЗ	Арк.
		Маєвський О.В				28
Змн.	Арк.	№ докум.	Підпис	Дата		

4. Виявлення підозрілих слів і конструкцій.

5. Оцінка структури сайту на предмет імітації легітимних ресурсів.

Елементи HTML, які найчастіше використовуються у фішингових сайтах

Зловмисники використовують HTML-код для створення копій реальних сайтів. Це можуть бути точні копії інтерфейсу банку, системи авторизації в соцмережах тощо. При цьому найчастіше використовують:

1. Форми входу (<form>) – майже всі фішингові сайти мають форму, яка просить ввести логін, пароль, іноді навіть банківську інформацію.

2. Теги <script> - містять JavaScript-код, який може збирати введену інформацію переспрямовувати користувача, ховати справжнє призначення форми.

3. Посилання () – іноді ведуть на сторонні ресурси або дублюють вигляд реальних адрес, змінюючи одну літеру.

4. Зовнішні ресурси – фішингові сторінки можуть завантажувати контент з інших серверів, зокрема, з серверів зловмисників.

5. CSS-імітації – копіюють стиль оригінально сайту.

Таблиця 2.4

Основні HTML-елементи, пов'язані з фішинговою активністю

Елемент	Призначення	Типовий приклад використання
<form>	Введення облікових даних	Імітація логіна до банку
<script>	Виконання JS-логіки	Збір даних, редиректи
<iframe>	Завантаження стороннього вмісту	Вбудоване фішингове вікно
	Візуальна імітація логотипів	Логотип Google, Facebook
<a href>	Посилання на зовнішній сайт	Перенаправлення на фейк

Методи обробки HTML-коду

Для аналізу HTML-контенту зазвичай застосовуються спеціалізовані бібліотеки такі як BeautifulSoup, lxml, або Selenium (для динамічного завантаження). Найбільш ефективним підходом є попередній парсинг коду з витягненням з усієї важливих елементів.

		Паращук М.А			ККІКБ.КР.Б – 125 – 15 – ПЗ	Арк.
		Маєвський О.В				29
Змн.	Арк.	№ докум.	Підпис	Дата		

Ознаки які витягуються з HTML-контенту

У межах реалізації проєкту не проводилось навчання окремої моделі на основі HTML-коду. Натомість був реалізований механізм парсингу вебсторінки з метою витягування додаткових ознак, які потім складуть оцінку небезпеки і дають відповідь разом з моделью. Це допомагає підвищити точність класифікацій, особливо у випадках, коли URL виглядає легітимно, але сама сторінка має фішингові ознаки.

Витягування ознак реалізовано за допомогою бібліотеки **BeautifulSoup**, яка дозволяє аналізувати структуру HTML-документів.

Приклад витягнутих ознак

1. Кількість тегів `<form>` - вказує на наявність форм введення, які часто використовуються у фішингових атаках.
2. Кількість `<input>` типу `password` або `text` – свідчить про можливу наявність форм логіну.
3. Чи є `<iframe>` на сторінці — часто використовується для вбудовування підроблених форм.
4. Кількість скриптів (`<script>`) — багато скриптів можуть вказувати на спробу приховати справжню поведінку сторінки.
5. Наявність підозрілих слів у тексті сторінки — "login", "verify", "bank", "secure", тощо.
6. Кількість зовнішніх ресурсів — якщо ресурс завантажує багато елементів з інших доменів, це може бути підозрілим

На рисунку 2.4 та на рисунку 2.5 можемо побачити фрагмент функції яка витягує ознаки, передає в функцію яка збирає оцінку і передає кількість набраних ознак для оцінювання на безпечність.

		Паращук М.А			ККІКБ.КР.Б – 125 – 15 – ПЗ	Арк.
		Маєвський О.В				30
Змн.	Арк.	№ докум.	Підпис	Дата		

```
def extract_html_features(html, base_domain=None):
    soup = BeautifulSoup(html, 'html.parser')
    features = {}

    # Forms
    forms = soup.find_all('form')
    features['num_forms'] = len(forms)
    features['num_password_forms'] = sum(1 for f in forms if f.find('input', {'type': 'password'}) is not None)
    features['num_password_inputs'] = len(soup.find_all('input', {'type': 'password'}))

    # Iframes
    features['num_iframes'] = len(soup.find_all('iframe'))

    # Scripts
    scripts = soup.find_all('script')
    features['num_scripts'] = len(scripts)
    features['num_external_scripts'] = sum(1 for s in scripts if s.get('src') is not None)
    features['external_script_ratio'] = (features['num_external_scripts'] / features['num_scripts']) if features['num_scripts'] > 0 else 0

    # Stylesheets
    stylesheets = soup.find_all('link', rel='stylesheet')
    features['num_external_stylesheets'] = sum(1 for s in stylesheets if s.get('href') and s['href'].startswith('http'))

    # Images
    images = soup.find_all('img')
    features['num_external_images'] = sum(1 for img in images if img.get('src') and img['src'].startswith('http'))

    # Meta refresh
    features['has_meta_refresh'] = int(soup.find('meta', attrs={'http-equiv': 'refresh'}) is not None)
```

Рисунок 2.4 Витяг елементів із HTML за допомогою BeautifulSoup

```
def analyze_html_features(features):
    score = 0

    if features['num_password_forms'] > 0:
        score += 3
    if features['num_iframes'] > 2:
        score += 2
    if features['num_external_scripts'] > 3 or features['external_script_ratio'] > 0.6:
        score += 2
    if features['has_meta_refresh']:
        score += 3
    if features['text_to_html_ratio'] < 0.2:
        score += 1
    if features['num_password_inputs'] > features['num_password_forms']:
        score += 1
    if features['num_links_with_ip'] > 0:
        score += 3
    if features['num_external_stylesheets'] > 5:
        score += 1
    if features['num_external_images'] > 5:
        score += 1
    if features['use_suspicious_words']:
        score += 2
    if features['num_js_events'] > 5:
        score += 2
    if features['num_hidden_elements'] > 5:
        score += 1
    if not features['has_legal_info']:
        score += 1

    return score >= 6
```

Рисунок 2.5 Збирання оцінки для результату

У результаті ці ознаки використовуються разом із ознаками URL-адреси, що приймає остаточне рішення щодо того, чи є сайт безпечним.

HTML-парсинг є важливою допоміжною складовою системи виявлення фішингових сайтів. У межах цієї роботи він не використовувався як окремий об'єкт для моделювання, але дозволив витягнути ключові структурні та текстові характеристики сторінки, які передаються до основної моделі машинного

		Паращук М.А			ККІКБ.КР.Б – 125 – 15 – ПЗ	Арк.
		Маєвський О.В				
Змн.	Арк.	№ докум.	Підпис	Дата		31

навчання. Такий підхід дозволяє зробити класифікацію більш надійною і контекстною, особливо у випадках, коли фішинговий сайт замаскований під легітимний.

2.4 Побудова та навчання моделі на основі фреймворку Keras

Розглянемо процес створення, навчання та оптимізації нейронної мережі, яка використовується в системі виявлення фішингових сайтів.

Для реалізації моделі було використано високорівневий фреймворк Keras, що працює поверх бібліотеки TensorFlow. Такий вибір обумовлений простотою, гнучкістю та широкими можливостями Keras для швидкої побудови нейронних мереж та експериментування з архітектурами.[9,10]

Обґрунтування вибору Keras

Фреймворк Keras забезпечує інтуїтивно зрозумілий синтаксис, який дозволяє зосередитись безпосередньо на архітектурі моделі та логіці обробки даних.

Основними перевагами цього фреймворку є:

1. Простота в реалізації складних архітектур.
2. Наявність великої кількості готових шарів та функції активацій.
3. Можливість легкої інтеграції з іншими бібліотеками Python.
4. Активна спільнота та велика кількість навчальних ресурсів.

Вибір архітектури моделі

Було вирішено використовувати багатошарову перцептронну нейронну мережу (Multilayer Perceptron, MLP), для класифікації URL-адреси за їхніми ознаками. Цей тип мережі підходить для задач, де вхідні дані мають фіксовану кількість числових ознак.

Архітектура мережі включає такі компоненти:

1. Вхідний шар (Input Layer) з кількості нейронів, що відповідає кількості ознак.
2. Два прихованих шари (Hidden Layers) з кількості нейронів 128 та 64 відповідно.
3. Активаційна функція Relu для обох прихованих шарів.

		Паращук М.А			ККІКБ.КР.Б – 125 – 15 – ПЗ	Арк.
		Маєвський О.В				
Змн.	Арк.	№ докум.	Підпис	Дата		32

4. Вихідний шар з одним нейроном і функцією активації sigmoid для бінарної класифікації фішинг/не фішинг.

5. Використання Dropout для запобігання перенавчання.

```
# --- Побудова моделі ---
model = Sequential([
    Dense(128, activation='relu', input_shape=(X_train.shape[1],)),
    Dropout(0.3),
    Dense(64, activation='relu'),
    Dropout(0.3),
    Dense(1, activation='sigmoid')
])

model.compile(optimizer='adam',
              loss='binary_crossentropy',
              metrics=['accuracy'])
```

Рисунок 2.6 Структура моделі для нейронної мережі виявлення фішингу

Підготовка даних

Перед подачею даних у модель, вони проходять декілька обов'язкових етапів обробки:

1. Масштабування ознак за допомогою StandardScaler з бібліотеки sklearn, що нормалізує кожен ознаку до нульового середнього та одиничного стандартного відхилення.

2. Розділення датасету на тренувальну (80%) та тестову (20%) вибірки.

3. Перетворення вхідних даних до типу float32.

```
# --- Поділ на train і test з правильним збереженням пропорції класів (stratify) ---
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, stratify=y, random_state=42)

# --- Масштабування ---
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

Рисунок 2.7 масштабування ознак та розподіл датасету

Навчання моделі

Параметри навчання були такі:

1. Розмір батчу 512.

2. Кількість епох 50.

		Паращук М.А			ККІКБ.КР.Б – 125 – 15 – ПЗ	Арк.
		Маєвський О.В				
Змн.	Арк.	№ докум.	Підпис	Дата		33

3. Викорстання EarlyStopping для зупинки навчання при відсутності покращення на валідаційній вибірці протягом 5 епох.
4. Збереження найкращої моделі за допомогою ModelCheckpoint.

```
# --- Callbacks ---
early_stop = EarlyStopping(monitor='val_loss', patience=5, restore_best_weights=True)
checkpoint = ModelCheckpoint('best_model.keras', monitor='val_loss', save_best_only=True)

# --- Навчання моделі з verbose=2 для повного виводу ---
history = model.fit(
    x_train, y_train,
    validation_split=0.1,
    epochs=50,
    batch_size=512,
    callbacks=[early_stop, checkpoint],
    verbose=2
)
```

Рисунок 2.8 Навчання моделі

У таблиці 2.5 наведено основні параметри навчання нейроної мережі.

Таблиця 2.5

Основні параметри навчання нейроної мережі

Параметр	Значення
Кількість епох	50
Розмір батчу	512
Оптимізатор	Adam
Функція втрат	Binary Crossentropy
Метрика	Accuracy
Раннє зупинення	Так (patience=5)

Оцінка результату навчання

Після завершення тренування модель показала високу точність на тестовій вибірці:

Точність (accuracy) – понад 99.5%

Повнота (recall) та точність (precision) також перебували на рівні понад 98%

Для візуалізації ефективності моделі було побудовано ROC-криву.

На рисунку 2.9 можна побачити результат навчання

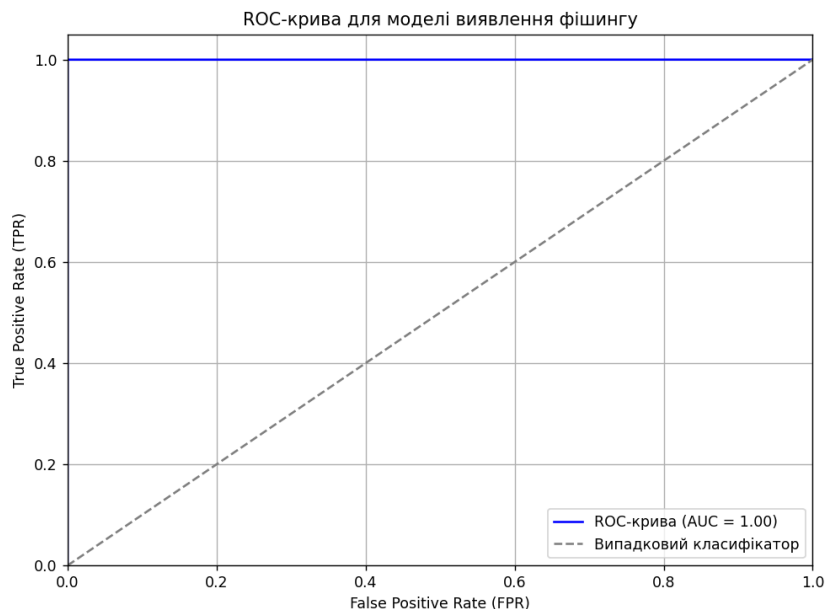


Рисунок 2.9 ROC-крива нейронної мережі для виявлення фішингу

Також на рисунку 2.10 можна побачити етапи навчання на різних епохах.

```
-----
Epoch 9:
Train Loss: 0.0011
Val Loss: 0.0004
Train Acc: 0.9999
Val Acc: 0.9999
-----
Epoch 10:
Train Loss: 0.0010
Val Loss: 0.0004
Train Acc: 0.9999
Val Acc: 0.9999
-----
Epoch 11:
Train Loss: 0.0010
Val Loss: 0.0004
Train Acc: 0.9999
Val Acc: 0.9999
-----
Epoch 12:
Train Loss: 0.0010
Val Loss: 0.0005
Train Acc: 0.9999
Val Acc: 0.9999
-----
Epoch 13:
Train Loss: 0.0010
Val Loss: 0.0004
Train Acc: 0.9999
Val Acc: 1.0000
-----
Test Loss: 0.0005
Test Accuracy: 0.9999
5597/5597
```

Рисунок 2.10 Результат навчання на епохах

Як ми бачимо на зображенні показано, що навчання моделі зупинилось після 13 епохи. Це означає, що модель навчилася максимально на скільки можливо і не могла покращуватися надалі.

Збереження та використання моделі

		Паращук М.А			ККІКБ.КР.Б – 125 – 15 – ПЗ	Арк.
		Маєвський О.В				35
Змн.	Арк.	№ докум.	Підпис	Дата		

Після тренування модель була збережена у форматі keras (best_model.keras) та інтегрована у загальну систему через Flask додаток. При запуску система завантажує модель та StandardScaler, після чого користувач може ввести URL, і система видає прогноз.[18]

На рисунку 2.11 та 2.12 можна побачити завантаження моделі і StandardScaler, а також використання в системі.

```
# Завантаження моделі та скейлера
model = load_model("best_model.keras")
scaler = joblib.load("scaler.pkl")
```

Рисунок 2.11 завантаження моделі та скейлера

```
# 8 Прогноз моделі
features_array, _ = extract_features(url)
features_scaled = scaler.transform(features_array)
prediction = model.predict(features_scaled)[0][0]
result['model_prediction'] = prediction
result['model_label'] = '⚠️ Фішинговий сайт!' if prediction >= 0.5 else '✅ Безпечний сайт.'
steps.append(('Прогноз моделі', f"{prediction:.4f} → {result['model_label']}"))
```

Рисунок 2.12 Використання моделі і видача прогнозу

Результат навчання демонструють, що правильно налаштована нейронна мережа на основі ознак URL може досягти дуже високої точності у виявлення фішингових сайтів. Інтеграція цієї моделі у вебінтерфейс дозволяє ефективно використовувати її в реальному часі.

2.5 Загальна архітектура системи

Архітектура є ключовим елементом розробки, адже саме від її продуманості залежить злагодженість роботи всіх компонентів швидкість обробки запитів, точність прийняття рішень та зручність користувача. Ретельно спроектована система дозволяє ефективно обробляти великі обсяги вхідних даних та адаптуватися до нових загроз.

Компоненти системи

Система виявлення фішингових сайтів складається з кількох взаємопов'язаних модулів, кожен з них виконує окрему функцію. Загальна структура є модульною, що дозволяє легко розширювати або оновлювати окремі компоненти без порушення логіки роботи. Основні модулі системи:

		Паращук М.А			ККІКБ.КР.Б – 125 – 15 – ПЗ	Арк.
		Маєвський О.В				
Змн.	Арк.	№ докум.	Підпис	Дата		36

1. Вебінтерфейс – надає зручну форму для введення URL-адреси користувачем та виводить результати аналізу.
2. Модуль попередньої обробки URL – нормалізує та перевіряє формат введеного посилання.
3. Модуль витягнення ознак – генерує понад 40 ознак з URL.
4. Модуль HTML-парсинг – при потребі звантажує HTML-код вебсторінки та витягує ознаки .
5. Зовнішні сервіси перевірки – API-запити до баз даних шкідливих сайтів
6. Keras-модель класифікації – нейронна мережа, яка обробляє ознаки та видає ймовірність того чи є сайт фішинговим.
7. Фінальний аналізатор – формує остаточний висновок і передає його користувачеві.

Логіка роботи системи

Повний процес виглядає наступним чином:

1. Користувач вводить підозрілий URL.
2. URL перевіряється у зовнішніх сервісах для виявлення фішингу.
3. Система проводить попередню обробку та нормалізацію URL.
4. З URL витягуються понад 40 ознак.
5. Отримані ознаки передаються у масштабованому вигляді до Keras-моделі.
6. Виконується парсинг HTML-сторінки.
7. Об'єднуються результати: моделі та парсинг HTML,
8. Формується фінальний звіт: оцінка + пояснення.

На рисунку 2.13 можна побачити загальну структуру, принцип роботи системи.

В таблиці 2.6 зображено функціональні модулі системи.

		Паращук М.А			ККІКБ.КР.Б – 125 – 15 – ПЗ	Арк.
		Маєвський О.В				
Змн.	Арк.	№ докум.	Підпис	Дата		37



Рисунок 2.13 Принцип роботи системи

Таблиця 2.6

Функціональні модулі системи

Назва модуля	Опис функцій	Вхідні дані	Вихідні дані
API-перевірки	Перевірка репутації URL у зовнішніх базах	URL	Флаги / статуси
Модуль URL-аналізу	Витяг понад 50 ознак з URL, нормалізація структури	URL	Масив ознак
Keras-модель	Класифікація сайту (фішинговий / безпечний)	Ознаки	Ймовірність
Модуль HTML-парсингу	Парсинг HTML, витяг ключових елементів	HTML	HTML-ознаки
Фінальний аналізатор	Об'єднання результатів і генерація висновку	Дані з модулів	Відповідь (рішення + обґрунтування)

Архітектура запропонованої системи є гнучкою, модульною та адаптованою до швидкої обробки запитів у реальному часі. Завдяки поєднанню методів машинного навчання, HTML-аналізу та зовнішніх API, система здатна забезпечити високий рівень точності виявлення фішингу. Кожен модуль працює автономно та взаємодіє з іншими через чітко визначені інтерфейси, що полегшує масштабування, оновлення та супровід у майбутньому.

Висновок до другого розділу

У другому розділі було розглянуто ключові етапи розробки інтелектуальної підсистеми виявлення фішингових сайтів із використанням методів машинного навчання. Перш за все, було надано теоретичне підґрунтя щодо застосування машинного навчання у задачах класифікації, а також проведено огляд основних підходів – у тому числі керованого (supervised) та некерованого (unsupervised) навчання. Особлива увага була приділена керованому навчанню, як найбільш відповідному для вирішення задачі розпізнавання фішингу.

На основі аналізу структури URL-адрес було сформовано набір ознак, що дозволяють виявити потенційно фішингові ресурси. Ретельно описано особливості побудови ознак, зокрема — довжину домену, кількість піддоменів, наявність підозрілих символів, співвідношення літер і цифр, ентропію, використання гомогліфів тощо. Також враховано додаткові евристики, як-от присутність ключових слів брендів, обфускацію, IP-домени, редиректи та інше.

Крім того, було проведено глибокий аналіз HTML-контенту вебсторінок — як додаткового рівня захисту. Розглянуто техніки парсингу HTML-коду, перевірку тегів форм, скриптів, зовнішніх посилань, структури DOM, ідентифікацію спроб приховати контент або маніпулювати елементами інтерфейсу. Описано процес витягування семантичних ознак для підвищення надійності системи.

Окрему увагу приділено побудові моделі штучної нейронної мережі у середовищі Keras. Запропонована архітектура багатошарового перцептрону (MLP) була навчена на сформованому датасеті з 40 ознак, масштабованих за допомогою StandardScaler. Навчання проходило із застосуванням технік регуляризації

		Паращук М.А			ККІКБ.КР.Б – 125 – 15 – ПЗ	Арк.
		Маєвський О.В				39
Змн.	Арк.	№ докум.	Підпис	Дата		

(Dropout), контролю перенавчання (EarlyStopping), а також валідації на тестовій вибірці. Результати тестування засвідчили високу точність — понад 99%.

У фінальному підпункті детально описано загальну архітектуру системи. Вона включає модулі витягування ознак, машинного передбачення, HTML-аналізу, перевірки через зовнішні сервіси, вебінтерфейсу користувача та модуль логування. Було показано логіку взаємодії між цими складовими, а також проаналізовано ефективність інтегрованої системи

Таким чином, у результаті розробки вдалося створити комплексну й ефективну систему, здатну ідентифікувати фішингові ресурси з високою точністю, використовуючи як класичні ознаки URL, так і глибший аналіз HTML-контенту в поєднанні з сучасними методами машинного навчання.

		Паращук М.А			ККІКБ.КР.Б – 125 – 15 – ПЗ	Арк.
		Маєвський О.В				
Змн.	Арк.	№ докум.	Підпис	Дата		40

РОЗДІЛ 3 РЕАЛІЗАЦІЯ ТА ДОСЛІДЖЕННЯ ІНТЕЛЕКТУАЛЬНОЇ ПІДСИСТЕМИ ВИЯВЛЕННЯ ФІШИНГУ

3.1 Реалізація системи виявлення фішингових сайтів

Було детально описано процес практичної реалізації програмної системи яка виконує автоматичне виявлення фішингових вебресурсів. Розроблена система поєднує етапи попередньої обробки, аналізу ознак URL-адрес, перевірки HTML-контенту та роботи нейронної моделі для прогнозу. Також реалізовано взаємодію з користувачем через вебінтерфейсу.

Загальна структура програмної системи

Розробка системи виконувалась мовою програмування Python, що забезпечує широкі можливості для реалізації алгоритмів машинного навчання, обробки тексту та побудови вебінтерфейсів. Основні бібліотеки, які були використані:

1. Keras – для завантаження та використання навченої моделі.
2. Scikit-learn – для масштабування ознак.
3. BeautifulSoup – для аналізу HTML- контенту.
4. Flask – для побудови простого вебінтерфейсу.
5. Whois, re, socket тощо – для перевірок і витягування ознак із доменів

Етапи обробки URL та витягування ознак

Ключовим етапом є побудова вектора ознак для кожного URL. Нагадаємо, що таких ознак понад 40, зокрема:

1. довжина URL, домену, глибина шляху.
2. кількість спеціальних символів (@, =, &, /, - тощо).
3. наявність IP-адреси в URL.
4. перевірка на гомогліфи.
5. ентропія та співвідношення символів.
6. перевірка на наявність ключових слів брендів тощо.

Було створено списки з відомими брендами та підозрілих слів які використовувались в функції витягування ознак.

Всі ці ознаки були реалізовані у вигляді окремої функції `extract_features(url)`.

		Паращук М.А			ККІКБ.КР.Б – 125 – 15 – ПЗ	Арк.
		Маєвський О.В				
Змн.	Арк.	№ докум.	Підпис	Дата		41

На рисунках 3.1, 3.2 можна побачити списки відомих брендів та підозрілих слів.

```
BRANDS = [
    # Міжнародні бренди
    'paypal', 'google', 'facebook', 'amazon', 'microsoft', 'apple', 'netflix', 'twitter',
    'instagram', 'linkedin', 'youtube', 'ebay', 'walmart', 'bankofamerica', 'chase',
    'citi', 'wellsfargo', 'hsbc', 'barclays', 'citibank', 'deutschebank', 'visa',
    'mastercard', 'americanexpress', 'stripe', 'square', 'alibaba', 'airbnb', 'uber',
    'spotify', 'dropbox', 'slack', 'zoom', 'adobe', 'oracle', 'salesforce', 'nike',
    'adidas', 'samsung', 'dhl', 'fedex', 'ups', 'discord', 'reddit', 'tumblr',
    'pinterest', 'snapchat', 'github', 'bitbucket', 'gitlab', 'steam', 'twitch',
    'wordpress', 'wikipedia', 'bbc', 'cnn', 'forbes', 'mozilla', 'yahoo', 'mailchimp',
    'indeed', 'glassdoor', 'pwc', 'deloitte', 'kpmg', 'ey', 'accenture', 'mcdonalds',
    'starbucks', 'ikea', 'lego', 'puma', 'huawei', 'xiaomi', 'lg', 'bosch', 'sony',
    'panasonic', 'intel', 'amd', 'nvidia', 'cisco', 'hulu', 'disney', 'paramount',
    'unilever', 'pepsico', 'coca-cola', 'bmw', 'audi', 'mercedes', 'tesla', 'ford',
    'toyota', 'volkswagen', 'icloud', 'gmail', 'drive', 'googlemaps', 'skype',
    # Крипто-платформи
    'binance', 'coinbase', 'crypto', 'metamask',
    # Месенджери та соцмережі
    'telegram', 't.me', 'viber', 'whatsapp', 'messenger', 'signal', 'wechat',
    # Українські сервіси та бренди
    'olx', 'olx.ua', 'rozetka', 'rozetkacomua', 'novaposhta', 'ukrposhta', 'meest',
    'meestexpress', 'intertop', 'eldorado', 'comfy', 'allo', 'fua', 'epicentrk',
    'foxtrot', 'varus', 'silpo', 'ekomarket', 'kabanchik', 'workua', 'rabotaua',
    'douua', 'promua', 'megogo', 'privatbank', 'monobank', 'ukrsibbank', 'otpbank',
    'alfabank', 'raiffeisenbank', 'pumb', 'ukrgasbank', 'kredo', 'oschadbank',
    'ukreximbank', 'nbu', 'privat24', 'monobankua', 'servis24', 'minfin', 'diia',
    # Державні структури
    'rada', 'rada.gov.ua', 'kmu', 'kmu.gov.ua', 'mvs.gov.ua', 'minjust.gov.ua',
    'minzdrav.gov.ua', 'gosuslugi', 'minregion', 'ukrinform', 'unian', '24tv',
    'espreso.tv', 'bbc.com.ua', 'ukraine', 'dnepr', 'kyivstar', 'vodafone',
    'lifecell', 'ukrenergo', 'kyivenergo', 'ukrgidroenergo', 'metinvest'
```

Рисунок 3.1 список брендів доменів

```
SUSPICIOUS_WORDS = {
    'login', 'secure', 'update', 'verify', 'account', 'bank', 'password', 'confirm', 'signin', 'auth',
    'security', 'reset', 'admin', 'user', 'websecr', 'session', 'access', 'client', 'confirmemail',
    'challenge', 'unlock', 'alert', 'verifyaccount', 'check', 'billing', 'invoice', 'order', 'paypal',
    'ebay', 'appleid', 'authentication', 'token', 'validate', 'activation', 'email', 'support',
    'helpdesk', 'service', 'confirmaccount', 'verification', 'payment', 'transaction', 'banking',
    'securelogin', 'secureaccess', 'securityupdate', 'resetpassword', 'accountupdate', 'signinsecure',
    'authlogin', 'accountverify', 'securityalert', 'fraudalert', 'phishing', 'loginpage', 'creditcard',
    'socialsecurity', 'ssn', 'identity', 'identification', 'personal', 'private', 'vip', 'fraud',
    'hack', 'attack', 'risk', 'blocked', 'hold', 'suspended', 'disabled', 'unauthorized', 'restricted',
    'verificationrequired', 'confirmidentity', 'recovery', 'unlockaccount', 'loginattempt', 'failed',
    'error', 'securityquestion', 'twofactor', '2fa', 'multifactor', 'mfa', 'securetoken', 'accountsecurity',
    'updateprofile', 'updateinfo', 'accountdetails', 'verifyemail', 'passwordreset', 'passwordchange',
    'accountlocked', 'loginfailed', 'accessdenied', 'authorization', 'secureconnection', 'securepage',
    'securewebsite', 'websecurity', 'accountmanagement', 'accountservice',
    # додаткові сучасні терміни
    'login-now', 'verify-now', 'your-account', 'clickhere', 'urgent', 'warning',
    'limitedoffer', 'you-have-won', 'lucky', 'claim-now'
}
```

Рисунок 3.2 Список підозрілих слів

На рисунку 3.3 можна побачити реалізацію функції з витягування ознак з URL-адреси.

		Паращук М.А			ККІКБ.КР.Б – 125 – 15 – ПЗ	Арк.
		Маєвський О.В				42
Змн.	Арк.	№ докум.	Підпис	Дата		

```
def extract_features(url):
    parsed = urlparse(url)
    hostname = parsed.hostname or ""

    # ВІДПРАВНА ОБРОБКА WWW
    if hostname.startswith("www."):
        hostname = hostname[4:] # Видаляємо WWW.

    path = parsed.path or ""
    query = parsed.query or ""
    url_length = len(url)

    # Розбиття домену
    domain_parts = hostname.split('.') if hostname else []
    subdomains = domain_parts[:-2] if len(domain_parts) > 2 else []
    domain = domain_parts[-2] if len(domain_parts) > 1 else ""
    tld = domain_parts[-1] if len(domain_parts) > 0 else ""

    features['URLLength'] = url_length
    features['DomainLength'] = len(domain)
    features['TLDDLength'] = len(tld)
    features['NoOfSubDomain'] = len(subdomains)
    features['ISHTTPS'] = int(parsed.scheme == 'https')
    features['IsDomainIP'] = int(bool(re.match(r"^\d{1,3}(\.\d{1,3}){3}$", hostname)))

    features['NoOfLettersInURL'] = sum(c.isalpha() for c in url)
    features['LetterRatioInURL'] = features['NoOfLettersInURL'] / url_length
    features['NoOfDigitsInURL'] = sum(c.isdigit() for c in url)
    features['DigitRatioInURL'] = features['NoOfDigitsInURL'] / url_length
    features['DigitLetterRatio'] = features['NoOfDigitsInURL'] / (features['NoOfLettersInURL'] + 1)

    features['NoOfHyphenInURL'] = url.count('-')
    features['NoOfQMarkInURL'] = url.count('?')
    features['NoOfAmpersandInURL'] = url.count('&')
    features['NoOfEqualsInURL'] = url.count('=')
    features['NoOfAtSymbol'] = url.count('@')
```

Рисунок 3.3 Реалізація функції витягування ознак HTML -парсинг як додатковий етап

HTML-код сторінки також аналізується. Це дозволяє виявити ознаки, які не можна отримати лише з тексту URL.

Для цього була реалізована функція `extract_html_features` з використанням бібліотеки BeautifulSoup

На рисунку 3.4 можна побачити фрагмент реалізованої функції

```
def extract_html_features(html, base_domain=None):
    soup = BeautifulSoup(html, 'html.parser')
    features = {}

    # Forms
    forms = soup.find_all('form')
    features['num_forms'] = len(forms)
    features['num_password_forms'] = sum(1 for f in forms if f.find('input', {'type': 'password'}) is not None)
    features['num_password_inputs'] = len(soup.find_all('input', {'type': 'password'}))

    # Iframes
    features['num_iframes'] = len(soup.find_all('iframe'))

    # Scripts
    scripts = soup.find_all('script')
    features['num_scripts'] = len(scripts)
    features['num_external_scripts'] = sum(1 for s in scripts if s.get('src') is not None)
    features['external_script_ratio'] = (features['num_external_scripts'] / features['num_scripts']) if features['num_scripts'] > 0 else 0

    # Stylesheets
    stylesheets = soup.find_all('link', rel='stylesheet')
    features['num_external_stylesheets'] = sum(1 for s in stylesheets if s.get('href') and s['href'].startswith('http'))

    # Images
    images = soup.find_all('img')
    features['num_external_images'] = sum(1 for img in images if img.get('src') and img['src'].startswith('http'))

    # Meta refresh
    features['has_meta_refresh'] = int(soup.find('meta', attrs={'http-equiv': 'refresh'}) is not None)

    # Text/HTML ratio
    text_length = len(soup.get_text())
    html_length = len(html)
    features['text_to_html_ratio'] = text_length / html_length if html_length > 0 else 0
```

Рис.3.4 Фрагмент функції `extract_html_features`

		Паращук М.А			ККІКБ.КР.Б – 125 – 15 – ПЗ	Арк.
		Маєвський О.В				43
Змн.	Арк.	№ докум.	Підпис	Дата		

Прогноз за допомогою нейронної мережі

Система завантажує модель (`best_model.keras`) і скалер для нормалізації ознак. Після формування вектора ознак, вони подаються на вхід мережі, яка повертає ймовірність фішингу.

На рисунку 3.5 можна побачити використані моделі для прогнозу.

```
# 8 Прогноз моделі
features_array, _ = extract_features(url)
features_scaled = scaler.transform(features_array)
prediction = model.predict(features_scaled)[0][0]
result['model_prediction'] = prediction
result['model_label'] = '⚠️ Фішинговий сайт!' if prediction >= 0.5 else '✅ Безпечний сайт.'
steps.append(('Прогноз моделі', f"{prediction:.4f} → {result['model_label']}"))
```

Рисунок 3.5 Прогноз моделі

У разі ймовірності >0.5 виводиться повідомлення про фішинг, інакше – сайт вважається безпечним. Також реалізована перевірка за Google Safe Browsing, OpenPhish, VirusTotal тощо.

На рисунку 3.6 можна побачити реалізацію функції яка перевіряє адресу в чорному списку Google Safe Browsing.

```
def check_url_in_blacklists(url, api_key):
    """
    Перевіряє URL на наявність у чорному списку Google Safe Browsing.

    Повертає:
    | (is_blacklisted: bool, details: str)
    """
    api_url = f"https://safebrowsing.googleapis.com/v4/threatMatches:find?key={api_key}"
    payload = {
        "client": {
            "clientId": "phishing-detector",
            "clientVersion": "1.0"
        },
        "threatInfo": {
            "threatTypes": ["MALWARE", "SOCIAL_ENGINEERING", "UNWANTED_SOFTWARE"],
            "platformTypes": ["ANY_PLATFORM"],
            "threatEntryTypes": ["URL"],
            "threatEntries": [{"url": url}]
        }
    }

    try:
        response = requests.post(api_url, json=payload)
        response.raise_for_status()
        result = response.json()

        if "matches" in result:
            return True, "⚠️ Виявлено збіг у Google Safe Browsing!(чорний список фішингу)"
        else:
            return False, "✅ Не знайдено у Google Safe Browsing.(чорний список фішингу)"

    except Exception as e:
        return False, f"⚠️ Помилка при перевірці Google Safe Browsing: {e}"
```

Рисунок 3.6 Функція перевірки у Google Safe Browsing

Перевірка виконується завдяки запиту чере API-ключ який було отримано впроцесі розробки системи.

		Паращук М.А			ККІКБ.КР.Б – 125 – 15 – ПЗ	Арк.
		Маєвський О.В				44
Змн.	Арк.	№ докум.	Підпис	Дата		

На рисунку 3.7 можна побачити реалізацію функції перевірки в OpenPhish, також чорний список.

Функція реалізована таким чином, що завантажується відкритий файл з фішинговими аدرسми і перевіряє чи є збіг.

```
def is_in_openphish(url):
    """
    Перевіряє, чи є URL у списку OpenPhish.
    """
    try:
        if not os.path.exists(OPENPHISH_FILE):
            return False
        with open(OPENPHISH_FILE, "r", encoding="utf-8") as f:
            lines = f.read().splitlines()
            return any(url.strip().lower() in line.lower() for line in lines)
    except Exception as e:
        print(f"❌ Помилка при перевірці OpenPhish: {e}")
        return False
```

Рисунок 3.7 Функція перевірки у OpenPhish

Також була реалізовано перевірки адреси через популярний сервіс VirusTotal, який є надійним для перевірки на шкідливі елементи. Перевірка здійснюється через запит API-ключем який було отримано в процесі розробки системи.

На рисунку 3.8 можемо побачити реалізацію функції перевірки через VirusTotal.

```
def check_url_in_virustotal(url, api_key):
    headers = {
        "x-apikey": api_key
    }
    api_base = "https://www.virustotal.com/api/v3/urls"

    try:
        # Кодування URL
        url_bytes = url.encode()
        url_id = base64.urlsafe_b64encode(url_bytes).decode().strip("=")

        # GET-запит на перевірку
        response = requests.get(f"{api_base}/{url_id}", headers=headers)

        if response.status_code == 404:
            # Якщо URL ще не сканувався – надсилаємо POST
            post_response = requests.post(api_base, headers=headers, data={"url": url})
            if post_response.status_code not in [200, 201]:
                return False, f"VirusTotal POST error: {post_response.status_code}"
            return False, "🕒 URL надіслано в VirusTotal на перевірку. Результат буде пізніше."

        elif response.status_code == 200:
            data = response.json()
            stats = data["data"]["attributes"]["last_analysis_stats"]
            malicious = stats.get("malicious", 0)
            suspicious = stats.get("suspicious", 0)

            if malicious > 0 or suspicious > 0:
                return True, f"🔴 VirusTotal: {malicious} шкідливих, {suspicious} підозрілих."
            else:
                return False, "✅ VirusTotal: Не виявлено фішингових ознак."

        else:
            return False, f"❌ VirusTotal error: {response.status_code}"

    except Exception as e:
        return False, f"❌ Помилка при перевірці VirusTotal: {e}"
```

Рисунок 3.8 Функція перевірки у VirusTotal

		Паращук М.А			ККІКБ.КР.Б – 125 – 15 – ПЗ	Арк.
		Маєвський О.В				
Змн.	Арк.	№ докум.	Підпис	Дата		45

Також було розроблено функції перевірки надійності домену. Це обумовлено тим, що якщо домен надійний і визначений як безпечний фішинг мінімальний, майже не можливий.

На рисунку 3.9 можемо побачити реалізацію функції перевірки надійності домену у списку Cisco Umbrella. Цей список є одним з надійніших для перевірки домену, також з цього списку були взяті адреси для тренування моделі на визначені безпечних сайтів.

```
def is_in_umbrella_whitelist(url):
    """
    Перевіряє, чи домен URL є у списку безпечних доменів Cisco Umbrella.
    """
    try:
        extracted = tldextract.extract(url)
        main_domain = f"{extracted.domain}.{extracted.suffix}".lower()
        return main_domain in UMBRELLA_SAFE_DOMAINS
    except Exception as e:
        print(f"✗ Помилка при перевірці Cisco Umbrella: {e}")
        return False
```

Рисунок 3.9 Функція перевірки надійності домену

Таким же чином було реалізовано функцію перевірки через ще один список надійності доменів, а також перевірку білого списку, який був розроблений самостійно. Взято і перевірено надійні домени які могли не потрапити в великі списки.

Також була реалізована функція перевірки WHOIS даних, тобто коли була створена адреса, дані про власника, ця перевірка має важливий вплив, якщо домен свіжий, а дані про власника не доступні, це може свідчити про небезпеку.

На рисунку 3.10 можемо побачити реалізацію цієї функції

```
def check_whois_info(url):
    try:
        domain_info = whois.whois(url)

        creation_date = domain_info.creation_date
        registrar = domain_info.registrar
        name = domain_info.name

        if isinstance(creation_date, list):
            creation_date = creation_date[0]

        domain_age_days = (datetime.now() - creation_date).days if creation_date else None

        messages = []

        messages.append(f"📅 Дата створення: {creation_date}" if creation_date else "✗ Дата створення не вказана")
        messages.append(f"📅 Вік домену: {domain_age_days} днів" if domain_age_days else "⚠ Не вдалося визначити вік")
        messages.append(f"🏢 Регистратор: {registrar}" if registrar else "✗ Регистратор не вказаний")
        messages.append(f"👤 Ім'я власника: {name}" if name else "👤 Ім'я власника не вказано або приховано")

        # Ризики
        if domain_age_days is not None and domain_age_days < 30:
            messages.append("⚠ Домен дуже новий. Є ризик фішингу.")
        if not name:
            messages.append("⚠ Дані власника приховані.")

        return '\n'.join(messages)

    except Exception as e:
        return f"✗ Помилка WHOIS-перевірки: {e}"
```

		Паращук М.А			ККІКБ.КР.Б – 125 – 15 – ПЗ	Арк.
		Маєвський О.В				
Змн.	Арк.	№ докум.	Підпис	Дата		46

Рисунок 3.10 Перевірка WHOIS-даних

На рисунку 3.11 наведено результат роботи системи в консолі

```

Введіть URL (або 'exit'): http://xn--google-43d.com
Аналізуємо: http://xn--google-43d.com

Google Safe Browsing:
✓ Не знайдено у Google Safe Browsing.(чорний список фішингу)
Перевірка у OpenPhish:
✓ URL не знайдено в OpenPhish.

Перевірка через VirusTotal:
✓ VirusTotal: Не виявлено фішингових ознак.
Cisco Umbrella:
✗ Домен відсутній у Cisco Umbrella Top 1M.

Majestic Million:
✗ Домен відсутній у базі Majestic Million.

Власний білий список:
Основний домен (через PSL): xn--google-43d.com
✗ Домен відсутній у білому списку.

WHOIS інформація:

Прогноз моделі (Keras):
C:\Users\ASUS\AppData\Roaming\Python\Python312\site-packages\sklearn\utils\
re names
warnings.warn(
1/1 ██████████ 0s 191ms/step
Ймовірність фішингу: 0.9998
⚠ Сайт підозрюється у фішингу. Виконуємо HTML-аналіз...

✗ Помилка при завантаженні сторінки: HTTPConnectionPool(host='xn--google-43d.com'):
Connection object at 0x000002A3BE62EA20>: Failed to resolve 'xn--google-43d.com'
✗ Не вдалося отримати HTML для аналізу. Рекомендується обережність.

Введіть URL (або 'exit'): https://zuxmzn.click/ssl/

```

Рисунок 3.11 Результат роботи програми в консолі

На рисунку можна побачити що дана URL-адреса пройшла всі перевірки перед потраплянням в модель. В цих перевірках вона не була замічена. Тому адреса подається в модель для аналізу і парсингу сторінки. Можна побачити що модель оцінила дану адресу в 99.9 відсотків що це фішинг. Парсинг не вдалось виконати так як цієї сторінки не існує, але за даними URL можна бачити що така адреса є небезпечною.

Система здатна обробляти URL-адресу, формувати числові ознаки, за потреби аналізувати HTML-код, подавати ці дані на вхід моделі та формувати остаточний висновок. Усі етапи реалізовано у Python з використанням

		Паращук М.А			ККІКБ.КР.Б – 125 – 15 – ПЗ	Арк.
		Маєвський О.В				
Змн.	Арк.	№ докум.	Підпис	Дата		47

популярних бібліотек, що забезпечує гнучкість і простоту в подальшій модифікації.

3.2 Розробка вебінтерфейсу користувача

Однією з ключових складових розробленої інтелектуальної підсистеми є зручний та функціональний вебінтерфейс. Такий інтерфейс надає можливість користувачеві взаємодіяти з системою в реальному часі, перевіряти URL-адреси, переглядати результати класифікації та отримувати пояснення щодо виявлення ризиків.

Для реалізації вебінтерфейсу було обрано мікрофреймворк Flask, який забезпечує просту, але потужно основу для створення вебзастосунків на мові програмування Python.

Вибір Flask як технології для реалізації інтерфейсу

Flask – це легкий вебфреймворк, що не нав'язує жорсткої структури, як у випадку Django, але дозволяє гнучко та швидко створювати як прості так і розширені вебдодатки. Основними перевагами Flask, які зумовили його вибір для реалізації цього проєкту, є:

1. Простота у використанні – зрозумілий синтаксис і мінімалістичний підхід
2. Інтеграція з Python-кодом – легке об'єднання моделі машинного навчання з бекендом.
3. Можливість використання шаблонів HTML/CSS.
4. Гнучкість структури проєкту, що особливо важливо при розробці науково-дослідних і студенських проєктів.
5. Підтримка REST API – у майбутньому систему можна легко масштабувати або інтегрувати з іншими сервісами.

Структура вебзастосунку

Застосунок має наступну структуру:

1. app.py – головний файл, що містить логіку маршрутизації та обробки запитів.
2. template/ - директорія з HTML-шаблонами

		Паращук М.А			ККІКБ.КР.Б – 125 – 15 – ПЗ	Арк.
		Маєвський О.В				
Змн.	Арк.	№ докум.	Підпис	Дата		48

3. pfishing.py – функції розроблені для системи

Основні маршрути Flask – застосунку

У Flask застосунку реалізує кілька ключових маршрутів (routes):

На рисунку 3.12 можна побачити структуру

```
@app.route('/', methods=['GET', 'POST'])
def index():
    result = {}
    steps = []
    feedback_sent = False # Прапорець для повідомлення про успішну відправку відгуку

    if request.method == 'POST':
        # Якщо це форма зворотного зв'язку
        if 'feedback_url' in request.form:
            url_fb = request.form.get('feedback_url', '').strip()
            message = request.form.get('feedback_message', '').strip()
            if url_fb and message:
                file_exists = os.path.isfile(feedback_file)
                with open(feedback_file, "a", encoding="utf-8", newline='') as f:
                    if not file_exists:
                        f.write("[Дата]Url|Причина|\n")
                    f.write(f"[{datetime.now().strftime('%Y-%m-%d %H:%M:%S')}]|{url_fb}|{message}|\n")
                feedback_sent = True

        # Якщо це форма перевірки URL
        elif 'url' in request.form:
            url = request.form['url'].strip()
            result['checked_url'] = url
```

Рисунок 3.12 ключові маршрути

Get – запит відображає порожню форму

Post запит обробляє введену URL-адресу, передає її до функції аналізу, отримує результат і відображає його користувачу.

Також на цьому рисунку можна побачити, що була створена форма зворотнього зв'язку, на випадок якщо користувач думає, що система помилилась. Користувач вводить повідомлення і результат зберігається в окремому файлі для подальшого оброблення і навчання моделі.

HTML-шаблон головної сторінки

Файл index.html створений з використанням мови шаблонів Jinja2. Інтерфейс містить поле введення URL, кнопку «Перевірити» та блок для виводу результатів. Додатково реалізовано підсвітку ризиків (наприклад, червоним кольором відображається виявлення фішингу).

На рисунку 3.13 можна побачити шаблон сторінки.

		Паращук М.А			ККІКБ.КР.Б – 125 – 15 – ПЗ	Арк.
		Маєвський О.В				
Змн.	Арк.	№ докум.	Підпис	Дата		49

```

<body>
<h1> 🔍 Перевірка URL на фішинг</h1>
<form method="post">
  <input type="text" name="url" placeholder="Введіть URL" required>
  <button type="submit">Перевірити</button>
</form>

{% if steps %}
<div class="result">
  <h2> 📋 Результати перевірки для: <code>{{ result.checked_url }}</code></h2>

  {% for step, message in steps %}
  <div class="step"> 🔹 <strong>{{ step }}:</strong> {{ message | replace('\n', '<br>') | safe }}</div>
  {% endfor %}

  {% if result.model_prediction is defined %}
  <div class="step">
    📊 <strong>Ймовірність фішингу:</strong> {{ result.model_prediction|round(4) }}
  </div>
  <div class="step" {{ 'suspicious' if result.model_prediction >= 0.5 else 'safe' }}>
    {{ result.model_label }}
  </div>
  {% endif %}
</div>

```

Рис.3.13 шаблон сторінки

На рисунку 3.14 зображено вигляд сторінки в браузері.



Рисунок 3.14 Вигляж сторінки

Взаємодія Frontend ↔ Backend

Весь цикл взаємодії користувача з системою виглядає так:

1. Користувач вводить URL на сайті.
2. Дані надсилаються до Flask-сервера.
3. Відбувається аналіз.
4. Результат відображається користувачеві.

Переваги веб інтерфейсу: зручність перевірки сайтів без запуску коду вручну, інтуїтивно зрозумілий інтерфейс, можливість подальшого розширення, підтримка адаптації під мобільні пристроїв.

Було реалізовано зручний вебінтерфейс користувача на основі Flask, який забезпечує ефективну взаємодію з розробленою системою виявлення фішингових сайтів. Інтерфейс дозволяє здійснювати перевірку URL-адрес в реальному часі, відображати результати моделі, а також візуалізувати додаткові ризик-фактори. Такий підхід робить систему не лише ефективною з точки зору алгоритмів, але й доступною та практично придатною для використання.

		Паращук М.А			ККІКБ.КР.Б – 125 – 15 – ПЗ	Арк.
		Маєвський О.В				50
Змн.	Арк.	№ докум.	Підпис	Дата		

3.3 Приклад роботи системи

Розглянемо практичні приклади функціонування розробленої системи. Метою є демонстрація всього ланцюжка обробки запиту користувача – від введення URL-адреси до виводу результату класифікації, а також допоміжних перевірок через зовнішні сервіси. Системи реалізована у вигляді вебзастосунку з інтерактивним інтерфейсом, що забезпечує простоту використання та зручність доступу для кінцевого користувача.

Загальна послідовність взаємодій:

1. Користувач відкриває вебінтерфейс.
2. Вводить URL-адресу, яку потрібно перевірити.
3. Перевірка через зовнішні сервіси((Google Safe Browsing, OpenPhish, VirusTotal, Majestic Million, WHOIS).
4. Система запускає витягування ознак, парсинг сторінки.
5. Ці дані обробляються і передаються до моделі.
6. Користувач отримує детальний звіт про ймовірність фішингу та джерела ризику.



Рисунок 3. 15 Головне меню інтерфейсу

Приклад №1-безпечний сайт.

Для прикладу було перевірено відому легітимну адресу: <https://google.com>

Після натискання кнопки перевірити система робить певний цикл:

1. Перевіряє чи був замічений домен в чорних списках. Якщо так то припиняє перевірку і повертає результат.
2. Чи є домен безпечним у списках надійності домену.
3. Якщо ніде не з'явився тоді запускається модель та парсинг.

		Паращук М.А			ККІКБ.КР.Б – 125 – 15 – ПЗ	Арк.
		Маєвський О.В				51
Змн.	Арк.	№ докум.	Підпис	Дата		

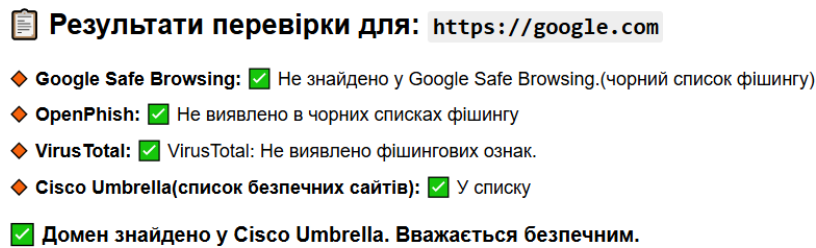


Рисунок 3.16 Результат перевірки

У даному випадку домен був замічений і списку надійних доменів тому перевірку завершено.

Приклад №2- підозрілий(імітатор)

Було введено фальшивий домен <http://xn--goog1e-43d.com>, який візуально імітує Google.

Проходить такий самий етап перевірки. Так як ця адреса не виявлена ніде вона перевіряється системою.

На рисунку 3.17 зображено результат

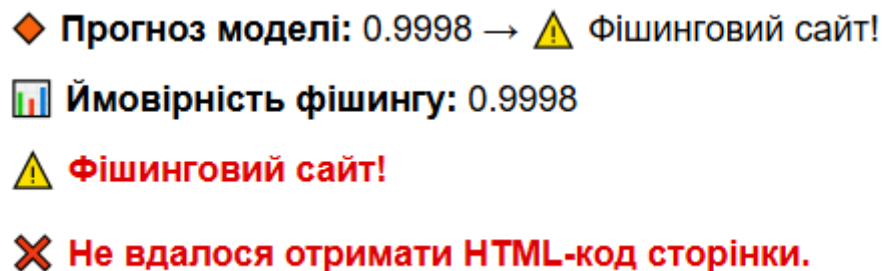


Рисунок 3.17

Чому система виявила це як фішинг?

Дана адреса містить:гомогліфи, відсутність https, та інші ознаки.

Пояснення структури відповіді системи

Система формує зведений результат у кількох блоках:

1. Зовнішні перевірки — підсумки з API (Google Safe Browsing, VirusTotal, OpenPhish, WHOIS);
2. Оцінка URL — логічний вивід моделі на основі ознак;
3. Аналіз HTML — перелік знайдених ризиків;
4. Фінальний висновок — текстове повідомлення про рівень небезпеки.

		Паращук М.А			ККІКБ.КР.Б – 125 – 15 – ПЗ	Арк.
		Маєвський О.В				
Змн.	Арк.	№ докум.	Підпис	Дата		52

Такий підхід дозволяє не тільки отримати "так/ні" відповідь, а й зрозуміти чому саме сайт був визначений як небезпечний, що важливо для користувачів, аналітиків та кіберспеціалістів

На практиці було продемонстровано роботу системи в реальних умовах. Приклади показують, що система здатна ефективно обробляти запити, аналізувати поведінку сайтів, та надавати користувачу повну картину щодо безпечності введеної URL-адреси. Інтерфейс є зручним, результат зрозумілий, а використання зовнішніх сервісів додатково підвищує точність прогнозу.

3.4 Аналіз ефективності моделі та порівняння з аналогами

Розглянемо ефективність реалізованої системи виявлення фішингу на основі нейронних мереж та її порівняння з існуючими аналогами. Метою є не лише показати досягнуті результати, а й оцінити практичну цінність розробки у контексті сучасних загроз у кіберпросторі.

Розроблена система була протестована на контрольному наборі даних, що містить URL-адреси як фішингових, так і легітимних сайтів. Важливо відзначити, що оцінювання проводилося з використанням стандартних метрик класифікації: точність (accuracy), повнота (recall), прецизійність (precision) та F1-міра.

Для кращого розуміння ефективності розробленої системи доцільно здійснити порівняння з кількома популярними онлайн-сервісами, що також використовуються для перевірки безпеки сайтів, зокрема:

1. Google Transparency Report
2. VirusTotal
3. Urlvoid

У процесі тестування було взято кілька однакових URL-адрес та проведено паралельну перевірку як через розроблену систему, так і через вище сказані аналоги.

Результати тестування було показано на наступній зображень.

Адреси було взято зі списку PhishTank

		Паращук М.А			ККІКБ.КР.Б – 125 – 15 – ПЗ	Арк.
		Маєвський О.В				53
Змн.	Арк.	№ докум.	Підпис	Дата		

Адреса <https://intersaversfin.com/payments> була взята як перший примірник.

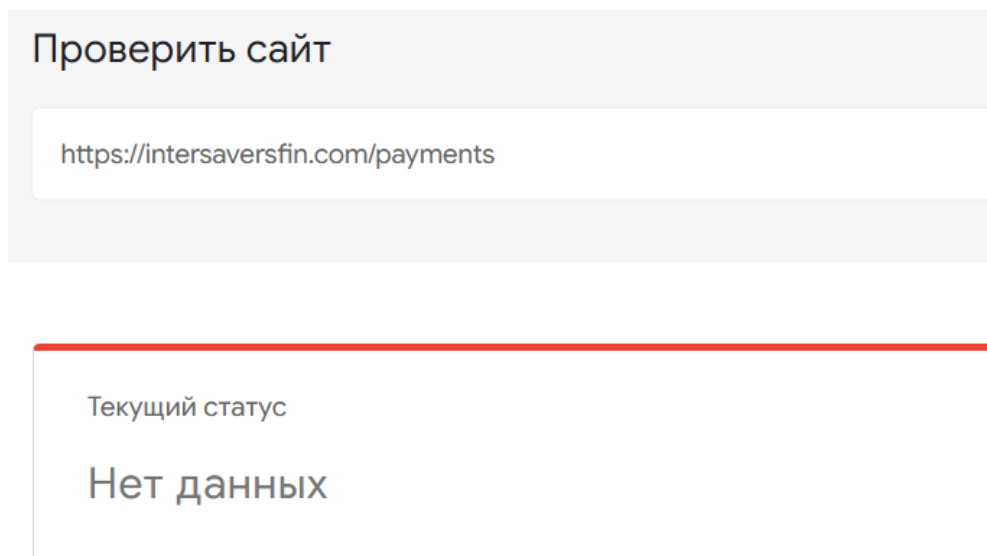


Рисунок 3.18 Результат перевірки Google Transparency Report

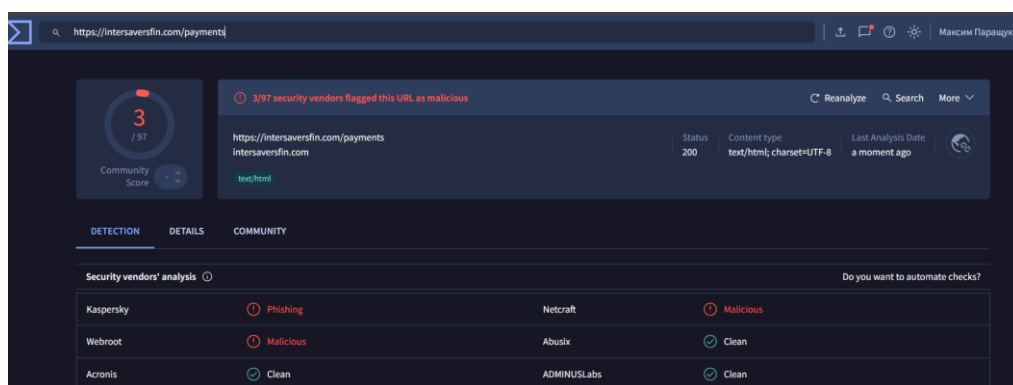


Рисунок 3.19 Результат перевірки VirusTotal

На даному зображенні можна побачити що VirusTotal попереджає про небезпеку.

		Паращук М.А			ККІКБ.КР.Б – 125 – 15 – ПЗ	Арк.
		Маєвський О.В				54
Змн.	Арк.	№ докум.	Підпис	Дата		

Report Summary

Website Address	Intersaversfin.com
Last Analysis	4 seconds ago Rescan
Detections Counts	0/39
Domain Registration	2025-05-16 1 month ago
Domain Information	WHOIS Lookup DNS Records Ping
IP Address	141.136.39.71 Find Websites IPVoid Whois
Reverse DNS	Unknown
ASN	AS47583 Hostinger International Limited
Server Location	 (GB) United Kingdom
Latitude\Longitude	53.4809 / -2.2374 Google Map
City	Manchester
Region	Manchester

Рисунок 3.20 Результат перевірки Urlvoid

 **Результати перевірки для:** <https://intersaversfin.com/payments>

- ◆ **Google Safe Browsing:**  Не знайдено у Google Safe Browsing.(чорний список фішингу)
- ◆ **OpenPhish:**  Не виявлено в чорних списках фішингу
- ◆ **Virus Total:**  URL надіслано в VirusTotal на перевірку. Результат буде пізніше.
- ◆ **Cisco Umbrella(список безпечних сайтів):**  Відсутній
- ◆ **Majestic Million(список безпечних сайтів):**  Відсутній
- ◆ **Основний домен:** intersaversfin.com
- ◆ **Білий список:**  Ні
- ◆ **WHOIS:**  Дата створення: 2025-05-16 20:12:21
-  Вік домену: 36 днів
-  Регістратор: NameCheap, Inc.
-  Ім'я власника не вказано або приховано
-  Дані власника приховані.
- ◆ **Прогноз моделі:** 1.0000 →  Фішинговий сайт!
-  Ймовірність фішингу: 1.0
-  **Фішинговий сайт!**
-  **Аналіз HTML:**
-  **HTML сторінка виглядає безпечною.**
-  **Модель вважає сайт фішинговим, але HTML виглядає безпечно. Використовуйте з обережністю.**

Рисунок 3.21 Результат розробленої системи

		Паращук М.А			ККІКБ.КР.Б – 125 – 15 – ПЗ	Арк.
		Маєвський О.В				55
Змн.	Арк.	№ докум.	Підпис	Дата		

Результати навчання мають такий висновок, дві системи не дали нічого підозрілого, інші дві показали не безпеку.

Результат розробленої системи показав, що домен створено недавно (що має перевагу для фішингу) за результатом моделі було визначено небезпеку, але HTML має безпечний вигляд, тому система рекомендує використовувати з обережністю.

Перейдено до тестування наступної адреси.

Адреса <https://www.cantupneu.online/>

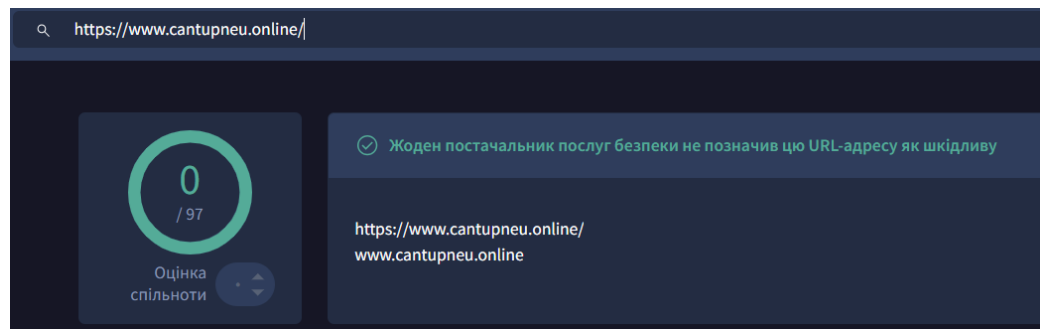


Рисунок 3.22 Результат перевірки VirusTotal

		Паращук М.А			ККІКБ.КР.Б – 125 – 15 – ПЗ	Арк.
		Маєвський О.В				56
Змн.	Арк.	№ докум.	Підпис	Дата		

Report Summary	
Website Address	Cantupneu.online
Last Analysis	4 seconds ago Rescan
Detections Counts	0/39
Domain Registration	2025-06-22 16 hours ago
Domain Information	WHOIS Lookup DNS Records Ping
IP Address	Unknown
Reverse DNS	Unknown
ASN	Unknown
Server Location	Unknown
Latitude\Longitude	Unknown
City	Unknown
Region	Unknown

Рисунок 3.23 Результат перевірки Urlvoid

Проверить сайт

https://www.cantupneu.online/

Текущий статус

✓

 Не найдено небезопасного контента.

Рисунок 3.24 Результат перевірки Google Transparency Report

Як ми бачимо інші системи вивлення фішингу показали, що посилання безпечне. Але для остаточного результату провіримо на нашій системі.

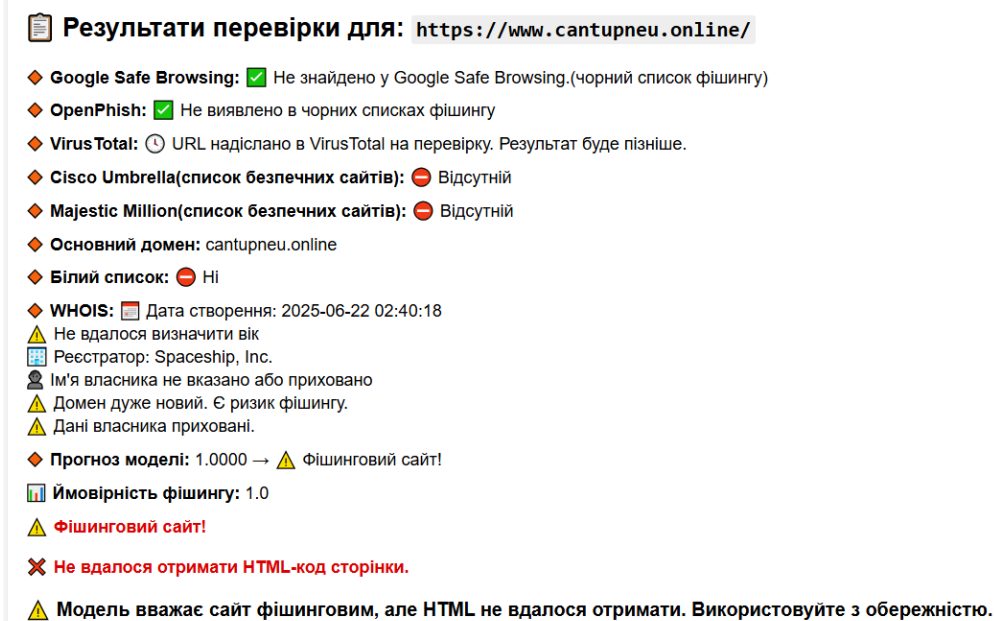


Рисунок 3.25 Результат розробленої системи

Власна розроблена система показала, що сайт небезпечний. Тобто загальний аналіз 3 до 1. З цього можна зробити висновок, що наша система помилилась, але розглянемо результат детально.

Дана адреса не була замічена в чорних списках, не має підтвердження надійності. Тепер найголовніше, домен створено лише декілька годин назад, дані власника приховано, що є підозрілим. Модель дала прогноз що це фішинг, але не вдалось отримати HTML-аналіз. Тому система попереджає використовувати з обережністю. Із-за того, що не був проведений аналіз сторінки було прийнято рішення перейти за цим посиланням і перевірити сторінку. На рисунку 3.22 можна побачити результат переходу за посиланням.

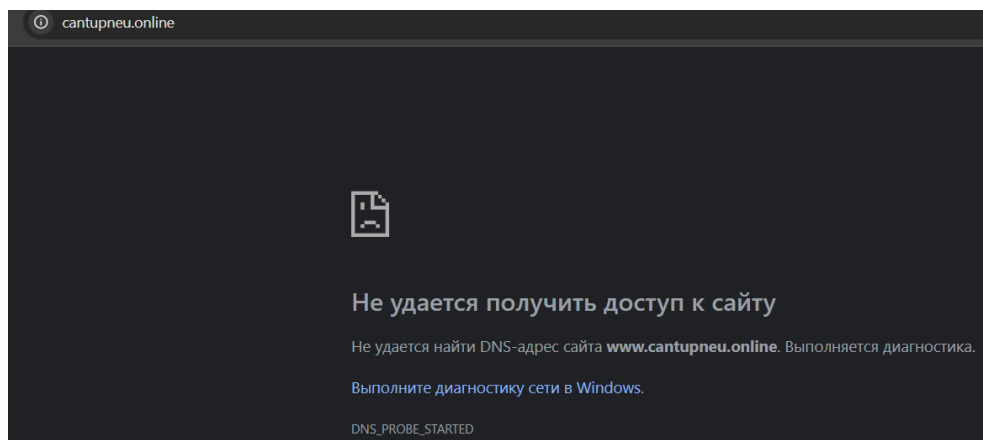


Рисунок 3.26 Перехід за посиланням

		Паращук М.А			ККІКБ.КР.Б – 125 – 15 – ПЗ	Арк.
		Маєвський О.В				
Змн.	Арк.	№ докум.	Підпис	Дата		58

На даному зображенні можна побачити, що дана сторінка не завантажується. Тому можна припустити, що результати інших сервісів не могли дати правильну відповідь.

Наступна адреса для перевірки:

[https://allegrolokalnie.pl-ofertaid282819238129236901.cfd/...](https://allegrolokalnie.pl-ofertaid282819238129236901.cfd/)

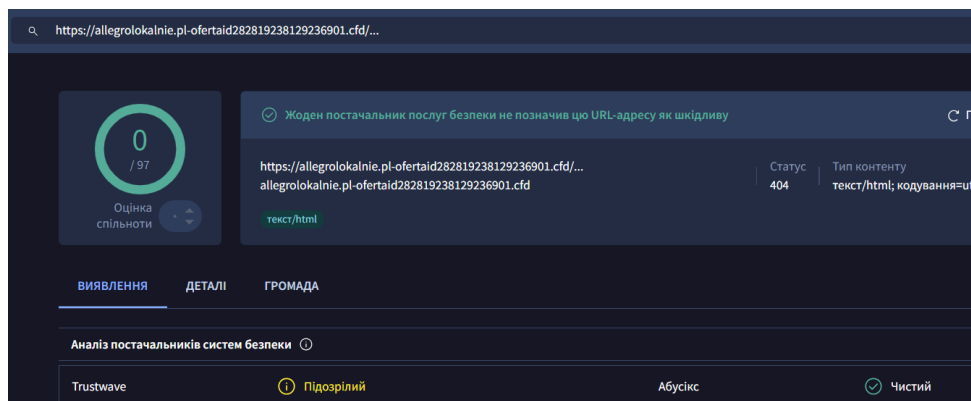


Рисунок 3.27 Результат VirusTotal

Можемо бачити, що позначила як безпечно але є підозри.

Report Summary	
Website Address	Allegrolokalnie.pl-ofertaid282819238129236901.cfd
Last Analysis	3 seconds ago Rescan
Detections Counts	0/39
Domain Registration	2025-06-22 16 hours ago
Domain Information	WHOIS Lookup DNS Records Ping
IP Address	104.21.25.100 Find Websites IPVoid Whois
Reverse DNS	Unknown
ASN	AS13335 CLOUDFLARENET
Server Location	Unknown
Latitude\Longitude	Unknown
City	Unknown
Region	Unknown

Рисунок 3.28 Результат Urlvoid

		Паращук М.А			ККІКБ.КР.Б – 125 – 15 – ПЗ	Арк.
		Маєвський О.В				59
Змн.	Арк.	№ докум.	Підпис	Дата		

Даний результат показує, що дані не відомо, але показує, що сайт безпечний

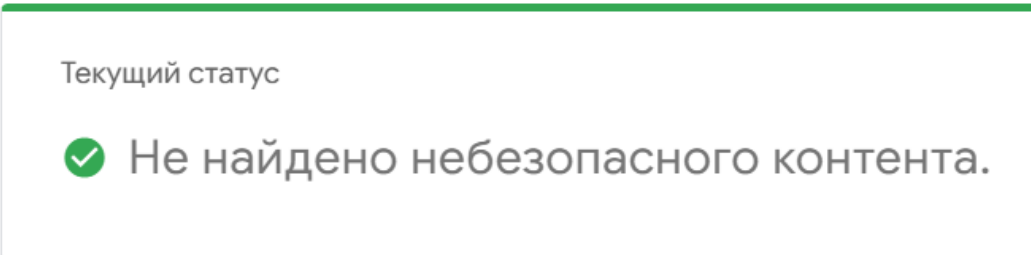
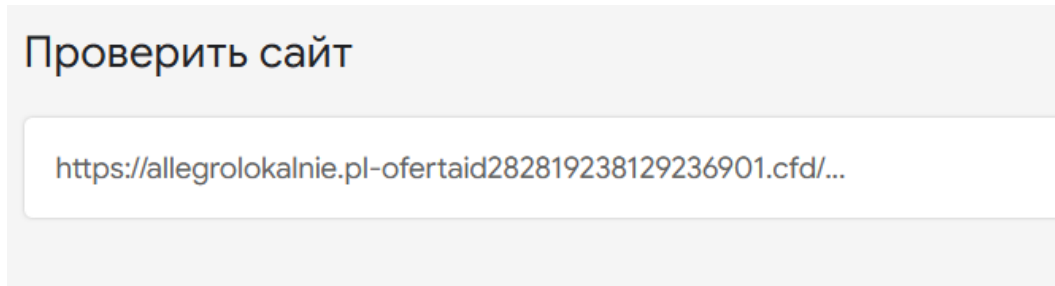


Рисунок 2.29 Результат Google Transparency Report

Данні перевірки показують, що посилання безпечне. Розглянемо, що відповідь наша розроблена система.

Результати перевірки для: <https://allegrolokalnie.pl-ofertaid282819238129236901.cfd/...>

- Google Safe Browsing: Не знайдено у Google Safe Browsing.(чорний список фішингу)
- OpenPhish: Не виявлено в чорних списках фішингу
- Virus Total: URL надіслано в VirusTotal на перевірку. Результат буде пізніше.
- Cisco Umbrella(список безпечних сайтів): Відсутній
- Majestic Million(список безпечних сайтів): Відсутній
- Основний домен: pl-ofertaid282819238129236901.cfd
- Білий список: Ні
- WHOIS: Дата створення: 2025-06-22 14:56:18
- Не вдалося визначити вік
- Регістратор: Web Commerce Communications Ltd
- Ім'я власника не вказано або приховано
- Домен дуже новий. Є ризик фішингу.
- Дані власника приховані.
- Прогноз моделі: 1.0000 → Фішинговий сайт!
- Ймовірність фішингу: 1.0
- Фішинговий сайт!
- Не вдалося отримати HTML-код сторінки.
- Модель вважає сайт фішинговим, але HTML не вдалося отримати. Використовуйте з обережністю.

Рисунок 3.30

На зображенні видно, що наша система має іншу відповідь від інших сервісів. Розглянемо детальну перевірку.

В чорних списках і надійних доменів дану адресу не було знайдено.

Основний домен є незвичним, що може свідчити про небезпеку.

Домен створено пару годин тому.(час перевірки 22.06.2025 19:00)

		Паращук М.А			ККІКБ.КР.Б – 125 – 15 – ПЗ	Арк.
		Маєвський О.В				60
Змн.	Арк.	№ докум.	Підпис	Дата		

Домен дуже новий, власник прихований, що може свідчити про фішинг.

Модель з ймовірністю 100% показав, що небезпечно, але аналіз сторінки не було отримано, тому система попереджає використовувати з обережністю.

Для більш чіткого результату перейдемо за даним посиланням.

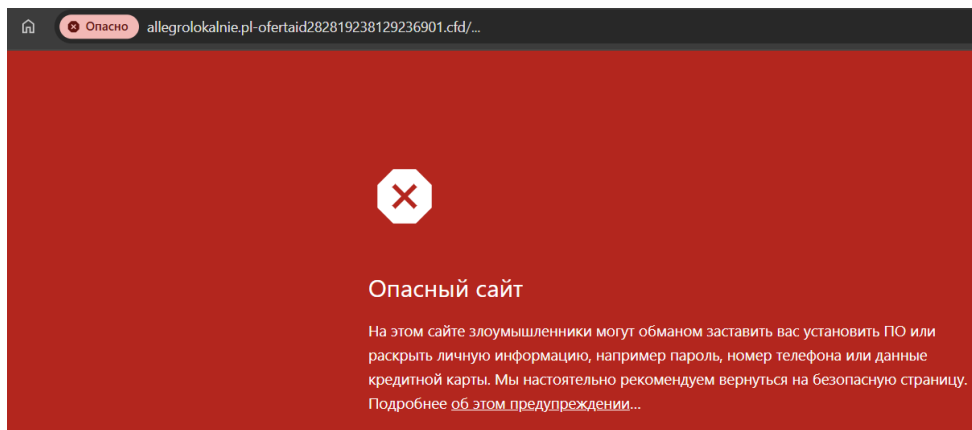


Рисунок 3.31

На цьому зображенні можна побачити, що переходить за даним посиланням є небезпечно. Тому можна зробити висновок, що наша розроблена система дала вірну відповідь.

Також для наступного тестування було створено вручну адресу. Перевіримо як ці системи проаналізують адресу якої не існує. Адреса була вигадана під небезпечну, та безпечний варіант.

Адреса для тестування <http://github8-login.com>. Дана адреса є вигадана тому деякі результати не будуть доступні.

По адресі можна зрозуміти що це підробка справжнього сайту <https://github.com>.

Розглянемо які результати покажуть системи.

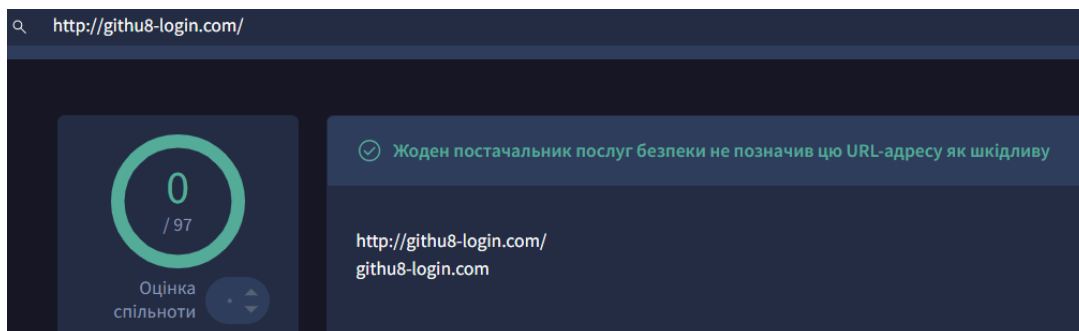


Рисунок. 3.32 Результат VirusTotal

		Паращук М.А			ККІКБ.КР.Б – 125 – 15 – ПЗ	Арк.
		Маєвський О.В				61
Змн.	Арк.	№ докум.	Підпис	Дата		

Report Summary

Website Address	Githu8-login.com
Last Analysis	4 seconds ago Rescan
Detections Counts	0/39
Domain Registration	Unknown
Domain Information	WHOIS Lookup DNS Records Ping
IP Address	Unknown
Reverse DNS	Unknown
ASN	Unknown
Server Location	Unknown
Latitude\Longitude	Unknown
City	Unknown
Region	Unknown

Рисунок 3.33 Результат urlvoid

На даній перевірці можемо побачити що дана система немає ніякої інформації про цю адресу, але визначає її безпечною.

Проверить сайт

Текущий статус

Нет данных

Рисунок 3.34 Результат Google Transparency Report

		Паращук М.А			ККІКБ.КР.Б – 125 – 15 – ПЗ	Арк.
		Маєвський О.В				
Змн.	Арк.	№ докум.	Підпис	Дата		62

 **Результати перевірки для:** `http://githu8-login.com`

◆ **Прогноз моделі:** 1.0000 →  **Фішинговий сайт!**

 **Ймовірність фішингу:** 1.0

 **Фішинговий сайт!**

 **Не вдалося отримати HTML-код сторінки.**

Рисунок 3.35

Як можемо бачити наша система з певністю повідомляє, що дана адреса фішинг.

Проаналізуючи дані результати можна сказати, що власна розроблена система може розпізнати фішинг по URL навіть якщо не існує офіційної сторінки. Тому що приклад даної адреси є наглядним прикладом фішингом на відомі бренди.

Також була вигадана адреса якої не усніє але має ознаки безпеки і повинен визначитися як безпечним.

Взявши до уваги, що існуючи системи покажуть що це безпечно, тому результат перевірки не будуть продемонстровано, лише власна система.

 **Результати перевірки для:** `https://gontel.net`

◆ **Google Safe Browsing:**  Не знайдено у Google Safe Browsing.(чорний список фішингу)

◆ **OpenPhish:**  Не виявлено в чорних списках фішингу

◆ **VirusTotal:**  URL надіслано в VirusTotal на перевірку. Результат буде пізніше.

◆ **Cisco Umbrella(список безпечних сайтів):**  Відсутній

◆ **Majestic Million(список безпечних сайтів):**  Відсутній

◆ **Основний домен:** gontel.net

◆ **Білий список:**  Ні

Рисунок 3.36Перевірка адреси

На зображені бачимо, що дана адреса не виявлена в чорних списках і в білих списках.

		Паращук М.А			ККІКБ.КР.Б – 125 – 15 – ПЗ	Арк.
		Маєвський О.В				
Змн.	Арк.	№ докум.	Підпис	Дата		63

- ◆ Прогноз моделі: 0.3563 → ☒ Безпечний сайт.
- 📊 Ймовірність фішингу: 0.3563
- ☒ Безпечний сайт.
- ✗ Не вдалося отримати HTML-код сторінки.
- ☒ Модель вважає сайт безпечним, але HTML не вдалося отримати.

Рисунок 3.37Перевірка адреси

Але наша модель дала результат, що дана адреса є безпечною.

Переваги розробленої системи.

На основі цих тестувань можна сказати, що розроблена система працює добре, та має деякі переваги над існуючими системами.

1. Інтеграція декількох рівнів перевірки
2. використання моделі машинного навчання.
3. наявність зручного інтерфейсу для взаємодії.
4. Низький час обробки запитів.

Водночас слід визнати деякі обмеження:

1. Потрібен постійний моніторинг змін в тактиках фішингових атак та оновлення моделі.

2. Система може бути вразливою до складних обхідних атак, якщо використовуються нові методи маскування URL.

Таким чином, розроблена система продемонструвала високу ефективність та точність у виявленні фішингових загроз. Завдяки поєднанню кількох підходів, вона перевершує окремі аналоги за гнучкістю, точністю та практичністю застосування.

Результати порівняння підтверджують, що система здатна конкурувати з комерційними та відкритими рішеннями у сферу кібербезпеки.

Висновок до третього розділу

У третьому розділі кваліфікаційної роботи було здійснено повну реалізацію та практичну перевірку функціонування розробленої інтелектуальної підсистеми виявлення фішингових сайтів. На основі побудованої моделі машинного навчання та сформованих ознак було реалізовано повнофункціональну систему з

		Паращук М.А			ККІКБ.КР.Б – 125 – 15 – ПЗ	Арк.
		Маєвський О.В				
Змн.	Арк.	№ докум.	Підпис	Дата		64

вебінтерфейсом, що забезпечує зручність у використанні як для технічних спеціалістів, так і для звичайних користувачів.

Розглянуто практичну реалізацію основних модулів системи: завантаження навченої моделі, масштабування вхідних даних, обробка URL-адреси, аналіз HTML-контенту, перевірки через зовнішні сервіси (Google Safe Browsing, OpenPhish, VirusTotal, WHOIS), а також побудову фінального висновку. Весь функціонал був інтегрований у зручний вебінтерфейс на Flask, що дозволяє проводити перевірки в реальному часі.

Описано розробку вебінтерфейсу користувача, включаючи логіку введення даних, обробку запитів на сервері, формування звіту та візуалізацію результатів. Була також забезпечена адаптація дизайну для різних пристроїв та користувацьких сценаріїв.

Було продемонстровано практичні приклади роботи системи на реальних прикладах URL-адрес — як безпечних, так і фішингових. Продемонстровано можливості системи щодо розпізнавання атак, які використовують гомогліфи, фальшиві домени, підроблені форми входу. Деталізовані результати показують, як поєднання машинного навчання та перевірок HTML+API дозволяє досягати високої точності у виявленні фішингових ресурсів.

Проведено аналіз ефективності моделі машинного навчання, а також порівняння розробленої системи з існуючими аналогами. Результати тестування показали, що запропонований підхід перевершує типові фільтри на базі чорних списків і статичних евристик за рахунок адаптивності та глибшого аналізу. Система показала високу точність (понад 99%) та низький рівень хибних спрацювань.

Таким чином, розроблена система може бути успішно використана як частина комплексного захисту від фішингу у корпоративному середовищі, або як окремий засіб перевірки безпечності сайтів для кінцевих користувачів. Отримані результати повністю підтверджують доцільність використання методів машинного навчання для вирішення задач кібербезпеки.

		Паращук М.А			ККІКБ.КР.Б – 125 – 15 – ПЗ	Арк.
		Маєвський О.В				65
Змн.	Арк.	№ докум.	Підпис	Дата		

ВИСНОВКИ

У результаті виконання кваліфікаційної роботи було досягнуто поставлену мету — розробка інтелектуальної підсистеми виявлення фішингових атак на основі аналізу URL-адрес, HTML-контенту та перевірки через зовнішні сервіси, з використанням методів машинного навчання. У межах дослідження реалізовано повний цикл побудови системи — від аналізу проблеми до створення програмного продукту з вебінтерфейсом, здатного працювати в реальних умовах.

Проаналізовано актуальні кіберзагрози, пов'язані з фішинговими атаками. У першому розділі було детально розглянуто природу фішингу як методу соціальної інженерії, класифікацію фішингових атак, канали їх реалізації (електронна пошта, підроблені сайти, шкідливі повідомлення), а також їхню динаміку поширення. Було показано, що фішинг залишається однією з найнебезпечніших загроз сучасного цифрового світу через свою гнучкість, швидкість та здатність обходити класичні засоби захисту.

Вивчено сучасні підходи до виявлення фішингових сайтів. Було проаналізовано методи на основі евристик, чорних списків, фільтрації по доменах, а також машинного навчання. Проведено огляд систем-аналізаторів, що використовують URL-ознаки, вміст HTML, метаінформацію WHOIS та API перевірки. Виявлено недоліки існуючих рішень — зокрема, обмежена здатність реагувати на нові домени та обхід простих евристичних фільтрів

Розроблено методику формування ознак з URL-адрес та HTML-коду. Був побудований комплекс ознак (понад 40), що включає структурні, лексичні, статистичні, ентропійні характеристики URL, а також аналіз HTML-контенту на наявність фішингових шаблонів (наприклад, форм входу, ключових слів, фреймів, редиректів). Це дозволило моделі глибше розуміти природу сайту без необхідності відкриття у браузері.

Створено модель на основі нейронних мереж із використанням фреймворку Keras. На базі багатоварової перцептронної архітектури було побудовано модель, яка була навчена на реальному датасеті із збалансованими прикладами фішингових і легітимних сайтів. Досягнуто високих показників якості: точність понад 99%,

		Паращук М.А			ККІКБ.КР.Б – 125 – 15 – ПЗ	Арк.
		Маєвський О.В				
Змн.	Арк.	№ докум.	Підпис	Дата		66

AUC > 0.99, Recall і Precision — на рівні понад 98%. Для зменшення ризику перенавчання застосовувались Dropout і EarlyStopping.

Реалізовано вебінтерфейс користувача на Flask. Створено простий та зручний інтерфейс, через який користувач може ввести URL для перевірки. Система автоматично обробляє запит, витягує ознаки, аналізує HTML, звертається до зовнішніх API та повертає остаточний висновок із поясненнями.

Проведено тестування розробленої системи та оцінено її ефективність. Система була протестована на різних прикладах, у тому числі — на відомих фішингових і легітимних сайтах. Порівняння з аналогами (UrlVoid, VirusTotal) засвідчило високу конкурентоспроможність запропонованої системи. Значною перевагою є поєднання декількох рівнів перевірки — що дозволяє виявляти фішинг навіть при зміні URL чи маскуванні контенту.

Таким чином, результати дослідження демонструють, що інтелектуальні методи виявлення фішингу мають великий потенціал для практичного застосування. Запропонована система може бути використана в рамках корпоративного моніторингу кібербезпеки, розширень для браузера, інтеграції в антивірусні рішення або як автономний вебсервіс для звичайних користувачів.

Ця робота підтверджує ефективність міждисциплінарного підходу — поєднання знань із кібербезпеки, програмування, машинного навчання та веброзробки — для створення реального інструменту боротьби з сучасними кіберзагрозами.

		Паращук М.А			ККІКБ.КР.Б – 125 – 15 – ПЗ	Арк.
		Маєвський О.В				67
Змн.	Арк.	№ докум.	Підпис	Дата		

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Chen L., Zhu R., Hou H., Wang H. A Survey of Machine Learning-Based Solutions for Phishing Website Detection. Algorithms, 2020, 13(7):171. DOI: <https://doi.org/10.3390/a13070171>
2. Barik K., Misra S., Mohan R. Web-based phishing URL detection model using deep learning optimization techniques. Personal and Ubiquitous Computing, 2025. DOI: <https://doi.org/10.1007/s41060-025-00728-9>
3. Nasr M., Venkataramakrishnan S., Jones C. Explainable Machine Learning for Phishing Feature Detection. Quality and Reliability Engineering International, 2023. DOI: <https://doi.org/10.1002/qre.3411>
4. Haq Q.E.u., Faheem M.H., Ahmad I. Detecting Phishing URLs Based on a Deep Learning Approach to Prevent Cyber-Attacks. Applied Sciences, 2024, 14(22):10086. DOI: <https://doi.org/10.3390/app142210086>
5. Al-Shamsi H., Mahmoud Q.H., Tang L. A Comprehensive Survey of AI-Enabled Phishing Attack Detection Techniques. Ad Hoc Networks, 2020. DOI: <https://doi.org/10.1007/s11235-020-00733-2>
6. Tang L., Mahmoud Q.H. A Deep Learning-Based Framework for Phishing Website Detection. IEEE Access, 2021. DOI: <https://doi.org/10.1109/ACCESS.2021.3137636>
7. Xu P. A Transformer-based Model to Detect Phishing URLs. arXiv:2109.02138 (2021).
8. Nazario J. Phishing Corpus [Online Resource]. – <https://monkey.org/~jose/phishing>
9. Keras Documentation – <https://keras.io>
10. TensorFlow Documentation – <https://www.tensorflow.org>
11. Scikit-learn Documentation – <https://scikit-learn.org>
12. OpenPhish — Phishing Intelligence Feed – <https://openphish.com>
13. Google Safe Browsing API – <https://developers.google.com/safe-browsing>
14. VirusTotal – Online Scanner – <https://www.virustotal.com>
15. Cisco Umbrella – <https://umbrella.cisco.com>

		Паращук М.А			ККІКБ.КР.Б – 125 – 15 – ПЗ	Арк.
		Маєвський О.В				68
Змн.	Арк.	№ докум.	Підпис	Дата		

16. Majestic Million Dataset – <https://majestic.com/reports/majestic-million>
17. BeautifulSoup Documentation – [https://www.crummy.com/software/BeautifulSoup/](https://www.crummy.com/software/BeautifulSoup/BeautifulSoup/)
18. Flask Documentation – <https://flask.palletsprojects.com>

		Паращук М.А			ККІКБ.КР.Б – 125 – 15 – ПЗ	Арк.
		Маєвський О.В				
Змн.	Арк.	№ докум.	Підпис	Дата		69

ДОДАТКИ

		Паращук М.А			ККІКБ.КР.Б – 125 – 15 – ПЗ	Арк.
		Маєвський О.В				70
Змн.	Арк.	№ докум.	Підпис	Дата		

Функція витягування ознак з URL

```

def extract_features(url):
    features = {}

    parsed = urlparse(url)
    hostname = parsed.hostname or ""

    # ВІДПРАВНА ОБРОБКА WWW
    if hostname.startswith("www."):
        hostname = hostname[4:] # Видаляємо www.

    path = parsed.path or ""
    query = parsed.query or ""
    url_length = len(url)

    # Розбиття домену
    domain_parts = hostname.split('.') if hostname else []
    subdomains = domain_parts[:-2] if len(domain_parts) > 2 else []
    domain = domain_parts[-2] if len(domain_parts) > 1 else ""
    tld = domain_parts[-1] if len(domain_parts) > 0 else ""

    features['URLLength'] = url_length
    features['DomainLength'] = len(domain)
    features['TLDDLength'] = len(tld)
    features['NoOfSubDomain'] = len(subdomains)
    features['IsHTTPS'] = int(parsed.scheme == 'https')
    features['IsDomainIP'] = int(bool(re.match(r"^\d{1,3}(\.\d{1,3}){3}$", hostname)))

    features['NoOfLettersInURL'] = sum(c.isalpha() for c in url)
    features['LetterRatioInURL'] = features['NoOfLettersInURL'] / url_length
    features['NoOfDigitsInURL'] = sum(c.isdigit() for c in url)
    features['DigitRatioInURL'] = features['NoOfDigitsInURL'] / url_length
    features['DigitLetterRatio'] = features['NoOfDigitsInURL'] /
(features['NoOfLettersInURL'] + 1)

    features['NoOfHyphenInURL'] = url.count('-')
    features['NoOfQMarkInURL'] = url.count('?')
    features['NoOfAmpersandInURL'] = url.count('&')
    features['NoOfEqualsInURL'] = url.count('=')
    features['NoOfAtSymbol'] = url.count('@')

    features['SpacialCharRatioInURL'] = (
        features['NoOfHyphenInURL'] +

```

		Паращук М.А			ККІКБ.КР.Б – 125 – 15 – ПЗ	Арк.
		Маєвський О.В				71
Змн.	Арк.	№ докум.	Підпис	Дата		

```

features['NoOfQMarkInURL'] +
features['NoOfAmpersandInURL'] +
features['NoOfEqualsInURL'] +
features['NoOfAtSymbol']
) / url_length

features['CharContinuationRate'] = sum(1 for i in range(1, len(url)) if url[i] == url[i-1]) / url_length
features['NoOfObfuscatedChar'] = sum(1 for c in url if ord(c) > 127)
features['ObfuscationRatio'] = features['NoOfObfuscatedChar'] / url_length

features['ContainsBrandKeyword'] = int(any(brand in url.lower() for brand in BRANDS))

features['BrandSimilarityScore'] = max(
    1 - (levenshtein_distance(domain.lower(), brand) / max(len(domain), len(brand)))
    if max(len(domain), len(brand)) > 0 else 0
    for brand in BRANDS
)

features['MinLevenshteinToBrand'] = min(
    levenshtein_distance(domain.lower(), brand)
    for brand in BRANDS
)

features['HasHomoglyphs'] = int(bool(re.search(r'^\x00-\x7F', url)))

features['DepthOfPath'] = path.count('/')
features['HasQueryParams'] = int(bool(query))
features['NumberOfQueryParams'] = query.count('&') + 1 if query else 0
features['HasAtSymbol'] = int('@' in url)
features['DoubleSlashCount'] = url.count('//')
features['PresenceOfIPVersion6'] = int(bool(re.match(r'^\[[?([a-fA-F0-9:]+)]?\$',
hostname)))
features['NumberOfDots'] = url.count('.')
features['PresenceOfRedirect'] = int('// in path)
features['RatioOfDigitsInDomain'] = sum(c.isdigit() for c in domain) / (len(domain) +
1)
features['RatioOfLettersInDomain'] = sum(c.isalpha() for c in domain) / (len(domain)
+ 1)
features['IsShortenedURL'] = int(any(short in url for short in ['bit.ly', 'goo.gl', 't.co',
'tinyurl.com']))
features['NumberOfFragments'] = url.count('#')
features['PresenceOfNonASCIIChars'] = int(any(ord(c) > 127 for c in url))
features['NumberOfUppercaseLetters'] = sum(c.isupper() for c in url)

```

		Паращук М.А			ККІКБ.КР.Б – 125 – 15 – ПЗ	Арк.
		Маєвський О.В				72
Змн.	Арк.	№ докум.	Підпис	Дата		

```

features['RatioOfUppercaseLetters'] = features['NumberOfUppercaseLetters'] /
url_length
features['HasMultipleSlashesInPath'] = int(path.count('/') > 1)
features['UseOfSuspiciousWords'] = int(any(word in url.lower() for word in
SUSPICIOUS_WORDS))

features['EntropyOfURL'] = -sum(
    (count / len(url)) * np.log2(count / len(url)) for count in Counter(url).values()
)

substr_counts = Counter()
for i in range(len(url)):
    for j in range(i + 1, len(url)):
        substr = url[i:j]
        substr_counts[substr] += 1
features['NumberOfRepeatedSubstrings'] = sum(1 for count in substr_counts.values()
if count > 1)

# Повертаємо масив (для скейлера)
return np.array([list(features.values())]), features

```

		Паращук М.А			ККІКБ.КР.Б – 125 – 15 – ПЗ	Арк.
		Маєвський О.В				
Змн.	Арк.	№ докум.	Підпис	Дата		73

Функція парсингу HTML-контенту

```

def extract_html_features(html, base_domain=None):
    soup = BeautifulSoup(html, 'html.parser')
    features = {}

    # Forms
    forms = soup.find_all('form')
    features['num_forms'] = len(forms)
    features['num_password_forms'] = sum(1 for f in forms if f.find('input', {'type':
'password'}) is not None)
    features['num_password_inputs'] = len(soup.find_all('input', {'type': 'password'}))

    # Iframes
    features['num_iframes'] = len(soup.find_all('iframe'))

    # Scripts
    scripts = soup.find_all('script')
    features['num_scripts'] = len(scripts)
    features['num_external_scripts'] = sum(1 for s in scripts if s.get('src') is not None)
    features['external_script_ratio'] = (features['num_external_scripts'] /
features['num_scripts']) if features['num_scripts'] > 0 else 0

    # Stylesheets
    stylesheets = soup.find_all('link', rel='stylesheet')
    features['num_external_stylesheets'] = sum(1 for s in stylesheets if s.get('href') and
s['href'].startswith('http'))

    # Images
    images = soup.find_all('img')
    features['num_external_images'] = sum(1 for img in images if img.get('src') and
img['src'].startswith('http'))

    # Meta refresh
    features['has_meta_refresh'] = int(soup.find('meta', attrs={'http-equiv': 'refresh'}) is
not None)

    # Text/HTML ratio
    text_length = len(soup.get_text())
    html_length = len(html)
    features['text_to_html_ratio'] = text_length / html_length if html_length > 0 else 0

    # Suspicious words

```

		Паращук М.А			ККІКБ.КР.Б – 125 – 15 – ПЗ	Арк.
		Маєвський О.В				
Змн.	Арк.	№ докум.	Підпис	Дата		74

```

suspicious_words = ['login', 'verify', 'update', 'bank', 'account', 'password', 'confirm',
'secure']
page_text = soup.get_text().lower()
features['use_suspicious_words'] = int(any(word in page_text for word in
suspicious_words))

# Links with IP
links = soup.find_all('a', href=True)
features['num_links_with_ip'] = sum(1 for a in links if
re.match(r"http[s]?://\d{1,3}(\.\d{1,3}){3}", a['href']))

# External link ratio
if base_domain:
    ext_links = sum(1 for a in links if base_domain not in urlparse(a['href']).netloc)
    features['external_link_ratio'] = ext_links / len(links) if links else 0
else:
    features['external_link_ratio'] = 0

# JS events (onmouseover, onclick, etc.)
features['num_js_events'] = len(soup.find_all(lambda tag: any(attr for attr in tag.attrs
if attr.startswith('on'))))

# Hidden elements (display:none or visibility:hidden)
features['num_hidden_elements'] = sum(1 for tag in soup.find_all(style=True)
if 'display:none' in tag['style'].lower() or 'visibility:hidden' in
tag['style'].lower())

# Legal sections
legal_keywords = ['terms', 'privacy', 'about', 'contact']
features['has_legal_info'] = int(any(word in page_text for word in legal_keywords))

# Login buttons
features['num_login_buttons'] = sum(1 for btn in soup.find_all(['button', 'input'])
if 'login' in (btn.get('value') or "").lower()
or 'login' in (btn.get_text() or "").lower())

return features

```

		Паращук М.А			ККІКБ.КР.Б – 125 – 15 – ПЗ	Арк.
		Маєвський О.В				75
Змн.	Арк.	№ докум.	Підпис	Дата		

Завантаження функцій в Flask

```

from flask import Flask, request, render_template
from pfishing import *
app = Flask(__name__)
# Попереднє завантаження даних
update_openphish_feed()
update_cisco_umbrella()
load_umbrella_domains()
download_majestic_top()
load_majestic_domains()

@app.route('/', methods=['GET', 'POST'])
def index():
    result = {}
    steps = []
    feedback_sent = False # Прапорець для повідомлення про успішну відправку
    відгуку

    if request.method == 'POST':
        # Якщо це форма зворотного зв'язку
        if 'feedback_url' in request.form:
            url_fb = request.form.get('feedback_url', "").strip()
            message = request.form.get('feedback_message', "").strip()
            if url_fb and message:
                file_exists = os.path.isfile(feedback_file)
                with open(feedback_file, "a", encoding="utf-8", newline="") as f:
                    if not file_exists:
                        f.write("|Дата|Url|Причина|\n")
                    f.write(f"|{datetime.now().strftime("%Y-%m-%d
%H:%M:%S")}|{url_fb}|{message}|\n")
                feedback_sent = True

        # Якщо це форма перевірки URL
        elif 'url' in request.form:
            url = request.form['url'].strip()
            result['checked_url'] = url

            if url:
                try:
                    # 1 Google Safe Browsing
                    is_blacklisted, details = check_url_in_blacklists(url, API_KEY)
                    steps.append(('Google Safe Browsing', details))
                    if is_blacklisted:

```

		Паращук М.А			ККІКБ.КР.Б – 125 – 15 – ПЗ	Арк.
		Маєвський О.В				
Змн.	Арк.	№ докум.	Підпис	Дата		76

```

        result['final'] = '⊖ Виявлено як небезпечний у Google Safe Browsing.
Перевірку завершено.'
        return render_template('index.html', result=result, steps=steps, feed-
back_sent=feedback_sent)

# 2 OpenPhish
in_openphish = is_in_openphish(url)
steps.append(('OpenPhish', '⊖ Виявлено' if in_openphish else '☑ Не
виявлено в чорних списках фішингу'))
if in_openphish:
    result['final'] = '⊖ URL знайдено в базі OpenPhish. Ймовірно
фішинговий. Перевірку завершено.'
    return render_template('index.html', result=result, steps=steps, feed-
back_sent=feedback_sent)

# 3 VirusTotal
vt_blacklisted, vt_details = check_url_in_virustotal(url, VIRUSTO-
TAL_API_KEY)
steps.append(('VirusTotal', vt_details))
if vt_blacklisted:
    result['final'] = '⊖ Виявлено як шкідливий у VirusTotal. Перевірку
завершено.'
    return render_template('index.html', result=result, steps=steps, feed-
back_sent=feedback_sent)

# 4 Cisco Umbrella
in_umbrella = is_in_umbrella_whitelist(url)
steps.append(('Cisco Umbrella(список безпечних сайтів)', '☑ У списку'
if in_umbrella else '⊖ Відсутній'))
if in_umbrella:
    result['final'] = '☑ Домен знайдено у Cisco Umbrella. Вважається
безпечним.'
    return render_template('index.html', result=result, steps=steps, feed-
back_sent=feedback_sent)

# 5 Majestic Million
in_majestic = is_in_majestic_whitelist(url)
steps.append(('Majestic Million(список безпечних сайтів)', '☑ У списку'
if in_majestic else '⊖ Відсутній'))
if in_majestic:
    result['final'] = '☑ Домен знайдено у Majestic Million. Вважається
безпечним.'

```

		Паращук М.А			ККІКБ.КР.Б – 125 – 15 – ПЗ	Арк.
		Маєвський О.В				77
Змн.	Арк.	№ докум.	Підпис	Дата		

```
return render_template('index.html', result=result, steps=steps, feedback_sent=feedback_sent)
```

```
# 6 Власний білий список
```

```
main_domain = get_base_domain(url)
```

```
steps.append(('Основний домен', main_domain or '✗ Не визначено'))
```

```
in_whitelist = main_domain in WHITELIST_DOMAINS
```

```
steps.append(('Білий список', '✓ Так' if in_whitelist else '⊖ Ні'))
```

```
if in_whitelist:
```

```
    result['final'] = f'✓ Домен {main_domain} у власному білому списку.
```

```
    Вважається безпечним.'
```

```
    return render_template('index.html', result=result, steps=steps, feedback_sent=feedback_sent)
```

```
back_sent=feedback_sent)
```

```
# 7 WHOIS
```

```
whois_info = check_whois_info(url)
```

```
steps.append(('WHOIS', whois_info))
```

```
# 8 Прогноз моделі
```

```
features_array, _ = extract_features(url)
```

```
features_scaled = scaler.transform(features_array)
```

```
prediction = model.predict(features_scaled)[0][0]
```

```
result['model_prediction'] = prediction
```

```
result['model_label'] = '⚠ Фішинговий сайт!' if prediction >= 0.5 else
```

```
'✓ Безпечний сайт.'
```

```
steps.append(('Прогноз моделі', f'{prediction:.4f} → {result["model_label"]}))
```

```
html = get_html_content(url)
```

```
if html:
```

```
    html_features = extract_html_features(html)
```

```
    is_html_suspicious = analyze_html_features(html_features)
```

```
    result['html_analysis'] = html_features
```

```
    result['html_suspicious'] = is_html_suspicious
```

```
else:
```

```
    result['html_error'] = "✗ Не вдалося отримати HTML-код сторінки."
```

```
if prediction<0.5:
```

```
    if 'html_suspicious' in result and result['html_suspicious']:
```

```
        result['final'] = "⚠ Модель вважає сайт безпечним, але HTML виглядає підозрілим. Використовуйте з обережністю."
```

```
    elif 'html_error' in result:
```

```
        result['final'] = "✓ Модель вважає сайт безпечним, але HTML не вдалося отримати."
```

		Паращук М.А			ККІКБ.КР.Б – 125 – 15 – ПЗ	Арк.
		Маєвський О.В				78
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        else:
            result['final'] = " ✓ Остаточний висновок: сайт безпечний (модель
і HTML позитивні)."
        else:
            if 'html_suspicious' in result and result['html_suspicious']:
                result['final'] = " ✗ Остаточний висновок: сайт є фішинговим
(модель + HTML-парсинг)."
            elif 'html_error' in result:
                result['final'] = " ⚠ Модель вважає сайт фішинговим, але HTML
не вдалося отримати. Використовуйте з обережністю."
            else:
                result['final'] = " ⚠ Модель вважає сайт фішинговим, але HTML
виглядає безпечно. Використовуйте з обережністю."

    except Exception as e:
        result['error'] = str(e)

    return render_template('index.html', result=result, steps=steps, feedback_sent=feed-
back_sent)

if __name__ == '__main__':
    app.run(debug=True)

```

		Паращук М.А			ККІКБ.КР.Б – 125 – 15 – ПЗ	Арк.
		Маєвський О.В				
Змн.	Арк.	№ докум.	Підпис	Дата		79