Міністерство освіти і науки України
Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра захисту інформації

# Бакалаврська кваліфікаційна робота на тему:

«Засіб для перевірки QR-кодів на наявність фішингових загроз»

Виконав: студент 4 курсу групи 1БС-21 б
спеціальності 125 Кібербезпека
_____ Владислав ШПАКОВАТИЙ

Керівник: к. т. н., доцент каф. ЗІ
_____ Віталій ЛУКІЧОВ
«12» червня 2025 р.

Рецензент: к. т. н. доцент каф. ПЗ
_____ Денис КАТЄЛЬНІКОВ
«12» червня 2025 р.

**Допущено до захисту**
Завідувач кафедри ЗІ
д. т. н., проф.
_____ Володимир ЛУЖЕЦЬКИЙ
«12» червня 2025 р.

Вінниця ВНТУ – 2025 року

Міністерство освіти і науки України
Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра захисту інформації
Рівень вищої освіти І (бакалаврський)
Галузь знань – 12 Інформаційні технології
Спеціальність – 125 Кібербезпека
Освітньо-професійна програма – Безпека інформаційних і комунікаційних систем

**ЗАТВЕРДЖУЮ**
**Завідувач кафедри ЗІ,**
**д. т. н,, проф.**
**Володимир ЛУЖЕЦЬКИЙ**
**«20» березня 2025 року**

# З А В Д А Н Н Я
## НА БАКАЛАВРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Владиславу ШПАКОВАТОМУ

1. Тема роботи: «Засіб для перевірки QR-кодів на наявність фішингових загроз» керівник роботи: Віталій ЛУКІЧОВ, к. т. н., доцент кафедри ЗІ, затверджені наказом ректора ВНТУ від 20 березня 2025 року №97.
2. Строк подання студентом роботи 12 червня 2025 р.
3. Вихідні дані до роботи:
   - Галузь застосування: Перевірка QR-кодів на наявність фішингових загроз.
   - Метод тестування: Динамічне тестування безпеки (DAST)
   - Використовувані інструменти: Google Safe Browsing, VirusTotal, URLScan.io).
   - Тип інструментів: Відкриті (open-source).
4. Зміст текстової частини: Вступ. 1 Аналіз методів та інструменів для перевірки QR-кодів на наявність фішингових загроз. 2 Метод для перевірки QR-кодів на наявність фішингових загроз. 3 Засіб для тестування QR-кодів на наявність фішингових загроз. Висновки. Список використаних джерел.
5. Перелік ілюстративного матеріалу: діаграма відсоткових значень фішингових атак за типами, блок схема роботи програмного засобу, головне вікно програми, результат перевірки безпечного QR-коду, результат перевірки небезпечного QR-коду, порівняння ефективності методів.

6. Консультанти розділів роботи

| Розділ | Прізвище, ініціали та посада консультанта | Підпис, дата | |
|---|---|---|---|
| | | Завдання видав | Завдання прийняв |
| 1 | В. ЛУКІЧОВ, к.т.н., доц. каф.ЗІ | 24.03.25 | 10.06 |
| 2 | В. ЛУКІЧОВ, к.т.н., доц. каф.ЗІ | 14.05.25 | 10.06 |
| 3 | В. ЛУКІЧОВ, к.т.н., доц. каф.ЗІ | 31.05.25 | 10.06 |

7. Дата видачі завдання _20__ _03__ 2025 року.

## КАЛЕНДАРНИЙ ПЛАН

| № з/п | Назва етапів бакалаврської кваліфікаційної роботи | Строк виконання етапів роботи | Примітк |
|---|---|---|---|
| 1 | Аналіз завдання. Вступ | 24.03.25 - 30.03.25 | |
| 2 | Аналіз інформаційних джерел за напрямком бакалаврської кваліфікаційної роботи | 31.03.25 - 04.05.25 | |
| 3 | Розробка рішень, моделей, алгоритмів | 14.05.25 - 04.05.25 | |
| 4 | Практична реалізація, моделювання, експериментування, результати | 05.05.25 - 14.05.25 | |
| 5 | Оформлення пояснювальної записки | 15.05.25 - 18.05.25 | |
| 6 | Попередній захист БКР | 19.05.25 - 26.05.25 | |
| 7 | Виправлення зауважень, підготовка ілюстративного матеріалу | 27.05.25 - 30.05.25 | |
| 8 | Представлення БКР до захисту, рецензування | 31.05.25 - 10.06.25 | |
| 9 | Захист БКР | 17.06.25 - 20.06.25 | |

Студент _____ Владислав ШПАКОВАТ
( підпис )

Керівник роботи _____ Віталій ЛУКІЧОВ
( підпис

## АНОТАЦІЯ

Бакалаврська кваліфікаційна робота складається з 72 сторінок формату А4, на яких є 19 рисунків, 6 таблиць, список використаних джерел містить 22 найменування.

Бакалаврська робота присвячена розробці засобу для перевірки QR-кодів на наявність фішингових загроз. Проведено аналіз сучасних методів виявлення фішингу. Розроблено програмний засіб, який дозволяє користувачам здійснювати поетапну перевірку QR-кодів за допомогою зовнішніх сервісів, отримувати розгорнутий звіт про потенційні загрози та зберігати історію перевірок. Реалізований інструмент підвищує рівень захисту від фішингових атак, орієнтований на зручність використання та інтеграцію в освітнє або корпоративне середовище.

Ключові слова: QR-код, фішинг, кібербезпека, аналіз URL, захист користувачів, розпізнавання QR-кодів, алгоритми виявлення загроз, кіберзлочинність, фішингові сайти.



## ABSTRACT

The bachelor's qualification work consists of 72 pages of A4 format, which include 19 figures, 6 tables, and the list of sources used contains 22 names.

The bachelor's thesis is devoted to the development of a tool for checking QR codes for phishing threats. An analysis of modern methods of detecting phishing has been conducted. A software tool has been developed that allows users to perform a step-by-step check of QR codes using external services, receive a detailed report on potential threats and store the history of checks. The implemented tool increases the level of protection against phishing attacks, focused on ease of use and integration into an educational or corporate environment.

Keywords: QR code, phishing attacks, cybersecurity, URL analysis, user protection, QR code recognition, threat detection algorithms, cybercrime, phishing sites.

# ЗМІСТ

# ВСТУП

Інформаційна безпека стала критично важливою складовою сучасного цифрового суспільства. У зв’язку з активним впровадженням мобільних технологій та цифрових каналів комунікації, користувачі дедалі частіше стикаються з новими формами кіберзагроз, які маскуються під звичні та зручні для використання інструменти. Одним із таких інструментів є QR-коди, що набули широкого поширення у фінансовій сфері, логістиці, освіті, медицині та інших галузях. Проте поряд із їх зростаючою популярністю спостерігається також активне використання цих кодів у фішингових кампаніях

Фішинг з використанням QR-кодів, відомий як QR-фішинг або quishing, є новим видом соціальної інженерії, який дає змогу зловмисникам приховувати шкідливі посилання у графічному коді. Оскільки користувач зазвичай не бачить вміст посилання до моменту відкриття, йому важко ідентифікувати загрозу. Це робить QR-фішинг особливо небезпечним у мобільному середовищі, де автоматичне відкриття посилань після сканування значно ускладнює контроль за безпекою.

У контексті сучасного стану кібербезпеки стає очевидною потреба у розробці засобів, які дозволяють користувачам заздалегідь перевіряти QR-коди на предмет фішингової активності без необхідності відкривати закодовані посилання. Огляд наявних рішень показав, що більшість доступних інструментів орієнтовані або на корпоративний сектор, або не забезпечують достатньої глибини перевірки. Крім того, багато існуючих програм і додатків не підтримують детальний аналіз структури URL, не дозволяють звертатися до зовнішніх сервісів оцінки загроз або вимагають складної інтеграції. Саме тому розробка зручного, автономного, функціонального програмного засобу для перевірки QR-кодів є актуальним завданням, що відповідає світовим тенденціям у сфері захисту інформації

Метою бакалаврської кваліфікаційної роботи є підвищення захисту користувачів від фішингових атак через QR-коди шляхом розробки програмного

засобу, який дозволяє автоматично аналізувати закодовані посилання на наявність фішингових загроз за допомогою зовнішніх сервісів перевірки.

Для досягнення поставленої мети в рамках роботи було вирішено такі задачі:

1) аналіз загроз інформаційної безпеки, пов’язаних із QR-фішингом, та критичний огляд наявних методів його виявлення;
2) дослідження можливостей зовнішніх сервісів перевірки посилань, зокрема Google Safe Browsing, VirusTotal і URLScan, а також їх порівняння за критеріями точності, швидкодії та зручності інтеграції;
3) обґрунтування вибору мови програмування, середовища розробки та програмної архітектури, з урахуванням вимог до швидкої обробки графічних зображень та взаємодії з API;
4) створення методу, який дозволяє поетапно обробляти QR-коди, витягувати URL, здійснювати багаторівневу перевірку і повертати користувачу узагальнений висновок;
5) створення користувацького інтерфейсу, який забезпечує простоту взаємодії з програмою, підтримує вставлення даних, збереження історії перевірок і виведення розгорнутих результатів;
6) проведення тестування функціональності програми на різноманітних прикладах, включаючи як безпечні, так і потенційно небезпечні QR-коди, для оцінки точності та надійності реалізованого засобу.

Виконання цих задач дозволило реалізувати повноцінний програмний інструмент, орієнтований на підвищення рівня цифрової безпеки користувачів при роботі з QR-кодами. Запропонований підхід відповідає актуальним потребам суспільства в доступних і ефективних засобах захисту від нових форм кіберзагроз.

Результати дослідження були апробовані через публікацію тез: Шпаковатий В. О. , Лукічов В. В. Засіб для перевірки QR-кодів на наявність фішингових загроз. Матеріали LIV науково-технічної конференції підрозділів ВНТУ, Вінниця, 24-27 березня 2025 р.

# 1 АНАЛІЗ МЕТОДІВ ТА ІНСТРУМЕНІВ ДЛЯ ПЕРЕВІРКИ QR-КОДІВ НА НАЯВНІСТЬ ФІШИНГОВИХ ЗАГРОЗ

## 1.1 Поняття фішинг та його вплив на сучасний світ

Фішинг – це форма соціальної інженерії, яка полягає в розсилці повідомленням, що нібито надходять з легітимних джерел, та спробі переконати одержувачів розкрити конфіденційну інформацію, таку як паролі та номери кредитних карток, це форма кібератаки, яка існує з середини 1990-х років, але методи, що використовуються, стають дедалі складнішими.

Світова статистика свідчить про стрімке зростання кількості фішингових інцидентів. За даними звітів міжнародних компаній у галузі кібербезпеки, понад 90% успішних кібератак починаються саме з фішингу. Особливо актуальною ця проблема стала в умовах пандемії COVID-19, коли значна частина людей перейшла на віддалену роботу, а підприємства не встигли належним чином адаптувати свої системи безпеки до нових загроз.

Окрему загрозу становлять фішингові кампанії, які орієнтовані на конкретні цілі – урядові установи, енергетичні компанії, медичні організації, великі корпорації. Такі атаки ретельно підготовлені, використовують соціальну інженерію та «цільову персоналізацію», що ускладнює їх виявлення традиційними засобами безпеки.

Широке розповсюдження штучного інтелекту і генеративних моделей (таких як ChatGPT) значно змінило ландшафт фішингових атак. Злочинці вже використовують AI для створення граматично правильних, переконливих повідомлень, які важко відрізнити від справжніх. Також автоматизовані системи дозволяють запускати масові фішингові кампанії без участі людини, масштабуючи атаки до глобального рівня.

Фішинг перестав бути лише «поштовою» проблемою – сучасні сценарії передбачають інтеграцію фішингу в багатоступеневі атаки, що включають злом

систем, крадіжку облікових даних, подальше проникнення в мережу компанії, розгортання шкідливого ПЗ або запуск програм-вимагачів.

За оцінками галузі, фінансові збитки від фішингових атак лише у 2023 році перевищили 3,5 мільярда доларів США, а збитки від супутніх атак на основі викрадених облікових даних можуть бути в рази вищими.

Фішингові атаки не показують ознак зменшення. Електронна пошта залишається основним вектором цих атак, але вони також можуть здійснюватися через SMS-повідомлення, соціальні мережі та месенджери. Незалежно від засобу передачі інформації, фішинг – це загроза, яку не можна ігнорувати.

У третьому кварталі 2023 року кількість фішингових атак зросла на 173% порівняно з попереднім кварталом (493,2 мільйона проти 180,4 мільйона) [1].

З моменту запуску ChatGPT наприкінці 2022 року обсяг фішингових електронних листів зріс на колосальні 1265%. Фішинг облікових даних продовжує своє стратосферне зростання зі збільшенням на 967%, що зумовлено головним чином вимогами груп програм-вимагачів, які шукають доступу до компаній в обмін на гроші.

Смішинг (SMS-фішинг) зріс до 39% мобільних загроз. Щомісяця надсилається 10 мільйонів повідомлень TOAD (доставка атак, орієнтованих на телефон). Ці фішингові повідомлення зі зворотним викликом допомагають жертвам розкрити конфіденційну інформацію та облікові дані.

Середній час, протягом якого користувачі потрапляють на фішингові електронні листи, становить менше 1 хвилини: середній час переходу за шкідливим посиланням після відкриття електронного листа становить 21 секунду, і лише 28 секунд потрібно цілі, щоб надати запитувану інформацію.

Кіберзлочинці надсилають приблизно 3,4 млрд фішингових електронних листів, що робить їх найпоширенішою формою кіберзлочинності [2] .

Графік зростання кількості фішингових атак у світі зображено на рис. 1.1.

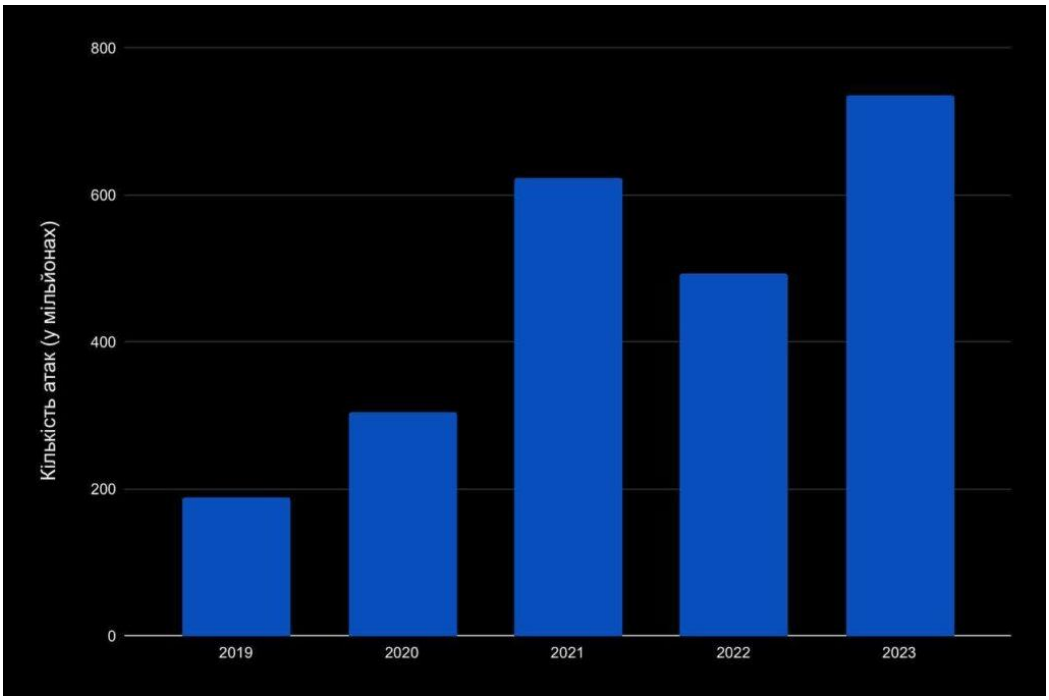



Рисунок 1.1 – Кількості фішингових атак по рокам

## 1.2 Використання штучного інтелекту в фішингу

У центрі уваги – фішингові загрози, що використовують штучний інтелект GenAI, безсумнівно, довів свою трансформаційну силу у підвищенні продуктивності в усіх компаніях. Однак, з іншого боку цієї трансформації є небезпечна правда: ШІ також перетворює новачків та пересічних порушників на кваліфікованих соціальних інженерів та досвідчених фішингових атак.

Автоматизуючи та персоналізуючи різні компоненти процесу атаки, ШІ прискорює та вдосконалює фішингові атаки, роблячи їх більш витонченими та важкими для виявлення.

GenAI швидко аналізує публічні дані , такі як інформація про організації та керівників, заощаджуючи час на розвідку для зловмисників та дозволяючи здійснювати точніші цільові атаки.

Чат-боти LLM створюють точні та правдоподібні фішингові повідомлення та електронні листи, усуваючи орфографічні та граматичні помилки.

GenAI може швидко створювати переконливі фішингові сторінки. У звіті ThreatLabz показано, як ChatGPT створив фішингову сторінку входу менш ніж за

10 запитів, і наведено ключові індикатори, на які слід звернути увагу під час ідентифікації фішингової сторінки.

Штучний інтелект розмив межу між справжнім та шахрайським контентом, що ще більше ускладнює розрізнення фішингових схем від легітимних веб-сторінок та цифрового спілкування.

У міру того, як дослідники ThreatLabz відстежували тенденції фішингу протягом 2023 року, також з'явилося кілька помітних передових тактик ШІ. Серед них було зростання популярності вішингу та діпфейкового фішингу, дедалі популярніших тактик соціальної інженерії, що використовують інструменти імперсонації на базі ШІ.

Розширені кампанії з використанням вішингу набирають популярності в усьому світі, що в деяких випадках призводить до значних фінансових втрат. Під час помітної спроби, яку ThreatLabz запобігла влітку 2023 року, фішингові зловмисники використали технологію штучного інтелекту для здійснення атаки вішингу, видаючи себе за генерального директора Zscaler Джея Чаудрі. У звіті детально описана послідовність подій, що слугує важливим нагадуванням для підприємств і співробітників про необхідність пильності щодо шахраїв, що використовують вішинг. ThreatLabz очікує подальшого зростання кількості цільових голосових фішингових кампаній, очолюваних такими групами, як Scattered Spider , наступного року. Оскільки ці зусилля спрямовані на отримання облікових даних співробітників, організаціям вкрай важливо посилити свій захист від фішингу, щоб запобігти несанкціонованому доступу та експлуатації.

Фішингові атаки з використанням діпфейків стануть однією з найскладніших кіберзагроз, керованих штучним інтелектом. Зловмисники тепер мають можливість створювати відеоконтент, який точно та безпомилково відтворює обличчя, голоси та манери. Ця маніпуляція вже проявилася в тривожних аспектах, наприклад, у виборчому процесі, де діпфейкові відео фабрикують неправдиві наративи або заяви політичних діячів. Ці відео можуть впливати на громадську думку, поширювати дезінформацію та підривати довіру до чесності виборчого процесу. Оскільки суспільство стає все більш залежним від цифрового зв'язку та споживання медіа, потенційні політичні та доленосні наслідки шахрайства з діпфейками, ймовірно,

виходитимуть далеко за рамки поточних застосувань, використання технології діпфейків становить значну загрозу для організацій, окремих осіб та суспільства в цілому.

Крім того, ThreatLabz спостерігав зростання шахрайства з QR-кодами , шахрайства з вербуванням , атак типу «браузер у браузері» (BitB) та атак типу «супротивник посередині» (AiTM), на рис. 1.2 наведено приблизні відсоткові значення розподілу фішингових атак за типами, засновані на останніх дослідженнях у сфері кібербезпеки [3].

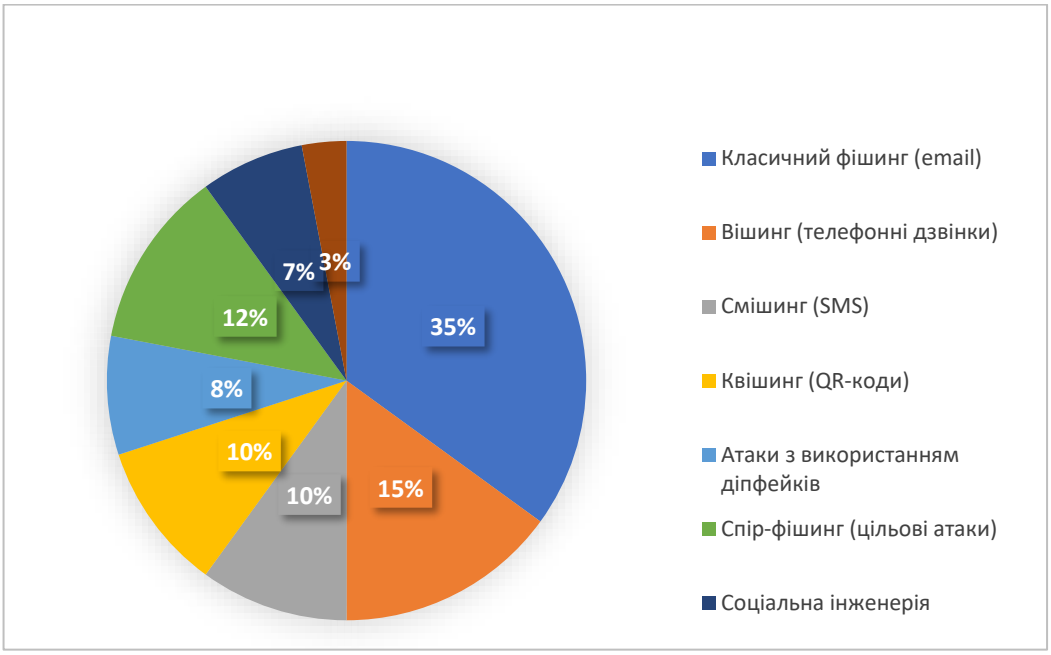



Рисунок 1.2 – Діаграма відсоткових значень фішингових атак за типами

## 1.3 Фішинг з використанням QR-кодів

У сучасному цифровому середовищі, де користувачі щоденно взаємодіють із візуальними та інтерактивними елементами, QR-коди стали звичним способом швидкого доступу до онлайн-ресурсів. Їх використовують у банківській справі, закладах харчування, рекламних кампаніях, транспорті, навчальних матеріалах і навіть у системах аутентифікації. Завдяки зручності та універсальності ці двовимірні коди стрімко здобули популярність у всьому світі. Проте з їх широким розповсюдженням зросла й зацікавленість з боку кіберзлочинців. QR-фішинг, або

QRishing, – це форма фішингової атаки, яка використовує QR-коди для обманного перенаправлення користувачів на шкідливі або підроблені вебсторінки з метою викрадення особистих даних, облікових записів чи фінансової інформації.

На відміну від класичних фішингових методів, які здебільшого базуються на електронній пошті або повідомленнях, QR-фішинг експлуатує довіру до технологій і відсутність попереднього перегляду посилання перед переходом. Користувачі звикли сканувати QR-коди без особливих застережень, що й дозволяє зловмисникам приховати під кодом фішингове посилання. Як результат, жертва не бачить жодних підозрілих ознак до моменту, коли вже перейшла на шкідливий ресурс. Саме ця "невидимість" є однією з основних причин стрімкого поширення цього типу атак.

Однією з найпоширеніших тактик є підміна QR-кодів у фізичному просторі. Зловмисники друкують власні коди, які ведуть на фішингові сторінки, і наклеюють їх поверх оригінальних на постерах, меню в ресторанах, автоматах з оплатою чи навіть у громадському транспорті. В результаті користувач, нічого не підозрюючи, сканує код і переходить, наприклад, на фальшиву сторінку входу до онлайн-банкінгу чи облікового запису Microsoft. Деякі атаки поширюються електронною поштою або в месенджерах – наприклад, користувач отримує повідомлення із QR-кодом, який нібито потрібно просканувати для підтвердження акаунту, перевірки безпеки чи участі в опитуванні.

Особливу небезпеку становить те, що більшість QR-кодів скануються з мобільних пристроїв. На телефонах зазвичай відсутні розширені антивірусні системи або засоби перевірки посилань, якими користуються настільні браузери. Крім того, деякі мобільні додатки автоматично відкривають посилання одразу після сканування, що ще більше знижує шанси на виявлення шахрайства. Це створює сприятливе середовище для атак, орієнтованих на викрадення мобільних облікових даних, банківської інформації або навіть ініціацію фінансових транзакцій через мобільні гаманці [4].

Впродовж 2023–2024 років експерти з кібербезпеки зафіксували значне зростання кількості фішингових кампаній з використанням QR-кодів. У багатьох із них фігурували підроблені логотипи Microsoft, Google, Amazon, банківських

установ та платформ для оплати. Зловмисники все частіше використовують QR-коди для обходу традиційних фільтрів електронної пошти, які орієнтуються на текстові або HTML-посилання. QR-код, що містить шкідливу URL-адресу, є складнішим для автоматичного виявлення, особливо в графічному форматі, вбудованому в тіло листа.

Найбільш вразливими до QR-фішингу є працівники великих організацій, які регулярно сканують коди для доступу до внутрішніх систем, підтвердження особи або проходження навчальних курсів. Атаки можуть призводити до втрати конфіденційних документів, зламу внутрішніх мереж або прямого фінансового збитку. Небезпека також зростає у зв'язку з популяризацією QR-кодів у системах двофакторної автентифікації – зловмисники імітують сторінки входу, що потребують сканування, збираючи при цьому одноразові коди доступу та облікові дані.

На цьому тлі надзвичайно важливою стає потреба в освіті користувачів щодо потенційних ризиків, пов'язаних із QR-кодами. Скептичне ставлення до будь-якого QR-коду, отриманого електронною поштою або розміщеного у підозрілому місці, повинно стати правилом. Також варто використовувати додатки, які дозволяють попередньо переглянути URL перед переходом. Компаніям рекомендується впроваджувати політики безпеки, які обмежують або контролюють використання QR-кодів у критичних процесах.

З урахуванням високого рівня довіри до QR-технологій, мобільної необізнаності користувачів і простоти створення шкідливих кодів, фішинг з використанням QR-кодів перетворюється на одну з найперспективніших та найнебезпечніших форм соціальної інженерії. Його ефективність базується не лише на технічних хитрощах, а й на психологічній невибагливості – ми звикли сканувати швидко і не задумуючись. Саме тому QRishing заслуговує на особливу увагу з боку фахівців з інформаційної безпеки, освітніх закладів та державних структур, які мають формувати свідоме цифрове мислення серед населення [5]. Статистика зростання кількості випадків використання qr-кодів у відсотках від загальної кількості атак зображено на рис. 1.3.

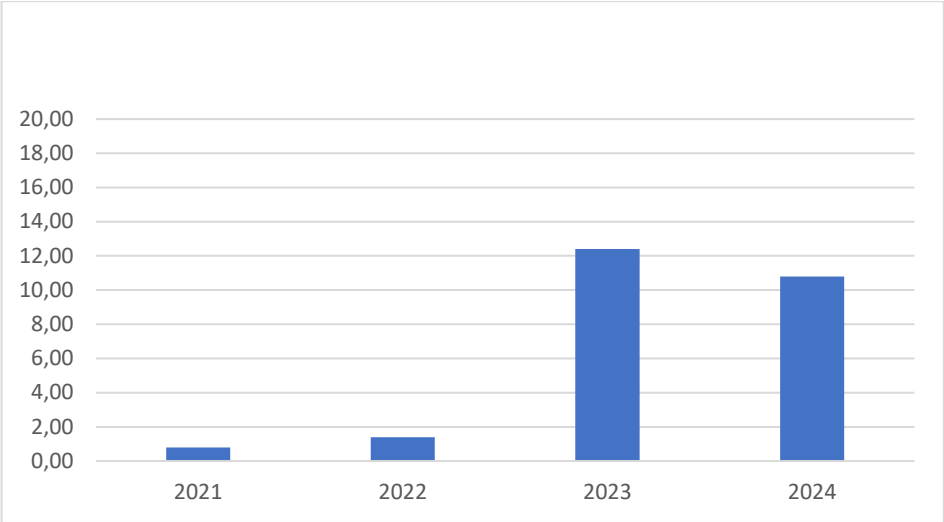



Рисунок 1.3 – Діаграма відсоткової частини використання QR-кодів для фішингу

## 1.4 Методи аналізу QR-кодів

Статичний аналіз. Статичний аналіз QR-кодів являє собою перший і найважливіший етап перевірки, який дозволяє виявити потенційні загрози без необхідності безпосередньої взаємодії з закодованим ресурсом. Сучасні системи статичного аналізу розвинулися у високотехнологічні комплекси, що поєднують традиційні методи з інноваційними підходами на основі штучного інтелекту.

На початковому етапі аналізу система виконує декодування QR-коду, перетворюючи графічне зображення у цифрові дані. Цей процес вже сам по собі може виявити певні аномалії – навмисні спотворення структури коду, додаткові графічні елементи або спроби обману алгоритмів розпізнавання. Сучасні алгоритми використовують методи комп'ютерного зору для виявлення навіть незначних маніпуляцій з візуальною частиною QR-коду, таких як мінімальні зміни в контрастності окремих модулів або накладення додаткових шаблонів.

Після успішного декодування починається основна фаза статичного аналізу – перевірка текстового вмісту. Якщо QR-код містить URL (що є найпоширенішим вектором атак), система застосовує багаторівневий аналіз посилання. Перший рівень включає базові перевірки синтаксису – коректність формату URL, наявність заборонених символів або явних спроб обману (наприклад, використання кирилічних літер, що візуально не відрізняються від латинських).

Наступний етап – перевірка доменного імені за допомогою інтегрованих чорних списків. Сучасні системи використовують не єдину статичну базу, а цілу мережу взаємопов'язаних джерел, що постійно оновлюються. До них входять як комерційні бази даних (Google Safe Browsing, VirusTotal), так і галузеві репозиторії (наприклад, фінансового сектору), а також децентралізовані системи обміну інформацією про загрози. Особливістю сучасних систем є їх здатність до предикативного аналізу – виявлення потенційно небезпечних ресурсів ще до їх офіційного внесення до чорних списків, на основі аномалій у поведінці або характеристиках домену.

Для аналізу скорочених URL сучасні системи використовують спеціалізовані алгоритми, які не просто розшифровують посилання, а й будують цілі ланцюжки перенаправлень. Вони аналізують кожен проміжний домен, його репутацію, географічну приналежність та інші параметри. Це дозволяє виявляти складні схеми, коли зловмисники використовують кілька рівнів перенаправлень для приховування кінцевого шкідливого ресурсу.

Окремим напрямком статичного аналізу є перевірка інших типів даних, які можуть бути закодовані в QR-коді. Для кожного типу вмісту (контактні дані vCard, географічні координати, Wi-Fi параметри тощо) розроблені спеціалізовані алгоритми перевірки, що виявляють потенційно небезпечні значення або формати.

Важливою складовою сучасного статичного аналізу є репутаційна система, яка оцінює не тільки безпосередній вміст QR-коду, а й контекст його використання. Наприклад, система може враховувати географічне розташування коду, час його активності, частоту сканувань та інші метадані, що дозволяє виявляти аномальні закономірності.

Сучасні системи статичного аналізу також інтегрують технології прогнозування ризиків на основі історичних даних. Вони можуть виявляти нові схеми атак, аналізуючи схожість з вже відомими загрозами, навіть якщо конкретний екземпляр QR-коду ще не зустрічався в базі даних [6].

Динамічний аналіз QR-кодів. Сучасний динамічний аналіз QR-кодів перетворився на складний багатоетапний процес, що значно відрізняється від примітивних методів перевірки, які використовувалися ще кілька років тому. Цей

підхід передбачає не просто пасивне "читання" закодованої інформації, а активне взаємодію з ресурсом, на який веде QR-код, з метою виявлення навіть найбільш витончених схем обману. Сучасні системи динамічного аналізу являють собою справжні лабораторії кібербезпеки, де кожен підозрілий QR-код проходить серйозну перевірку за декількома напрямками одночасно.

Основним інструментом динамічного аналізу стали розвинені системи пісочниць (sandbox), які за останні роки набули неймовірної складності. Сучасна пісочниця – це не просто ізольоване середовище, а цілий віртуальний світ, що точно імітує різні апаратні платформи, операційні системи та навіть специфічні конфігурації пристроїв. Вони здатні емулювати роботу процесорів різних архітектур (x86, ARM, MIPS), графічних прискорювачів, мережевих інтерфейсів і навіть периферійних пристроїв. Це дозволяє виявляти атаки, які націлені на конкретні моделі пристроїв або використовують уразливості певних компонентів системи. Наприклад, деякі шкідливі сайти містять експлойти, що активуються тільки при певних характеристиках процесора або версії графічного драйвера – сучасні пісочниці здатні виявляти і такі складні загрози.

Особливу увагу приділяють аналізу поведінки вебсторінок у пісочниці. Сучасні системи реєструють і аналізують абсолютно всі дії, що відбуваються під час завантаження сторінки: від запитів до зовнішніх серверів до спроб доступу до локальних ресурсів. Вони здатні виявляти навіть найменші аномалії в поведінці сторінки, такі як спроби прихованого завантаження додаткових ресурсів, виконання підозрілих JavaScript-скриптів або маніпуляції з DOM-деревом. При цьому аналізується не тільки сам факт таких дій, а й їх послідовність, тимчасові характеристики та взаємозв'язки – це дозволяє виявляти складні багатоетапні атаки, коли шкідливий код активується лише за певної послідовності дій.

Окремим напрямком динамічного аналізу став розвиток технологій емуляції мобільних пристроїв. Враховуючи, що більшість QR-кодів сканується саме смартфонами, сучасні системи здатні точно імітувати роботу сотень різних моделей мобільних пристроїв з урахуванням їх апаратних особливостей, версій операційних систем і навіть специфіки роботи мобільних браузерів. Вони враховують такі фактори, як роздільна здатність екрану, особливості сенсорного

введення, наявність і версії різних додатків. Це критично важливо, оскільки багато фішингових атак спеціально розраховані саме на мобільні пристрої і можуть не проявляти себе при перевірці на десктопних системах.

Сучасні системи динамічного аналізу йдуть ще далі – вони не просто пасивно спостерігають за поведінкою сторінки, а активно з нею взаємодіють. Спеціальні алгоритми імітують дії реального користувача: кліки по кнопках, введення тестових даних у форми, навігацію по сторінці. При цьому реєструється будь-яка підозріла реакція: спроби відправки даних на невідомі сервери, зміни в інтерфейсі, поява нових елементів. Особливу увагу приділяють аналізу форм введення – сучасні системи здатні виявляти навіть професійно підроблені форми входу, які візуально не відрізняються від справжніх, але мають характерні відмінності у коді або поведінці.

Темпоральний аналіз – ще один важливий аспект сучасних систем динамічної перевірки. Він передбачає реєстрацію і аналіз часових характеристик роботи сторінки: затримок між діями користувача і реакцією системи, часу завантаження окремих елементів, періодичності мережевих запитів. Це дозволяє виявляти приховані процеси, такі як фонове завантаження шкідливого коду або спроби підключення до командних серверів. Наприклад, нехарактерні затримки при відправці форми можуть свідчити про спробу перехоплення введених даних.

Мережевий аналіз – невід'ємна частина сучасного динамічного дослідження. Сучасні системи реєструють і аналізують всі мережеві запити, що йдуть з тестованої сторінки: DNS-запити, HTTP/HTTPS-запити, WebSocket-з'єднання. Вони здатні виявляти підозрілі шаблони комунікації, такі як звернення до доменів з підозрілою репутацією, використання нестандартних портів або спроби встановлення зворотного з'єднання. Особливу увагу приділяють аналізу SSL/TLS-з'єднань – перевіряється валідність сертифікатів, підтримувані шифри, наявність вразливостей.

Сучасні системи динамічного аналізу також включають потужні інструменти для виявлення обфускації коду. Вони здатні аналізувати і деобфусцирувати JavaScript, виявляти динамічно завантажувані скрипти, аналізувати техніки

прихованого виконання коду. Це дозволяє виявляти складні схеми, коли шкідливий код завантажується частинами або активується лише за певних умов.

Прогрес у галузі динамічного аналізу QR-кодів не зупиняється. Вже зараз ведуться роботи з інтеграції технологій штучного інтелекту для більш точного прогнозування поведінки сторінок, розробляються методи квантового аналізу для виявлення особливо складних загроз, тестуються системи колективної обробки даних, де інформація про загрози збирається з мільйонів пристроїв по всьому світу. Все це робить сучасні системи динамічного аналізу надійним бар'єром на шляху QR-фішингу та інших кіберзагроз, пов'язаних з використанням QR-кодів [7].

Машинне навчання в аналізі QR-кодів. Машинне навчання (ML) відіграє ключову роль у сучасних системах виявлення фішингових загроз у QR-кодах. На відміну від традиційних методів, які базуються на заздалегідь визначених правилах, ML-алгоритми здатні навчатися на великих обсягах даних і виявляти складні шаблони, пов'язані з фішингом. Одним із найпоширеніших підходів є класифікація URL за допомогою природномовної обробки (NLP). Цей метод передбачає аналіз текстового вмісту URL та вебсторінки для виявлення ознак фішингу. Наприклад, алгоритми можуть оцінювати схожість доменного імені з легальними сайтами, наявність підозрілих ключових слів (таких як "login", "secure", "account") або незвичайну структуру URL.

Для класифікації URL використовуються як традиційні алгоритми, такі як Support Vector Machines (SVM) або Random Forest, так і сучасні моделі глибинного навчання, такі як BERT або Transformer. Останні особливо ефективні, оскільки вони здатні аналізувати контекст і семантику тексту, що дозволяє виявляти більш складні схеми обману. Наприклад, модель BERT може визначити, що URL "hxxps://secure-login-amazon.com" є підозрілим, навіть якщо він містить слова, схожі на легальний сайт Amazon. Такі моделі навчаються на великих наборах даних, що включають як безпечні, так і фішингові URL, що дозволяє їм досягати високої точності у виявленні загроз.

Окремим напрямком у використанні машинного навчання є застосування комп'ютерного зору для аналізу QR-кодів. Цей метод особливо корисний для виявлення візуальних підробок, коли зловмисники накладають логотипи відомих

компаній на QR-коди, щоб викликати довіру. Згорткові нейронні мережі (CNN) можуть аналізувати зображення QR-кодів і виявляти аномалії, такі як нестандартні кольори, розмитість або наявність додаткових графічних елементів. Наприклад, якщо QR-код містить логотип банку, але його візуальні характеристики відрізняються від офіційних зразків, алгоритм може класифікувати його як потенційно шкідливий.

Машинне навчання також використовується для аналізу поведінки вебсторінок у реальному часі. Наприклад, моделі можуть відстежувати, як сторінка реагує на взаємодію з користувачем (натискання кнопок, введення даних), і виявляти підозрілі дії, такі як спроби відправки даних на невідомі ффсервери. Це дозволяє виявляти фішингові сайти, які імітують легальні сервіси, але мають відмінності у поведінці. Незважаючи на свою ефективність, ML-методи вимагають великих обсягів даних для навчання та оновлення [8].

Гібридні системи захисту є найбільш ефективними системами захисту від фішингових QR-кодів поєднують кілька методів аналізу, що дозволяє компенсувати недоліки окремих підходів. Гібридні системи зазвичай включають статичний аналіз URL, динамічну перевірку в пісочниці та машинне навчання для остаточного прийняття рішення. Наприклад, коли користувач сканує QR-код, система спочатку перевіряє URL на належність до чорних списків і аналізує його лексичні ознаки. Якщо результат неоднозначний, URL відкривається в пісочниці для динамічного аналізу поведінки вебсторінки. Одночасно ML-модель оцінює контекст і семантику сторінки, щоб виявити приховані ознаки фішингу.

Одним із прикладів таких гібридних систем є Trend Micro QR Scanner, який поєднує чорні списки, емуляцію мобільного браузера та власні алгоритми машинного навчання для комплексної перевірки. Це дозволяє не тільки блокувати відомі фішингові сайти, але й виявляти нові, ще не додані до баз даних загрози. Іншим прикладом є інтеграція QR-сканерів із сервісами на кшталт VirusTotal, які агрегують дані з різних антивірусних рушіїв і чорних списків, що підвищує ймовірність виявлення шкідливих посилань.

Перспективним напрямком у розвитку гібридних систем є використання блокчейн-технологій для верифікації QR-кодів. Наприклад, легальні QR-коди

можуть підписуватися цифровим підписом, що дозволить перевіряти їх справжність без необхідності глибокого аналізу. Також активно розвивається інтеграція QR-сканерів із IoT-пристроями, такими як розумні камери, які можуть автоматично сканувати та перевіряти коди в громадських місцях. Це особливо актуально для запобігання масовим фішинговим атакам, коли зловмисники замінюють QR-коди на рекламних стендах або в меню ресторанів [9]. Порівння цих методів наведено в таблиці 1.1.

Таблиця 1.1 – Порівння методів аналізу QR-кодів

| Характеристика | Статичний аналіз | Динамічний аналіз | Машинне навчання (ML) | Гібридні системи |
|---|---|---|---|---|
| Вхідні дані | Графічне зображення QR, текст, URL | Відкриття URL у віртуальному у середовищі | Текст URL, поведінка вебсторінки, зображення QR | Всі дані з інших методів |
| Перевірка URL | Синтаксис, чорні списки, репутація, короткі посилання | Аналіз поведінки сторінки, мережеві запити, скрипти | Класифікація URL з NLP, аналіз семантики | Послідовна/паралельна перевірка URL всіма методами |
| Складність | Відносно низька, швидка перевірка | Висока, вимагає ресурсів і часу | Середня/висока, потребує навчання та оновлень | Висока, комплексна реалізація |
| Переваги | Швидкий початковий скринінг, виявлення аномалій у структурі і URL | Виявлення складних схем, прихованих скриптів, поведінки | Виявлення нових шаблонів і контексту, гнучкість | Висока точність, компенсація слабких сторін окремих методів |
| Обмеження | Не виявляє поведінкові атаки, залежить від баз даних | Вимагає багато ресурсів, довший час перевірки | Залежність від якості даних і навчання моделей | Висока складність впровадження, вартість |

## 1.5 Утиліти для аналізу посилань

Google Safe Browsing – це важлива служба безпеки, розроблена командою безпеки Google, метою якої є виявлення та блокування доступу до небезпечних веб-сайтів у глобальній мережі Інтернет. Ця служба працює як онлайн-щит, що захищає користувачів від потенційних кіберзагроз, таких як шкідливе програмне забезпечення (малваре), фішинг, шахрайські сайти, а також інші типи небезпечного контенту, які можуть завдати шкоди або викрасти особисті дані.

Google Safe Browsing створена для того, щоб автоматично виявляти шкідливі веб-ресурси та негайно повідомляти про них користувачів і власників сайтів. Завдяки постійному оновленню своєї бази даних зловмисних сайтів, служба ефективно відстежує нові загрози та дозволяє уникнути випадкового переходу на небезпечні сторінки. Сервіс працює на основі складних алгоритмів, машинного навчання та аналізу величезних масивів даних з усього світу, що робить його одним із найнадійніших інструментів у сфері онлайн-безпеки.

Основна функція сервісу – це генерація попереджень користувачам веб-браузерів і додатків, які підтримують цю технологію. Коли користувач намагається відкрити сайт, що визнаний небезпечним, система показує спеціальне сповіщення або блокуючу сторінку з детальним поясненням, що цей ресурс може містити загрозу (наприклад, фішингову атаку або зараження шкідливим ПЗ). Це допомагає захистити користувачів від потенційних втрат – фінансових, конфіденційних або репутаційних.

Google Safe Browsing значно підвищує рівень безпеки в інтернеті, забезпечуючи захист понад п’яти мільярдів пристроїв – це величезна аудиторія, що охоплює різні платформи: мобільні телефони, комп’ютери, планшети, браузери та інші додатки. Впровадження такої системи дає змогу знизити кількість успішних фішингових атак і шкідливих заражень, зберігаючи при цьому конфіденційність і безпеку користувачів.

Крім безпосереднього захисту користувачів є позитивний вплив на репутацію веб-сайтів. Якщо сайт визнано безпечним, це підвищує рівень довіри до нього та його рейтинг у результатах пошуку Google. Навпаки, сайти, які потрапляють до

чорного списку через шкідливу активність, отримують попередження в результатах пошуку, що може призвести до зниження трафіку і репутаційних втрат для власників таких ресурсів.

Безпечний перегляд Google працює в реальному часі, регулярно оновлюючи базу даних потенційно шкідливих сайтів. Ці оновлення відбуваються автоматично і поширюються на всі пристрої, які використовують сервіс. Таким чином, користувачі отримують максимально актуальну інформацію про загрози.

Google також публікує регулярні звіти про прозорість, де розкриває статистику попереджень, які система показала користувачам по всьому світу. Ці звіти допомагають підвищувати обізнаність про кіберзагрози і спонукають власників веб-сайтів удосконалювати заходи безпеки.

Впровадження Google Safe Browsing не тільки сприяє безпеці користувачів, але й позитивно впливає на довіру до веб-сайту та рейтинг у пошукових системах. Пріоритет безпеки в Інтернеті відповідає прагненню Google створити безпечніший Інтернет для користувачів у всьому світі. Це важливий компонент відповідальної веб-розробки та сприяє зміцненню довіри між користувачами та операторами веб-сайтів, схема роботи показана на рис. 1.4.

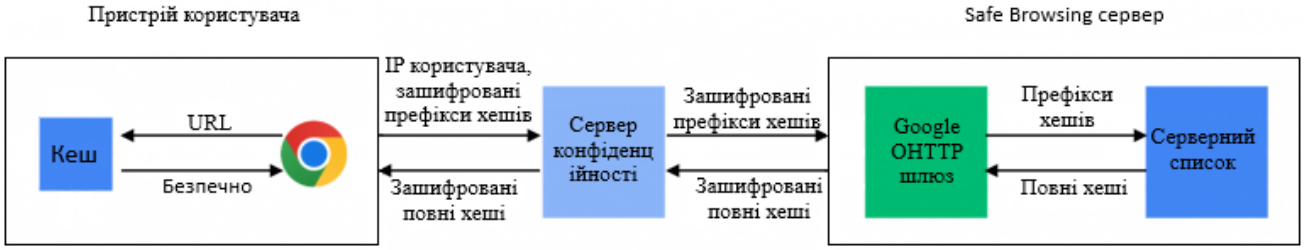



Рисунок 1.4 – Схема роботи Google Safe Browsing

Приблизно п’ять мільярдів пристроїв використовують технологію Google Safe Browsing. Коли системи Google визначають сайт як потенційно шкідливий, Безпечний перегляд запускає попередження для користувачів. Ці попередження призначені для того, щоб запобігти відвідування шкідливих сайтів і допомогти їм залишатися в безпеці в Інтернеті.

Google Safe Browsing інтегрована у популярні веб-браузери, такі як Google Chrome, Mozilla Firefox, Apple Safari та інші, а також у мобільні додатки і

операційні системи. Коли користувач або додаток робить запит на завантаження сторінки, запит проходить перевірку на наявність у базі загроз. Якщо сайт вважається небезпечним, браузер або додаток відобразить сторінку із застереженням, детально пояснюючи потенційну небезпеку.

Коли користувач веб-переглядача або програми з підтримкою безпечного перегляду намагається отримати доступ до небезпечного вмісту в Інтернеті, він побачить сторінку із застереженням, яка пояснює, що вміст, до якого він намагається отримати доступ, може бути шкідливим. Коли сайт, який безпечний перегляд визначив як шкідливий, з'являється в результатах пошуку Google, показується попередження поруч із цим сайтом у результатах [10].

VirusTotal – цей сервіс перевіряє елементи за допомогою понад 70 антивірусних сканерів і служб блокування URL/доменів, а також безліч інструментів для вилучення сигналів із досліджуваного вмісту. Будь-який користувач може вибрати файл зі свого комп'ютера за допомогою браузера та надіслати його до VirusTotal. VirusTotal пропонує низку методів надсилання файлів, включаючи основний загальнодоступний веб-інтерфейс, засоби завантаження з робочого столу, розширення браузера та програмний API. Веб-інтерфейс має найвищий пріоритет сканування серед загальнодоступних методів подання. Надсилання може бути написано будь-якою мовою програмування за допомогою публічного API на основі HTTP. Як і у випадку з файлами, URL-адреси можна надсилати кількома різними засобами, включаючи веб-сторінку VirusTotal, розширення браузера та API.

Після надсилання файлу чи URL-адреси основні результати надаються суб'єкту, що подав, а також між партнерами з перевірки, які використовують результати для вдосконалення власних систем. Таким чином, надсилаючи файли, URL-адреси, домени тощо до VirusTotal, ви сприяєте підвищенню глобального рівня IT-безпеки.

Цей основний аналіз також є основою для кількох інших функцій, у тому числі спільноти VirusTotal: мережі, яка дозволяє користувачам коментувати файли та URL-адреси та обмінюватися нотатками один з одним. VirusTotal може бути корисним для виявлення зловмисного вмісту, а також для ідентифікації

помилкових спрацьовувань – звичайних і нешкідливих елементів, визначених як шкідливі одним або декількома сканерами.

VirusTotal є безкоштовним для кінцевих користувачів для некомерційного використання. Хоча розробник працюють з механізмами, що належать багатьом різним організаціям, VirusTotal не розповсюджує та не рекламує жодні з цих механізмів третіх сторін.

Зведені дані VirusTotal є результатом багатьох різних антивірусних механізмів, сканерів веб-сайтів, інструментів аналізу файлів і URL-адрес, а також внесків користувачів. Інструменти визначення характеристик файлів і URL-адрес, які ми збираємо, охоплюють широкий спектр цілей: евристичні механізми, завідомо несправні підписи, вилучення метаданих, ідентифікація шкідливих сигналів тощо.

Звіти про сканування, створені VirusTotal, надаються загальнодоступній спільноті VirusTotal. Користувачі можуть залишати коментарі та голосувати, чи є певний вміст шкідливим. Таким чином користувачі допомагають поглибити колективне розуміння спільнотою потенційно шкідливого вмісту та виявляти хибні спрацьовування (тобто нешкідливі елементи, визначені одним або кількома сканерами як шкідливі).

У діаграмі показано загальну кількість попереджень у веб-переглядачі/програмі, які відображаються користувачам, і загальну кількість результатів пошуку, у яких відображаються попередження за тиждень на рис.1.5.

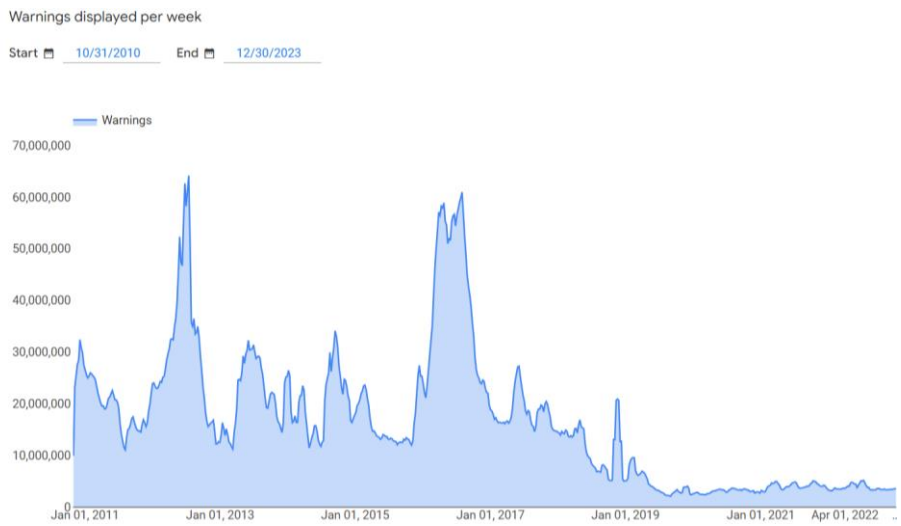



Рисунок 1.5 – Кількість попереджень, що відображаються за тиждень

Вміст надісланих файлів або сторінок також може надаватися преміум-клієнтам VirusTotal. Корпус файлів, створений у VirusTotal, надає фахівцям з кібербезпеки та розробникам продуктів безпеки цінну інформацію про поведінку нових кіберзагроз і зловмисного програмного забезпечення. Завдяки нашим комерційним пропозиціям преміум-послуг VirusTotal надає кваліфікованим клієнтам і антивірусним партнерам інструменти для виконання комплексного пошуку на основі критеріїв для виявлення зразків шкідливих файлів і доступу до них для подальшого вивчення. Це допомагає організаціям виявляти та аналізувати нові загрози та створювати нові засоби пом’якшення та захисту.

У деяких випадках сканування веб-сайтів виконується шляхом запиту до баз даних постачальників, які надані VirusTotal і зберігаються в наших приміщеннях, а в інших випадках через запити API до рішення антивірусної компанії. Таким чином, щойно певний учасник заносить URL-адресу в список блокувань, це негайно відображається у вердиктах для користувачів.

VirusTotal не тільки повідомляє вам, чи виявило дане антивірусне рішення надісланий файл як шкідливий, але також відображає мітку виявлення кожного механізму (наприклад, I-Worm.Allaple.gen). Те ж саме стосується сканерів URL-адрес, більшість із яких розрізнятимуть сайти зі зловмисним програмним забезпеченням, фішингові сайти, підозрілі сайти тощо. Деякі механізми нададуть додаткову інформацію, чітко вказуючи, чи належить дана URL-адреса до певної бот-мережі, на який бренд націлений певний фішинговий сайт тощо[11].

PhishTank – це безкоштовна спільнота, де користувачі можуть надсилати, перевіряти, відстежувати та ділитися фішинговими даними. Це база даних фішингових URL-адрес, які спільнота PhishTank підтвердила як шахрайські. Ці фішингові URL-адреси зазвичай представляють сайти, які намагаються викрасти інформацію користувача, таку як імена користувачів, паролі та дані кредитної картки, змушуючи користувачів повірити, що вони отримують доступ до законної служби.

PhishTank працює за допомогою системи голосування. Коли користувач надсилає ймовірну фішингову URL-адресу, вона стає ключовою частиною

«фішингового танку». Потім інші користувачі можуть «проголосувати», перевіряючи або оскаржуючи, чи веб-сайт справді є спробою фішингу. Коли URL-адреса отримує достатню кількість дійсних голосів, вона оголошується підтвердженим фішинговим сайтом. Згодом ця URL-адреса потрапляє в базу даних фішингових загроз PhishTank і стає доступною для будь-кого, полегшуючи структуру потужних систем захисту від таких небезпек.

Ефективне використання PhishTank передбачає участь у екосистемі шляхом надсилання, перевірки та використання бази даних. Зареєструвавшись у PhishTank, користувачі можуть надсилати підозрілі URL-адреси на перевірку. Як користувач спільноти, вони також можуть голосувати за пропозиції інших користувачів, додаючи спільноті загальну силу кібербезпеки.

З іншого боку, організації або окремі особи, які бажають використовувати базу даних PhishTank, можуть інтегрувати її зі своєю існуючою інфраструктурою безпеки. Для інтеграції надається API, що дозволяє отримати доступ до оновлених даних про фішинг для цілей блокування або дослідження.

Хоча PhishTank є ключовим компонентом комплексної стратегії кібербезпеки, він не повинен замінювати інші заходи безпеки. Користувачі повинні продовжувати практикувати хороші звички безпеки в Інтернеті, наприклад не натискати підозрілі посилання, використовувати надійні паролі та оновлювати програмне забезпечення системи. Використання можливостей PhishTank в поєднанні з іншими заходами безпеки є оптимальною комбінацією для забезпечення безпеки в Інтернеті. Це дозволяє користувачам Інтернету діяти як колективний захист від спроб фішингу, перетворюючи кібербезпеку з підходу, що базується на продукті, на більш надійну стратегію, орієнтовану на спільноту. Розуміючи роботу та цінність PhishTank, ми зміцнюємо наші знання про фішингові загрози, покращуємо захист і робимо цифровий світ безпечнішим. PhishTank справді є супердержавою в галузі кібербезпеки, яка перетворює нас із пасивних користувачів на активних захисників у

величезному цифровому просторі. За даними PhishTank за перші п’ять місяців 2025 року спільнотою було перевірено та підтверджено понад 450 000 нових фішингових URL-адрес, що свідчить про високу активність шахраїв у кіберпросторі. Завдяки швидкому розпізнаванню нових фішингових сайтів PhishTank допоміг запобігти потенційним втратам фінансових коштів і конфіденційних даних у понад 3 мільярдах користувачів по всьому світу. Інтеграції з різними платформами та антивірусними рішеннями збільшилися на 35% у порівнянні з минулим роком, що свідчить про зростаюче визнання значення PhishTank серед фахівців з безпеки [12]. Порівння цих утиліт наведено в табл. 1.2.

Таблиця 1.2 – Порівння утиліт аналізу посилань

| Критерій | Google Safe Browsing | VirusTotal | PhishTank |
|---|---|---|---|
| Призначення | Виявлення небезпечних сайтів (шкідливе ПЗ, фішинг тощо) і попередження користувачів | Перевірка URL/файлів понад 70 антивірусами та інструментами | Виявлення та підтвердження фішингових URL-адрес спільнотою |
| Джерела даних | Власні алгоритми Google, база шкідливих сайтів | Антивірусні сканери, служби блокування URL/доменів, користувацькі надсилання | Користувачі спільноти, які голосують за достовірність фішингу |
| Метод перевірки | Автоматичний аналіз і попередження у браузері або результатах пошуку | Сканування URL або файлу з використанням декількох механізмів одночасно | Голосування користувачів за фішинговість URL після подання |
| Платформа/Доступ | Вбудовано у браузери (Chrome, Firefox), API | Веб-інтерфейс, API, розширення, десктопні програми | Веб-портал і API для перевірки/подання URL |
| Оновлення бази | Постійно оновлюється командою Google | Динамічне оновлення антивірусними постачальниками, постійне оновлення сигнатур | Оновлення бази на основі голосів користувачів |

## 1.6 Фішингові атаки останніх років

Фішингові атаки вже давно перестали бути поодинокими інцидентами – вони стали інструментом систематичного тиску на державні установи, бізнес і окремих користувачів. Останні роки засвідчили зростання кількості й складності таких атак, у яких зловмисники використовують дедалі витонченіші методи впливу на людей і технології. У цьому розділі розглянуто приклади найбільш резонансних фішингових кампаній, що мали значний вплив на різні сектори – від урядових структур до міжнародних корпорацій.

1) Фішингова кампанія проти держорганів України (2023). У серпні 2023 року Державна служба спеціального зв'язку та захисту інформації України повідомила про масштабну фішингову кампанію, спрямовану на українські державні установи та критичну інфраструктуру. Атаку здійснювало хакерське угруповання UAC-0056 (також відоме як UNC2589), яке пов'язують із Російською Федерацією. Зловмисники розсилали електронні листи з темою «Оновлення безпеки» та вкладенням у вигляді документа Word або архіву, що містив шкідливе ПЗ, зокрема новий варіант шкідника SmokeLoader. Після відкриття вкладення та активації макросів на комп'ютері жертви встановлювався бекдор для віддаленого доступу. Метою атаки було шпигування, збирання документів та облікових даних співробітників урядових установ. Кампанія була частиною ширшої кібервійни, яка триває з початку повномасштабного вторгнення Росії в Україну [13].
2) Атака на Change Healthcare (США, 2024). У лютому 2024 року компанія Change Healthcare, один із лідерів у сфері медичних технологій у США, стала жертвою фішингової атаки, внаслідок якої було скомпрометовано облікові дані співробітників. Зловмисники з угруповання ALPHV/BlackCat використовували методи соціальної інженерії для отримання доступу до внутрішніх систем компанії. Після отримання облікових даних вони змогли проникнути в інфраструктуру медичних послуг, що призвело до серйозних

збоїв у роботі компанії, включаючи порушення обробки рахунків та страхових вимог. Внаслідок цього, компанія UnitedHealth була змушена виплатити викуп на суму 22 мільйони доларів для відновлення систем, однак хакери так і не повернули викрадені дані [14].

3) Фішингова атака на Pepco Group (Європа, 2024). В лютому 2024 року європейський рітейлер Pepco Group став жертвою фішингової атаки, в результаті якої компанія втратила 15,5 мільйонів євро. Зловмисники використали методи соціальної інженерії, відправляючи фальшиві електронні листи, які імітували внутрішню корпоративну комунікацію. Мета цих листів полягала в тому, щоб переконати співробітників компанії здійснити переказ коштів на шахрайські рахунки.
4) Атака на MGM Resorts і Caesars Entertainment (США, 2023). У вересні 2023 року хакерське угруповання Scattered Spider провело фішингові атаки на дві великі готельні мережі – MGM Resorts та Caesars Entertainment. Зловмисники здійснили атаки за допомогою соціальної інженерії, видаючи себе за працівників служби підтримки, щоб отримати доступ до корпоративних облікових записів. В результаті атак, Caesars Entertainment була змушена заплатити викуп у розмірі 15 мільйонів доларів, а MGM Resorts зазнала масштабних збоїв у роботі своїх систем.
5) Атака на OpenAI (США, 2024). У жовтні 2024 року американська компанія OpenAI, відома своїми передовими розробками у сфері штучного інтелекту, стала мішенню кібератаки, організованої китайським хакерським угрупованням, відомим під назвою SweetSpecter. Це угруповання вже раніше фігурувало у звітах про кібершпигунство, спрямоване на західні технологічні компанії. Хакери розгорнули цільову фішингову кампанію, націлену на співробітників OpenAI. Вони розсилали спеціально підготовлені шкідливі електронні листи, які містили вкладення із зловмисним програмним забезпеченням. Після відкриття таких файлів зловмисники отримували віддалений доступ до внутрішніх мереж і систем компанії. Це дозволяло їм

потенційно переглядати, копіювати або вивозити конфіденційну інформацію. Основною метою атаки було викрадення інтелектуальної власності, пов'язаної з найновішими дослідженнями і моделями OpenAI у сфері штучного інтелекту. Така інформація має не лише високу економічну, але й стратегічну цінність, зокрема в контексті змагання між США та Китаєм за технологічне лідерство у сфері AI [15].

6) Атака через Google Ads на користувачів Binance (2022). У 2022 році кіберзлочинці використали рекламну платформу Google Ads для просування фішингових копій популярних сайтів криптовалютних бірж, зокрема Binance. Зловмисники створювали підроблені рекламні оголошення, які зазвичай з’являлись вище за офіційні результати пошуку. Користувачі, натискаючи на такі оголошення, перенаправлялися на фальшиві сторінки входу, що візуально майже не відрізнялися від справжніх. На цих підроблених сайтах жертви вводили свої облікові дані, які миттєво передавались зловмисникам. Внаслідок цього було викрадено мільйони доларів у криптовалюті, а також виявлено численні спроби подальшого несанкціонованого доступу до акаунтів [16].
7) Масова фішингова кампанія проти користувачів Office 365 (2022–2023). З 2022 по 2023 роки фішингові атаки на користувачів Microsoft Office 365 були націлені в основному на корпоративний сектор. Зловмисники розсилали підроблені листи, які імітували повідомлення від Microsoft, з метою викрадення паролів до облікових записів користувачів. В результаті цієї кампанії було скомпрометовано десятки тисяч акаунтів, що призвело до витоку конфіденційної інформації, порушення робочих процесів та значних фінансових втрат для компаній [17].

У табл. 1.3 наведена детальна оцінка ризику фішингового посилання, сформована на основі аналізу статичних ознак. Ці ознаки включають ключові параметри, які допомагають ідентифікувати потенційно шкідливі URL-адреси.

Таблиця 1.3 – Оцінка ризику фішингового посилання на основі статичних ознак

| Критерій | Приклад значення | Оцінка (1-10) | Пояснення |
|---|---|---|---|
| Структура URL | Використання кирилічних символів у домені | 7 | Можливість візуального обману |
| Репутація домену | Домен у чорному списку VirusTotal | 8 | Високий рівень ризику |
| Наявність редиректів | Три перенаправлення перед кінцевою сторінкою | 6 | Складна схема маскування |
| Схожість із відомими сайтами (NLP) | URL містить слова “login”, “bank”, “verify” | 9 | Типова ознака фішингової активності |
| Геолокація сервера | Незвичайна країна для даного типу сервісу | 5 | Може свідчити про аномалію |

0-3.9 - Низький ризик (можна дозволити доступ)

4.0-6.9 - Середній ризик (рекомендується попередження)

7.0-10 - Високий ризик (автоматичне блокування або ручна перевірка)

## 1.7 Постановка завдання

У зв’язку з активним впровадженням QR-кодів у цифровий простір та щоденне використанням їх для доступу до веб-ресурсів, оплати, автентифікації, реклами та інших сервісів, зростає ризик використання таких кодів як вектора для фішингових атак. Завдяки своїй зручності та універсальності QR-коди стрімко інтегрувалися у взаємодію між користувачем та цифровими сервісами. Проте їхня популярність також зробила їх привабливою ціллю для зловмисників, які використовують QR-коди як засіб доставки фішингових посилань, що ведуть до підроблених веб-сайтів або заражених ресурсів.

Проблема полягає в тому, що користувач зазвичай не бачить, куди саме веде QR-код до моменту відкриття посилання. Це створює ідеальне середовище для атак типу «QR-фішинг» – коли шкідливе посилання вбудовується в код, і нічого не підозрюючий користувач відкриває сайт, який імітує легітимний ресурс. Такі атаки можуть призводити до викрадення облікових даних, особистої інформації, доступу до фінансових акаунтів або встановлення шкідливого програмного забезпечення на пристрій користувача.

Постає потреба у створенні простого у використанні та ефективного засобу для перевірки QR-кодів на наявність фішингових загроз – інструменту, який дозволить звичайному користувачеві чи спеціалісту з безпеки попередньо перевірити URL-адресу, закодовану в зображенні, без її відкриття.

З урахуванням цього, метою бакалаврської кваліфікаційної роботи є створення програмного засобу, що дозволяє здійснювати такі дії:

1) розпізнавати QR-код із графічного файлу  та витягувати з нього закодоване посилання;
2) перевіряти отриману URL-адресу за допомогою кількох незалежних сервісів;
3) отримувати як стислу оцінку безпечності, так і розгорнуті технічні деталі;
4) автоматично зберігати історію перевірок для подальшого аудиту;
5) забезпечувати зручний інтерфейс користувача.

## 1.8 Висновки до розділу

У цьому розділі було здійснено комплексний аналіз проблеми фішингових атак, зосереджений насамперед на новітніх формах їх реалізації через QR-коди. Детально розглянуто еволюцію фішингових технологій, їхній вплив на інформаційну безпеку, а також специфіку фішингових кампаній, які використовують методи соціальної інженерії в поєднанні з графічними елементами, зокрема QR-кодами.

Фішинг, як одна з найдавніших і водночас найдинамічніших форм кіберзагроз, не втрачає своєї актуальності, а навпаки – набуває дедалі складніших і витонченіших форм. Сучасні зловмисники активно впроваджують технології штучного інтелекту, автоматизацію масових розсилок, генерацію підроблених сторінок і підмінну цифрових елементів інтерфейсу, що значно ускладнює виявлення загроз. У цьому контексті особливої уваги заслуговує QR-фішинг – форма атаки, при якій шахраї використовують QR-коди як засіб маскування шкідливих URL-адрес.

Зазначене явище набуло широкого поширення з огляду на стрімке зростання популярності QR-кодів у повсякденному житті – в оплатах, маркетингових кампаніях, онлайн-реєстраціях, автентифікації тощо. Саме ця масовість і, водночас, відсутність очевидного контролю з боку користувача створюють сприятливі умови для фішингових атак. QR-код, на відміну від звичайного гіперпосилання, не дозволяє попередньо переглянути адресу безпосередньо — він відкриває її лише після сканування, що мінімізує шанси на візуальне розпізнавання шахрайства. Крім того, більшість QR-кодів використовуються з мобільних пристроїв, де часто бракує потужних засобів захисту, порівняно з десктопними браузерами або спеціалізованими корпоративними системами.

У результаті аналізу встановлено, що хоча ризики QR-фішингу є реальними та значними, існують ефективні способи виявлення та протидії таким загрозам. Зокрема, були розглянуті і проаналізовані сучасні інструменти, що дозволяють перевіряти вміст QR-кодів ще до їх відкриття – серед них Google Safe Browsing, VirusTotal, URLScan.io. Ці сервіси забезпечують багаторівневу перевірку URL-адрес, що дозволяє своєчасно виявляти потенційно небезпечні або вже підтверджені фішингові сайти. Додатково, деякі з них інтегрують механізми машинного навчання, евристичного аналізу та взаємодії з глобальними базами даних про загрози, що суттєво підвищує точність і швидкість реагування на підозрілі елементи.

# 2 МЕТОД ДЛЯ ПЕРЕВІРКИ QR-КОДІВ НА НАЯВНІСТЬ ФІШИНГОВИХ ЗАГРОЗ

## 2.1 Метод для тестування QR-кодів на наявність фішингових загроз

У рамках розробки програмного засобу для перевірки QR-кодів на наявність фішингових загроз було обрано комбінований підхід, який поєднує кілька етапів перевірки через використання зовнішніх аналітичних сервісів для підвищення надійності та точності результатів. Цей метод передбачає поетапну обробку вхідного зображення, розпізнавання URL із QR-коду, його валідацію, а також подальшу перевірку на можливі фішингові загрози. Для досягнення більш високої надійності перевірки використовуються три незалежні сервіси, які забезпечують комплексну оцінку безпеки URL: Google Safe Browsing, VirusTotal та URLScan.io.

1) Зчитування QR-коду. Для цього використовується бібліотека, здатна розпізнавати та витягувати текстову інформацію з візуальних елементів, таких як зображення QR-кодів. Коли користувач завантажує зображення QR-коду через інтерфейс програми, зображення передається на декодування, і якщо зчитаний текст є URL, він надалі обробляється для перевірки на можливі загрози. Окрім того, якщо QR-код не містить коректного URL, програма видасть помилку, сповіщаючи користувача.
2) Валідація URL, що міститься в QR-коді. Для цього застосовується регулярний вираз, який перевіряє, чи є введене посилання правильним і чи відповідає воно стандартам URL-формату. Якщо URL має помилки у структурі або не відповідає вимогам, програма виводить повідомлення про некоректність введеного посилання, запобігаючи надходженню шкідливих даних у подальші перевірки.
3) Перевірка за трьома зовнішніми аналітичними модулями для оцінки безпеки посилання:

- Google Safe Browsing – цей модуль перевіряє URL на наявність потенційних загроз, таких як фішинг, шкідливе програмне забезпечення або небажане ПЗ. За допомогою цього API здійснюється перевірка URL на базі інформації про відомі шкідливі сайти, які постійно оновлюються. Система Google Safe Browsing працює на принципі зіставлення введеного URL з базою даних небезпечних ресурсів, повідомляючи користувача, чи є сайт небезпечним або безпечним.
- VirusTotal – цей модуль надає детальну перевірку URL на основі аналізу безпеки за допомогою десятків різних антивірусних движків та інших безпекових інструментів. Сканування посилання на VirusTotal дає змогу оцінити, чи містить сайт шкідливий контент, за допомогою статистики виявлених загроз різними антивірусними та безпековими сервісами. Результат цієї перевірки дає як загальну інформацію про небезпеку ресурсу, так і більш детальну інформацію щодо виявлених загроз.
- URLScan.io – цей модуль аналізує сайт та надає додаткову інформацію про його активність, включаючи спостереження за мережевими запитами, використовуваними ресурсами та іншими поведінковими характеристиками. Якщо ресурс містить елементи, що можуть бути пов'язані з фішингом або іншими шкідливими діями, URLScan.io надає відповідний статус та рекомендації для користувача.

4) Після перевірок усі сервіси надають результати, які підсумовуються в одну звітну форму. Кожен з результатів перевірки (Google Safe Browsing, VirusTotal та URLScan.io) надається з коротким описом ситуації, що дозволяє користувачу одразу зрозуміти, чи є URL небезпечним.

Таким чином, даний метод дозволяє забезпечити комплексну перевірку QR-кодів і забезпечити високу надійність тестування на фішингові загрози, а також допомогти користувачам виявляти шкідливі посилання в QR-кодах з високим рівнем точності.

Блок схема роботи програмного засобу зображена на рис. 2.1

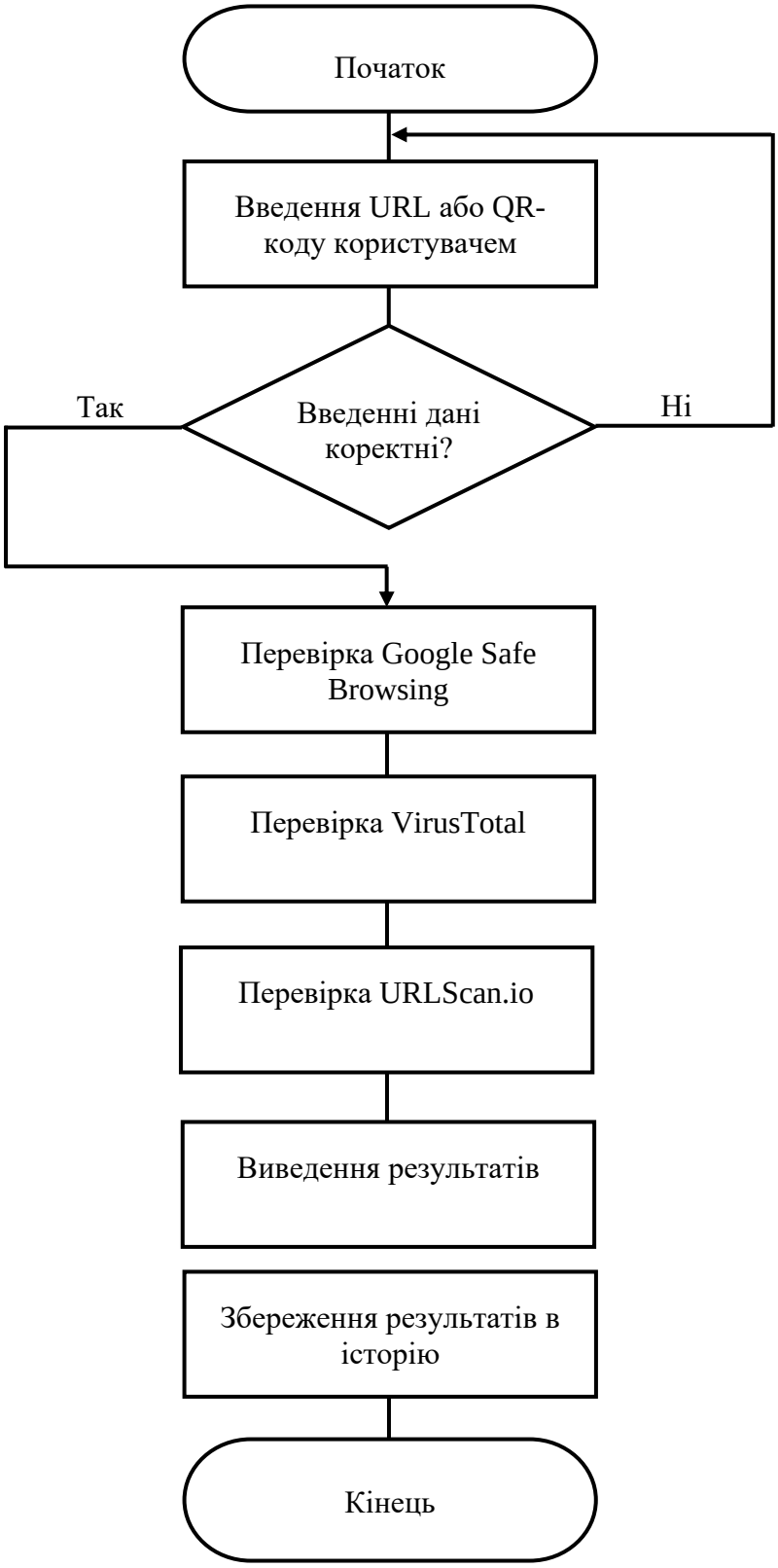



Рисунок 2.1 – Блок схема роботи програмного засобу

Користувач може вставити посилання або QR-код у спеціально призначене поле введення, використовуючи контекстне меню. Після того як дані були

вставлені, програма автоматично визначає тип введеного контенту. Якщо це QR-код, програма використовує бібліотеку для зчитування зображень, щоб розпізнати його вміст і витягти URL. Якщо введений текст є звичайною URL-адресою, програма переходить до наступного етапу перевірки без додаткової обробки.

Коли URL успішно отримано, програма здійснює його первинну валідацію. Вона перевіряє, чи правильно побудована адреса, використовуючи регулярні вирази для перевірки структури посилання. Це включає перевірку на наявність таких елементів, як http:// або https://, а також перевірку на правильність інших компонентів URL, таких як домен, порти та шляхи. Якщо URL не відповідає стандартам або містить помилки, програма інформує користувача про це, і подальші кроки перевірки не виконуються. Користувач отримує повідомлення про помилку у форматі, який дозволяє зрозуміти, що саме в URL є некоректним.

У випадку, коли URL є коректним, програма переходить до основного етапу – перевірки на наявність потенційних загроз за допомогою трьох незалежних аналітичних сервісів. Кожен з цих сервісів проводить своєрідну перевірку URL на предмет фішингових сайтів, шкідливих програм, підозрілих поведінкових патернів або іншої небезпечної активності, пов’язаної з посиланням.

Першим виконується запит до сервісу Google Safe Browsing, який аналізує URL і перевіряє, чи є посилання в базах даних, що містять відомості про небезпечні ресурси. Якщо посилання виявляється потенційно небезпечним, користувач отримує відповідне повідомлення, що посилання було заблоковано через виявлені фішингові або шкідливі елементи.

Наступним сервісом для перевірки є VirusTotal, який використовує величезну кількість антивірусних та інших безпекових інструментів для сканування URL. Він дає додаткову оцінку безпеки на основі численних аналізів з різних джерел. Якщо сервіс знаходить загрози, користувач отримує деталізовану інформацію про потенційні небезпеки, які можуть виникнути при відвідуванні цього ресурсу.

Завершальною перевіркою є URLScan.io, який сканує посилання на наявність шкідливих або небезпечних елементів, зокрема на наявність скриптів, які можуть

бути використані для фішингу або для компрометації системи. Він також перевіряє репутацію вебсайту, з'ясовує, чи є на ньому інші підозрілі або шахрайські елементи, такі як приховані редиректи, атаки через JavaScript тощо.

Після завершення перевірок усіх трьох сервісів програма надає користувачеві підсумковий результат. Це може бути повідомлення про те, що URL є безпечним (якщо жоден із сервісів не виявив загроз), або попередження про можливу небезпеку (якщо хоча б один з сервісів знайшов загрози). Користувач отримує точну кількість загроз, виявлених кожним з сервісів, що дозволяє йому більш детально оцінити ризики. Якщо загрози незначні, посилання позначається як «потенційно небезпечне», якщо загрози серйозні – як «небезпечне».

Додатково програма дозволяє користувачеві переглядати історію перевірок, що дає змогу повернутись до результатів попередніх перевірок, зберігаючи всю важливу інформацію для подальшого аналізу. Це дозволяє зберігати всю інформацію про проведені перевірки і дає користувачеві змогу знову перевірити посилання за потреби. Користувач може зберігати записи перевірок, що допомагає в подальшій роботі з URL або для подальшого аналізу безпеки.

Об'єднання трьох методів перевірки URL та QR-кодів в єдину програму обумовлено їхньою ефективністю та широко застосовуваними у реальних ситуаціях. Це включає перевірку через Google Safe Browsing для виявлення відомих загроз, аналіз через VirusTotal для більш глибокої перевірки репутації URL та сканування через URLScan.io для пошуку підозрілої активності на веб-сторінці. Завдяки модульній архітектурі програми додавання нових методів перевірки. У майбутньому це дозволить розширювати функціонал програми, включаючи інтеграцію нових аналітичних сервісів або можливість віддаленого тестування без необхідності змінювати існуючий код.

Після того, як користувач вводить URL або вставляє його за допомогою контекстного меню, програма автоматично ініціює процес перевірки і надає користувачу візуальний зворотний зв'язок. Кожен метод перевірки – чи то Google Safe Browsing, VirusTotal, чи URLScan – реалізує свою логіку перевірки, при цьому

на кожному етапі користувач отримує чітку інформацію про поточний стан процесу. Кожен з цих методів забезпечує специфічну перевірку, яка орієнтована на різні аспекти безпеки та допомагає виявити потенційно небезпечні чи зловмисні посилання.

Для кожного з методів, після натискання відповідної кнопки, програма автоматично ініціює процес перевірки, виводячи на екран поетапні логи з описом ходу роботи кожного методу. Наприклад, при використанні Google Safe Browsing програма спочатку показує повідомлення про початок звернення до сервісу, потім виводить інформацію про статус перевірки та надає результат. Аналогічно працюють і інші методи, де результат перевірки виводиться по завершенню процесу в терміналі, а сам результат може бути занесений до історії перевірок. Програма підсумовує результати всіх перевірок і виводить інформацію, яка дає чітке уявлення про загальний рівень небезпеки URL.

Якщо більшість перевірок виявляє загрози, ресурс отримає високий рівень небезпеки. У випадку, якщо загрози знайдено лише в одній чи двох перевірках, результат буде вказувати на помірну небезпеку. Таким чином, користувач отримує зрозумілу та конкретну інформацію про рівень безпеки того чи іншого ресурсу. Використання цих трьох методів забезпечує користувача надійним і швидким інструментом для перевірки безпеки URL та QR-кодів. Вибір саме цих сервісів обґрунтований їх високою ефективністю, доступністю API та здатністю швидко надавати точний аналіз, що гарантує надійний захист користувачів при використанні інтернет-ресурсів.

## 2.2 Модуль тестування Google Safe Browsing

Google Safe Browsing – це потужний сервіс безпеки, створений для виявлення і попередження про шкідливі або небезпечні сайти. Його мета – забезпечити безпеку користувачів Інтернету, запобігаючи доступу до сайтів, що містять шкідливі програми, фішингові елементи чи інші загрози. Система використовує

спеціалізовану інфраструктуру для сканування веб-сайтів і виявлення відомих небезпечних ресурсів, забезпечуючи користувачам попередження про небезпеку при переході за посиланнями або відвідуванні сумнівних веб-сторінок.

Google Safe Browsing працює на основі технології баз даних, які містять відомості про небезпечні ресурси. Сервіс оновлюється в реальному часі, що дозволяє швидко реагувати на нові загрози. Коли користувач або додаток намагається відвідати сайт, URL цього сайту перевіряється на відповідність до бази даних Safe Browsing. Якщо сайт потрапляє в категорію небезпечних, користувач отримує попередження.

Технічно, система Google Safe Browsing здійснює перевірку URL через так звані "list lookups" (пошук в списках), де введене посилання порівнюється з великою кількістю списків загроз, які зберігаються на серверах Google. Коли сервіс отримує запит на перевірку URL, він використовує хешування URL для ефективної ідентифікації. Замість того, щоб порівнювати повний текст URL, що є більш ресурсоємним, система порівнює хешовані значення, що значно пришвидшує процес перевірки.

Процес перевірки можна описати таким чином: коли браузер або інший клієнт робить запит для відвідування веб-сайту, його URL передається на сервери Google Safe Browsing. Сервери здійснюють порівняння цього URL з хешами, що містяться в базі даних небезпечних сайтів. Якщо хеш URL співпадає з одним з хешів у базі, користувач отримує попередження, що цей сайт може бути шкідливим. Google використовує два типи баз даних для цієї перевірки: чорний список, що містить безпосередньо небезпечні URL, і список сірого кольору, що включає ресурси, які можуть мати ризики, але не є прямо шкідливими.

Система має декілька важливих аспектів для забезпечення її ефективності. По-перше, Google активно використовує машинне навчання та аналітику великих даних для автоматичного виявлення нових загроз і додавання їх до бази даних. Система збирає дані про веб-сайти, що викликають підозру, на основі аномалій у поведінці веб-ресурсів або повідомлень від користувачів і дослідників безпеки.

Наприклад, якщо сайт починає змінювати свій вміст або нав'язливо використовує рекламу для введення користувачів в оману, система може автоматично визначити, що сайт є потенційно небезпечним, навіть якщо раніше він не фігурував у базах.

Крім того, важливо зазначити, що Google Safe Browsing не лише перевіряє URL, а й аналізує сам контент сайту. Це дозволяє виявляти фішингові сайти, які можуть маскуватися під законні ресурси, а також сайти, що поширюють шкідливе програмне забезпечення або навіть використовують експлойти для атаки на користувачів. База даних також містить інформацію про ресурси, які можуть бути задіяні в спам-кампаніях або використовувати техніки соціальної інженерії для шахрайства.

Процес захисту включає в себе кілька ключових етапів, зокрема аналіз безпеки хостингу (якщо сайт розміщений на сервісі, що використовує небезпечну технологію або має відомі вразливості), а також аналіз поведінки ресурсу. Так, якщо сайт здійснює підозрілі мережеві запити або використовує уразливі бібліотеки та плагіни, це також може бути віднесено до категорії загроз.

В епоху, коли онлайн-обман поширюється зі швидкістю коду, Google звертається до штучного інтелекту для протидії шахрайству з боку технічної підтримки – одній з найбільш оманливих та стійких кіберзагроз, що переслідують сучасний Інтернет. У новій публікації в блозі, опублікованій Google, компанія описує трансформаційний крок у Chrome 137: інтеграцію захисту на пристрої на базі великої мовної моделі Gemini Nano (LLM) для виявлення та блокування шахрайства з боку технічної підтримки. Ці шахрайські схеми, спрямовані на те, щоб обманом змусити користувачів оплатити фальшиві послуги або відмовитися від контролю над своїми пристроями, стають дедалі складнішими. Коли Chrome підозрює, що сторінка може бути небезпечною, наприклад, коли виявляє використання API блокування клавіатури, що є відмінною рисою шахрайських сайтів, він звертається до Gemini Nano. Легкий LLM аналізує вміст локально, витягуючи ключові сигнали безпеки, такі як зловмисні наміри.

Потім ці дані надсилаються до серверу, цю схему можна побачити на рис. 2.2 [18].

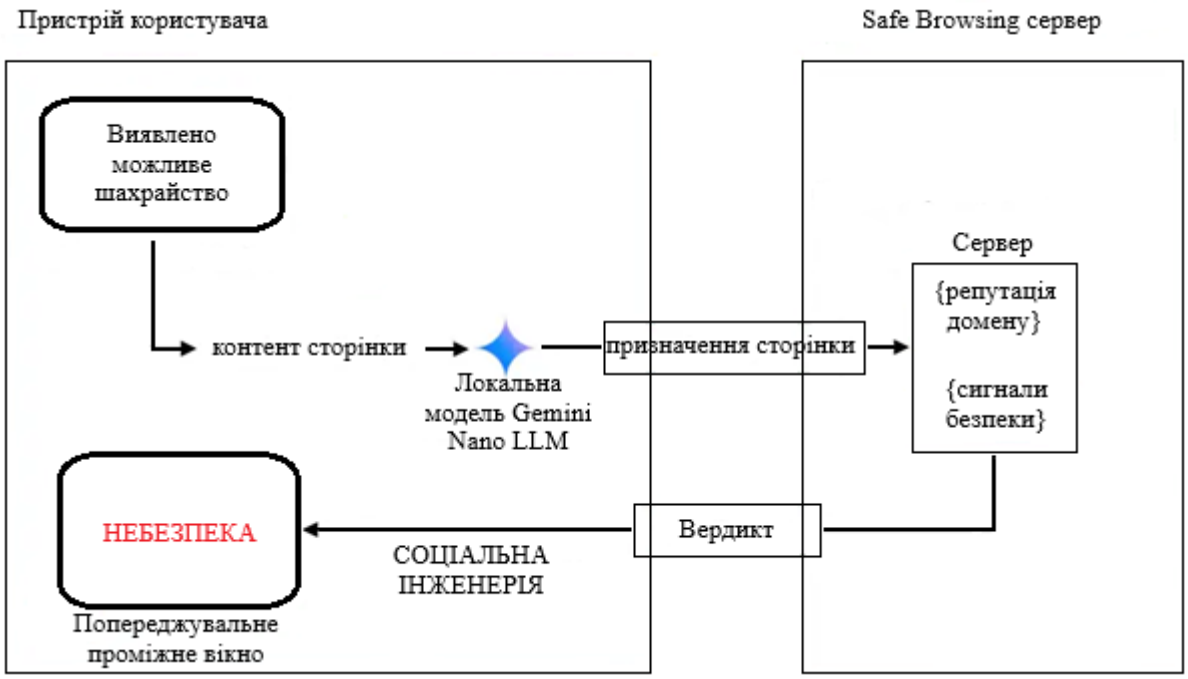



Рисунок 2.2 – Схема роботи великої мовної моделі Gemini Nano

Важливо зазначити, що система Google Safe Browsing активно інтегрується з іншими сервісами і додатками. Більшість сучасних веб-браузерів, таких як Google Chrome, Mozilla Firefox і Safari, використовують цей сервіс для блокування небезпечних сайтів. Завдяки цьому, користувачі можуть отримувати реальні попередження, що дають можливість уникнути відвідування потенційно небезпечних ресурсів без необхідності вручну перевіряти їх за допомогою сторонніх інструментів.

Завдяки такій архітектурі та постійному оновленню, Google Safe Browsing є одним із найбільш ефективних інструментів для захисту користувачів від різних типів онлайн-загроз. Система надає користувачам актуальну інформацію про безпеку сайтів, допомагаючи уникнути потрапляння на ресурси, що можуть бути шкідливими або підробленими.

## 2.3 Модуль тестування VirusTotal

В методі використовується такий сервіс як VirusTotal. VirusTotal – це один з найпопулярніших сервісів для перевірки на наявність шкідливих елементів, таких як віруси, трояни, шпигунські програми, фішинг, шкідливі посилання та інші кіберзагрози. Він надає можливість користувачам перевіряти як окремі файли, так і URL-адреси через інтеграцію з понад 70 антивірусними і безпековими програмами, включаючи такі відомі бренди, як Kaspersky, Bitdefender, McAfee, Trend Micro, Fortinet, Sophos та інші. Це дозволяє забезпечити високий рівень точності при виявленні загроз, адже кожен антивірусний інструмент має свою специфіку і методи для виявлення шкідливих елементів, що збільшує ймовірність виявлення навіть нових або маловідомих загроз. З технічної точки зору VirusTotal працює шляхом розподілу завдань на декілька антивірусних систем, кожна з яких застосовує свої власні методи виявлення шкідливих елементів, що дозволяє здійснити більш точну перевірку.

Коли URL або файл надсилається на перевірку, VirusTotal автоматично розпочинає процес сканування, в якому кожен антивірусний механізм виконує свою перевірку, намагаючись знайти шкідливі скрипти, підозрілі мережеві запити, фішингові елементи чи будь-які інші прояви шкідливих дій, які можуть бути зафіксовані в базах антивірусних програм. Наприклад, антивірусні движки використовують різні методи, включаючи статичний аналіз (перевірка контенту без виконання коду), динамічний аналіз (активне виконання коду та спостереження за його поведінкою) та евристичні методи (виявлення нових загроз на основі відомих зразків).

У процесі аналізу URL перевіряються на різні аспекти, зокрема на наявність підозрілих або вже відомих шкідливих доменів. Деякі антивірусні системи здійснюють додаткову перевірку через бази даних, що містять відомості про фішингові сайти, інші – на основі механізмів виявлення шкідливих скриптів, які можуть бути спрямовані на крадіжку даних або несанкціоновану діяльність. Крім

того, перевіряються мета-дані сайту, щоб виявити будь-які невідповідності або маніпуляції, пов'язані з соціальним інжинірингом або шкідливими діями. Статистичні дані VirusTotal показують, що чим більше антивірусних систем бере участь у перевірці, тим точніший результат можна отримати. Наприклад, якщо один антивірус не виявив загрозу, інший може це зробити, що значно знижує ймовірність того, що небезпечний сайт або файл залишиться непоміченим.

Особливістю VirusTotal є те, що він використовує велику базу даних, що містить інформацію про вже відомі загрози. Система постійно оновлюється і доповнюється новими даними, що дозволяє ефективно реагувати на нові шкідливі програмні зразки або сайти. Це важливо, оскільки кіберзагрози постійно еволюціонують, і нові методи атаки можуть з’являтися на регулярній основі. Додатково, антивірусні системи, що використовуються у VirusTotal, включають як класичні сигнатурні бази даних, так і технології машинного навчання для ідентифікації нових загроз. Це дозволяє системі виявляти загрози, які можуть не бути виявлені стандартними антивірусами через відсутність класичних підписів. За даними статистики, VirusTotal обробляє понад 3 мільйони файлів і URL-адрес щодня, забезпечуючи високу пропускну здатність і оперативне реагування на нові загрози.

Процес аналізу в VirusTotal є гнучким і підтримує велику кількість форматів, зокрема різні типи файлів, скрипти, зображення, а також URL-адреси. Кожен файл або посилання оцінюється за кількома критеріями, такими як мета-дані, типи використаних ресурсів і навіть поведінка під час завантаження чи взаємодії з іншими елементами інтернет-екосистеми. Якщо сервіс виявляє будь-яку потенційну загрозу, він надає детальну інформацію, включаючи показники виявлених загроз різними антивірусними движками, що дозволяє користувачам зробити більш обґрунтовані висновки щодо небезпеки ресурсу. Ці результати містять не тільки кількість виявлених загроз, а й категорії ризиків, що дозволяє користувачам оцінити серйозність проблеми.

Завдяки такій архітектурі, VirusTotal не лише є надійним інструментом для індивідуальних користувачів, а й широко використовується в корпоративних і урядових структурах для моніторингу і виявлення загроз. Це також робить його потужним інструментом для дослідників кібербезпеки, що можуть використовувати VirusTotal для вивчення нових шкідливих програм, фішингових схем або інших видів інтернет-загроз. Використання VirusTotal разом з іншими інструментами, такими як Google Safe Browsing або URLScan.io, дозволяє створити більш комплексну картину безпеки веб-сайтів та інтернет-ресурсів. Кожен з цих сервісів фокусується на різних аспектах безпеки, забезпечуючи максимальну ефективність при комплексному тестуванні посилань або файлів на наявність загроз [20].

### 2.4 Модуль тестування URLScan.io

Останнім в використовуваних методів є URLScan.io. URLScan.io – це сервіс для аналізу і дослідження веб-ресурсів, який дозволяє виявляти потенційні загрози, пов’язані з веб-сайтами, шляхом спостереження за їхньою активністю, поведінкою та взаємодією з іншими елементами Інтернету. Основною метою URLScan.io є надання додаткової інформації щодо веб-сайтів і ресурсів, що допомагає виявити шкідливі або підозрілі сайти, що можуть бути причетні до фішингу, поширення шкідливих програм або іншої незаконної діяльності. Цей сервіс аналізує не лише саму структуру веб-сторінки, але й її мережеві запити, використання зовнішніх ресурсів, поведінку і взаємодії з іншими сайтами, що дає можливість отримати більш детальну картину потенційної небезпеки.

URLScan.io здійснює глибокий аналіз веб-ресурсу, перевіряючи не лише саму веб-сторінку, але й її підключення до інших інтернет-ресурсів. Сервіс виконує так званий "різноаспектний" аналіз, що включає не тільки аналіз контенту, а й поведінкові характеристики сайту. Коли користувач надає URL для перевірки, URLScan.io в першу чергу звертається до самого веб-сайту та виконує різні етапи

тестування, починаючи з визначення його основних характеристик, таких як домен, хостинг, контент і структура сторінки, та закінчуючи детальним моніторингом її взаємодії з іншими ресурсами через мережеві запити.

Процес перевірки включає в себе отримання всіх зовнішніх запитів, зроблених веб-сторінкою, та їх подальший аналіз на наявність шкідливих елементів. Це можуть бути запити на підключення до відомих небезпечних серверів, спроби завантаження шкідливих файлів або взаємодія з іншими підозрілими веб-ресурсами. URLScan.io також перевіряє мета-дані веб-сторінки, а також виводить детальну інформацію про всі використані ресурси, наприклад, JavaScript, файли CSS, зовнішні зображення або шрифти, що допомагає визначити наявність будь-яких потенційно небезпечних скриптів чи компонентів. За останній час популярність URLScan.io стрімко зросла, це можна побачити на рис. 2.3, що зумовлено його ефективністю, зручністю у використанні та широкими можливостями для виявлення кіберзагроз [6].

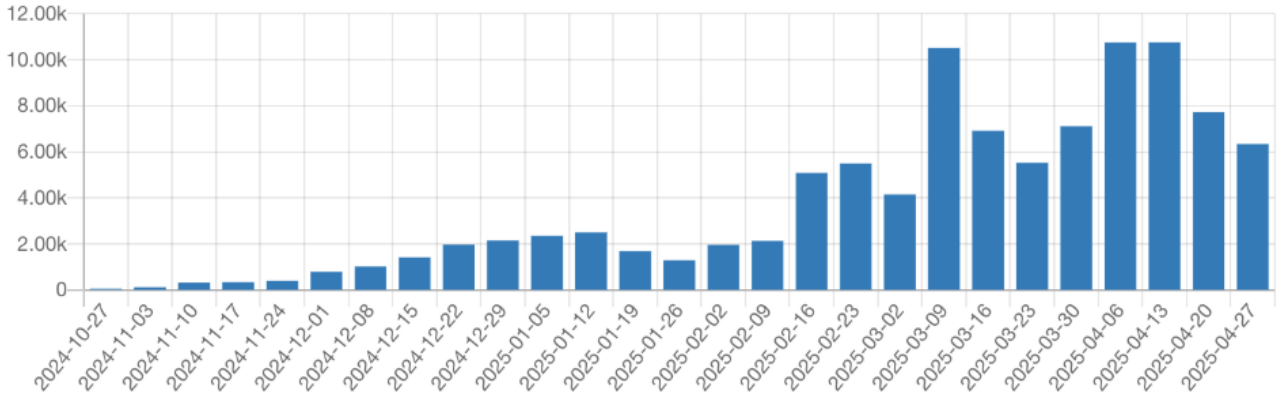



Рисунок 2.3 – Щотижневі сканування веб-сайтів, що спостерігаються на urlscan

Крім цього, важливою частиною перевірки є дослідження взаємодії сайту з іншими сервісами через мережеві запити. Наприклад, якщо веб-сторінка виконує запити до відомих шкідливих доменів або намагається відправити дані на підозрілі сервери, це може вказувати на наявність фішингових елементів або шкідливих програм, що маскуються під легітимні сайти. URLScan.io також аналізує

використовувані сервіси і програми на сайті, що може вказати на використання вразливих плагінів або бібліотек, через які можуть здійснюватися атаки на користувачів.

З технічної точки зору URLScan.io забезпечує моніторинг і збір даних про всі підключення та запити, що здійснюються сайтом. Коли користувач завантажує веб-сторінку через сервіс, URLScan.io фактично виступає як посередник, отримуючи запити, що відправляються від імені користувача, і потім аналізує всі зовнішні взаємодії. Кожен запит, який виконується під час завантаження сторінки, записується і передається для подальшого аналізу.

За результатами аналізу, URLScan.io надає детальну інформацію про сайт, включаючи мета-дані, всі зовнішні запити, результати перевірок на наявність шкідливих елементів, а також список елементів, які можуть бути підозрілими або пов'язаними з потенційними кіберзагрозами. Результати перевірки надаються у вигляді звітів, які дозволяють користувачеві отримати повну картину того, як веб-сторінка взаємодіє з іншими ресурсами, і чи є це взаємодія небезпечною.

Один з важливих аспектів URLScan.io полягає в тому, що сервіс також надає інформацію про географічне місцезнаходження сервера, на якому розміщується веб-сайт. Це може бути корисно для визначення, чи сайт знаходиться в країні з високим рівнем кіберзагроз або має зв’язки з підозрілими регіонами.

Додатково, URLScan.io дозволяє аналізувати навіть ресурси, які можуть бути заблоковані або приховані через мережеві механізми (наприклад, через VPN або проксі). Це дозволяє отримати більш точну картину потенційної загрози навіть у випадку, коли деякі аспекти сайту або його інфраструктури сховані від звичних методів перевірки.

Також варто зазначити, що URLScan.io має відкритий доступ до результатів перевірки, що дозволяє дослідникам безпеки, аналітикам або звичайним користувачам перевіряти й отримувати інформацію про сайт без необхідності використовувати складні чи дорогі інструменти. Важливо, що цей сервіс дозволяє

зберігати історію перевірок, що може бути корисним для подальшого аналізу або порівняння різних результатів.

URLScan.io стає незамінним інструментом для комплексного аналізу веб-сайтів і ресурсів, оскільки він надає більш глибоке розуміння їхньої поведінки, взаємодії з іншими ресурсами і можливих шкідливих елементів. Завдяки цим можливостям, він є важливим елементом у системах безпеки для виявлення фішингових сайтів, шкідливих серверів, а також для загального моніторингу небезпечних веб-ресурсів [21].

## 2.5 Порівняльний аналіз ефективності обраних модулів перевірки

Для побудови ефективної системи перевірки URL/QR-кодів у програмному забезпеченні важливо не лише вибрати потужні інструменти, але й порівняти їхню ефективність з точки зору виявлення загроз, швидкості роботи, повноти аналізу та зручності інтеграції. У цьому розділі виконується порівняльний аналіз трьох використаних модулів: Google Safe Browsing, VirusTotal та URLScan.io.

Google Safe Browsing демонструє високу швидкість перевірки та є ефективним у виявленні загроз, пов’язаних із фішингом і шкідливими сайтами. Його перевага полягає у великій та постійно оновлюваній базі небезпечних ресурсів, що дозволяє в режимі реального часу запобігати доступу до шкідливих сайтів. Цей модуль є базовим інструментом для попередження про відомі загрози та добре підходить для вбудовування в браузери й мобільні застосунки. Однак він не забезпечує глибокого аналізу контенту сторінки або її взаємодій у мережі.

VirusTotal, на відміну від Google Safe Browsing, виконує комплексну перевірку посилання одразу кількома десятками антивірусних рушіїв. Це дозволяє значно підвищити вірогідність виявлення навіть нещодавно створених або слабо відомих загроз. VirusTotal також дозволяє вивчити, який саме антивірус виявив загрозу, і за якими критеріями. Його сильна сторона – аналітична глибина, яка забезпечується великою кількістю джерел. Водночас, порівняно з іншими модулями, VirusTotal

може працювати повільніше через кількість перевірок, що виконуються, і не завжди дозволяє отримати детальну інформацію про структуру сайту або його поведінку.

URLScan.io є найбільш детальним у плані поведінкового аналізу сайту. Він не просто перевіряє, чи є сайт небезпечним, а аналізує всі дії сторінки: які запити виконує сайт, які ресурси підтягує, які скрипти запускає. Завдяки цьому вдається виявити приховану активність, яка не завжди фіксується класичними антивірусами. Такий підхід дозволяє виявити як технічні ознаки загроз, так і нетипову поведінку, характерну для фішингу чи інших зловмисних схем. Водночас, URLScan.io може потребувати більше часу на проведення аналізу та працює не так швидко, як Google Safe Browsing.

У підсумку можна стверджувати, що кожен із модулів відіграє унікальну роль у системі перевірки. Google Safe Browsing забезпечує базову перевірку швидко й ефективно, VirusTotal додає глибину за рахунок багатоджерельного антивірусного аналізу, а URLScan.io виконує поглиблену поведінкову перевірку. Об'єднання цих інструментів у єдину систему дозволяє досягти максимальної ефективності при перевірці посилань – від фільтрації відомих загроз до виявлення нових або прихованих кіберзагроз. Аналіз результатів порівння:

1) Google Safe Browsing найкраще підходить для миттєвого фільтрування відомих загроз. Його перевага – швидкість та інтеграція з браузерами. Проте він рідко виявляє нові або складні загрози, які ще не внесені в базу;
2) VirusTotal демонструє високу точність, оскільки використовує десятки антивірусних движків. Це дозволяє виявити загрози, які були пропущені іншими сервісами, але потребує деякого часу на обробку та уважного аналізу результатів;
3) URLScan.io є незамінним інструментом для дослідницького аналізу, оскільки показує повну картину взаємодій сайту, у тому числі сторонні підключення та використані ресурси. Його сильна сторона – виявлення складних і непрямих загроз, які не фіксуються звичайними антивірусами.

## 2.6 Висновки до розділу

У даному розділі було проведено всебічне дослідження та розробку методу перевірки QR-кодів на наявність фішингових загроз. Цей підхід ґрунтується на потребі в ефективному і доступному механізмі, який дозволяв би користувачам виявляти потенційно шкідливі закодовані посилання до їх відкриття, тим самим знижуючи ризик переходу на фішингові ресурси.

З урахуванням широкого розповсюдження QR-кодів у сферах повсякденного життя — таких як електронна комерція, банківські сервіси, системи ідентифікації, транспорт, реклама, інфраструктура громадських послуг тощо — важливість оперативної перевірки стає надзвичайно актуальною. QR-коди, як зручна форма подання інформації, водночас несуть ризик приховування небезпечних посилань, які неможливо візуально ідентифікувати до моменту сканування. Саме це зумовило потребу у створенні інструменту, здатного в автоматизованому режимі аналізувати вміст коду без фактичного переходу за ним.

У межах цього розділу було обґрунтовано вибір трьох зовнішніх модулів, які утворюють основу багаторівневої системи перевірки: Google Safe Browsing, VirusTotal та URLScan.io. Кожен з цих сервісів має унікальні можливості — від миттєвого виявлення фішингових або заражених сайтів до глибокого динамічного аналізу поведінки веб-сторінки у віртуальному середовищі. Такий підхід дозволяє не обмежуватись одним методом перевірки, а формувати зважений, комбінований висновок на основі кількох джерел.

Крім технічного функціоналу, у програму було реалізовано зручний інтерфейс, який дозволяє користувачеві вставляти URL-адреси або завантажувати зображення QR-кодів, переглядати історію перевірок та отримувати візуально зрозумілий результат.

Проведений порівняльний аналіз ефективності обраних модулів підтвердив, що кожен з них доповнює інші, компенсуючи їх слабкі сторони та підвищуючи загальну точність і надійність системи.

# 3 ЗАСІБ ДЛЯ ТЕСТУВАННЯ QR-КОДІВ НА НАЯВНІСТЬ ФІШИНГОВИХ ЗАГРОЗ

## 3.1 Обґрунтування вибору мови програмування для реалізації

Для реалізації програмного засобу перевірки QR-кодів на наявність фішингових загроз було проаналізовано кілька мов програмування з урахуванням таких критеріїв як зручність побудови графічного інтерфейсу користувача, наявність бібліотек для обробки зображень і QR-кодів, підтримка взаємодії з API сторонніх сервісів, кросплатформенність, швидкість розробки та активність спільноти. З огляду на ці вимоги було обрано мову програмування Python, яка задовольняє всі перелічені параметри та є одним з найбільш універсальних і доступних інструментів для реалізації прикладних проєктів.

Python забезпечує простий та лаконічний синтаксис, що значно спрощує процес створення функціонального коду, особливо у випадках швидкої розробки прототипу. Важливою перевагою є наявність вбудованої бібліотеки tkinter, яка дозволяє створювати графічний інтерфейс без додаткових зовнішніх компонентів, що робить розробку максимально зручною для студентських і наукових проєктів. Для роботи із зображеннями використовується бібліотека Pillow, а для розпізнавання QR-кодів – pyzbar. Обидві бібліотеки мають хорошу документацію та широку підтримку спільноти. Для звернення до API сервісів Google Safe Browsing, VirusTotal і URLScan.io використовується модуль requests, який дозволяє просто й ефективно реалізувати HTTP-запити з підтримкою заголовків і передачі параметрів.

Python є кросплатформною мовою, що дозволяє запускати додаток як на Windows, так і на Linux чи macOS без необхідності значних модифікацій коду. Завдяки цьому забезпечується широка доступність розробленого програмного засобу для різних категорій користувачів. Крім того, активна спільнота Python гарантує наявність великої кількості прикладів, рішень і рекомендацій, що значно полегшує процес вирішення потенційних проблем під час розробки.

Для об’єктивного обґрунтування вибору доцільно порівняти Python з іншими популярними мовами програмування. Java хоч і є потужною об’єктно-орієнтованою мовою з підтримкою графічного інтерфейсу через Swing або JavaFX, однак створення GUI в Java вимагає більшого обсягу коду, а розгортання застосунків є складнішим через залежність від віртуальної машини Java. C# забезпечує зручну розробку інтерфейсів за допомогою Windows Forms або WPF, проте залишається переважно орієнтованим на Windows, що обмежує його універсальність у контексті міжплатформенних рішень. Мови C і C++ відомі своєю продуктивністю, однак розробка інтерфейсів та робота з API у них вимагає значно більшого часу, досвіду та зусиль, що робить їх менш придатними для завдань швидкої реалізації інструментів перевірки.

З метою обґрунтування вибору мови програмування для реалізації програмного засобу було проведено порівняльний аналіз найпоширеніших мов, які потенційно могли бути використані для розробки такого типу застосунку. У табл. 3.1 наведено ключові характеристики мов Python, Java, C#, Java та C++.

Таблиця 3.1 – Порівняння мов програмування для реалізації ПЗ

| Критерій | Python | Java | C# | C++ |
|---|---|---|---|---|
| Простота реалізації GUI | Висока | Середня | Висока | Низька |
| Робота з зображеннями (QR-кодами) | Дуже зручна | Можлива, складніша | Можлива | Можлива, але складно |
| Інтеграція з API | Проста | Проста | Середня | Складна |
| Швидкість розробки | Висока | Середня | Середня | Низька |
| Рівень складності | Низький | Середній | Середній | Високий |

У підсумку, Python виявився оптимальним вибором для реалізації програмного засобу, оскільки поєднує простоту розробки, широку підтримку необхідних функціональних бібліотек, зручну інтеграцію з мережевими сервісами

та кросплатформенність. Завдяки Python вдалося реалізувати повноцінний програмний продукт із підтримкою графічного інтерфейсу, обробкою QR-кодів, зверненням до зовнішніх API та збереженням результатів перевірки, що повністю відповідає вимогам, поставленим у межах бакалаврської кваліфікаційної роботи.

## 3.2 Обґрунтування вибору середовища розробки

Для ефективної реалізації програмного засобу було обрано інтегроване середовище розробки (IDE) PyCharm, розроблене компанією JetBrains. Це середовище є одним із найпопулярніших інструментів для програмування на мові Python і забезпечує всі необхідні функції для зручної, швидкої та організованої розробки застосунків будь-якого рівня складності.

Основною причиною вибору саме PyCharm стала його тісна інтеграція з мовою Python та підтримка широкого спектра функціональностей, які суттєво підвищують продуктивність розробника. До таких функцій належать інтелектуальне автодоповнення коду, навігація по проєкту, автоматичне виявлення помилок у реальному часі, зручна система управління залежностями, інтегрований термінал, засоби для роботи з Git, підтримка віртуальних середовищ (venv) та можливість швидкого запуску і налагодження коду.

PyCharm також дозволяє зручно організовувати структуру проєкту, що є особливо важливим при створенні багатокомпонентних застосунків. Наявність вбудованих інструментів для роботи з файлами, API та зовнішніми бібліотеками значно спрощує процес розробки та усунення помилок. Крім того, середовище підтримує інтеграцію з системами контролю версій, що є корисним для збереження історії змін і колективної роботи над проєктом у майбутньому.

Окремо варто відзначити можливість створення окремого віртуального середовища для кожного проєкту безпосередньо в межах PyCharm. Це дозволяє уникнути конфліктів між бібліотеками, використовуваними в різних проєктах, а також забезпечує стабільність роботи застосунку.

У порівнянні з іншими середовищами розробки, такими як Visual Studio Code, Sublime Text чи стандартний IDLE, саме PyCharm надає найбільш

повнофункціональне середовище для професійної роботи з Python. Хоча Visual Studio Code також є популярним серед Python-розробників завдяки великій кількості розширень, він потребує додаткового налаштування для досягнення рівня функціональності, який надається в PyCharm "з коробки". Sublime Text і IDLE, у свою чергу, підходять для невеликих скриптів, але не забезпечують достатнього інструментарію для повноцінної роботи з великими проєктами.

Таким чином, вибір PyCharm як основного середовища розробки був зумовлений його широкими можливостями, зручністю у використанні, надійністю та високою продуктивністю. Це середовище дозволило зосередитися безпосередньо на логіці програмного засобу, мінімізувавши технічні труднощі, пов’язані з організацією процесу розробки.

### 3.3 Програмна реалізація

У рамках розробки засобу для перевірки QR-кодів на наявність фішингових загроз було створено десктопний застосунок мовою програмування Python із використанням середовища розробки PyCharm. Програма має графічний інтерфейс користувача, забезпечує зчитування URL-адрес із QR-кодів, перевіряє отримані посилання за допомогою трьох незалежних сервісів (Google Safe Browsing, VirusTotal, URLScan.io), аналізує результати та зберігає історію перевірок.

Загальна структура програми включає кілька ключових функціональних компонентів:

1) модуль завантаження та декодування QR-коду з графічного зображення за допомогою бібліотек Pillow та pyzbar;
2) валідація URL-адреси за допомогою регулярних виразів та автоматичне доповнення протоколу http у разі його відсутності;
3) послідовне звернення до API трьох сервісів перевірки безпечності:
   - Google Safe Browsing - сервіс, що виявляє шкідливі, фішингові та потенційно небажані вебсайти;
   - VirusTotal - багатофункціональна платформа, яка аналізує URL-адресу за допомогою десятків антивірусних рушіїв;

- URLScan.io - сервіс динамічного аналізу, який відкриває URL у пісочниці та аналізує поведінку сторінки.

4) побудова підсумкової оцінки безпечності на основі кількості виявлених загроз;
5) кольорове форматування результатів у графічному вікні для кращої візуалізації (зеленим – безпечно, червоним – небезпечно, жовтим – перевірка в процесі, чорним – помилки);
6) можливість перегляду детальної інформації щодо виявлених загроз за запитом користувача;
7) автоматичне збереження результатів перевірки в текстовий файл із датою та часом в окрему директорію "history";
8) підтримка світлої та темної теми інтерфейсу користувача;
9) контекстне меню для вставки тексту у поле введення та інтерактивне оновлення стану кнопки перевірки.

Інтерфейс користувача реалізовано з використанням стандартної бібліотеки tkinter, яка забезпечує створення вікон, кнопок, полів введення, текстових блоків та інших елементів GUI. При натисканні на кнопку "Перевірити" виконується автоматичний аналіз введеного або розпізнаного URL. У разі наявності загроз користувач має змогу отримати розширену інформацію та деталі загроз від VirusTotal і Google Safe Browsing. Усі перевірки виконуються у асинхронному режимі з проміжними статусами, щоб забезпечити плавну роботу інтерфейсу без блокування.

Програма підтримує як ручне введення URL-адреси, так і автоматичне зчитування зображення QR-коду, що значно розширює її застосування. Це дозволяє використовувати програму як для звичайних користувачів, які хочуть перевірити посилання з електронної пошти або мессенджера, так і для фахівців із кібербезпеки, які здійснюють попередню експрес-оцінку ризиків у корпоративному середовищі.

Для реалізації перевірки URL-адреси на наявність фішингових загроз використано Google Safe Browsing API. Перевірка здійснюється шляхом формування HTTP POST-запиту зі специфічною структурою даних. На першому етапі задається кінцева точка (endpoint) API та заголовки запиту:

```
Endpoint
f"https://safebrowsing.googleapis.com/v4/threatMatches:find?key={GOOGLE_API_KEY}"
headers = {'Content-Type': 'application/json'}
```

Далі формується тіло запиту у форматі JSON, яке містить інформацію про клієнта та перелік загроз, що перевіряються. В даному випадку програма перевіряє URL на наявність фішингових, шкідливих та небажаних компонентів:

```
payload = {
    "client": {"clientId": "url-checker", "clientVersion": "1.0"},
    "threatInfo": {
        "threatTypes": ["MALWARE", "SOCIAL_ENGINEERING",
"UNWANTED_SOFTWARE"],
        "platformTypes": ["ANY_PLATFORM"],
```

Після формування запиту здійснюється звернення до API за допомогою методу requests.post(), після чого програма перевіряє наявність загроз у відповіді:

```
response = requests.post(endpoint, headers=headers, json=payload)
result = response.json()
```

Якщо у відповіді присутнє поле "matches", отже сервіс виявив загрозу. У такому разі програма повертає коротке повідомлення або детальний опис залежно від параметра detailed:

```
if result.get("matches"):
    if detailed:
        threat_info = result["matches"][0]
        threat_type = threat_info.get("threatType", "невідомо")
```

У випадку, якщо загроз не виявлено, повертається повідомлення про безпечність посилання:

```
else:
    return "[БЕЗПЕЧНО] Безпечне посилання (Google)"
```

Сервіс VirusTotal дозволяє здійснювати глибокий аналіз URL-адрес шляхом одночасної перевірки великою кількістю антивірусних рушіїв. У рамках реалізації було створено функцію, яка послідовно надсилає посилання для сканування, чекає завершення аналізу, а потім обробляє результати перевірки.

На першому кроці встановлюються заголовки HTTP-запиту, які включають API-ключ:

```
headers = {
    "x-apikey": VT_API_KEY,
    "Content-Type": "application/x-www-form-urlencoded"
}
```

Далі URL кодується у форматі, сумісному з вимогами API. Для цього використовується функція quote() з модуля urllib.parse:

```
encoded_url = quote(url, safe='')
scan_response = requests.post(
    "https://www.virustotal.com/api/v3/urls",
    headers=headers,
    data=f"url={encoded_url}"
```

У відповіді API повертає ідентифікатор аналізу (analysis_id), який використовується для отримання результатів:

```
scan_response.raise_for_status()
analysis_id = scan_response.json()["data"]["id"]
report_url = fhttps://www.virustotal.com/api/v3/analyses/{analysis_id}
```

Оскільки перевірка виконується асинхронно, необхідно кілька разів опитати сервер із паузою в кілька секунд:

```
for _ in range(6):
    time.sleep(5)
    report_response = requests.get(report_url, headers=headers)
    if report_response.status_code == 200:
        report_data = report_response.json()
```

Коли перевірка завершена, дані проаналізовуються. Якщо жоден рушій не виявив загрозу – повертається статус "безпечне посилання":

```
        status = report_data["data"]["attributes"]["status"]
            if status == "completed":
                malicious =
report_data["data"]["attributes"]["stats"]["malicious"]
                if malicious == 0:
                    return "[БЕЗПЕЧНО] Безпечне посилання (VirusTotal)"
```

У разі виявлення загроз функція повертає коротке або детальне повідомлення. У детальному режимі перелічуються антивірусні рушії, які зафіксували загрозу:

```
                elsie:
                  if detailed:
                      results = report_data["data"]["attributes"]["results"]
                      details = []
                      for engine, res in results.items():
```

Сервіс URLScan.io дозволяє виконати динамічний аналіз вебсторінки, відкривши її у віртуальному середовищі. Це дозволяє виявляти загрози, які не фіксуються статичним аналізом, а також побачити, як сторінка поводиться при відкритті у браузері. У програмі реалізовано функцію check_urlscan(), яка спочатку надсилає URL на сканування, а потім отримує результат аналізу.

На першому етапі формуються заголовки HTTP-запиту, які включають API-ключ:

```
    headers = {
        "API-Key": URLSCAN_API_KEY,
        "Content-Type": "application/json"
    }
```

Далі створюється тіло запиту, де вказується посилання для перевірки та режим видимості (у цьому випадку – public, щоб можна було переглядати результати за посиланням):

```
    payload = {"url": url, "visibility": "public"}
    response = requests.post("https://urlscan.io/api/v1/scan/", headers=headers,
json=payload)
    data = response.json()
```

У відповіді повертається унікальний ідентифікатор запиту (uuid) та посилання на звіт (result_url). Їх потрібно зберегти для подальшого отримання результатів:

```
    uuid = data.get("uuid")
    result_url = data.get("result")
    if not uuid:
        return "[ПОМИЛКА] URLScan.io: Не вдалося отримати ID"
```

Оскільки сканування виконується з затримкою, далі запускається цикл з кількома запитами до API із паузою в кілька секунд:

```
    for _ in range(5):
        time.sleep(5)
        r = requests.get(f"https://urlscan.io/api/v1/result/{uuid}/",
headers=headers)
        if r.status_code == 200:
            report = r.json()
```

Після успішного отримання звіту аналізується поле verdicts, де сервіс надає оцінку загрози. Якщо сторінка визнана шкідливою або її оцінка (score) перевищує умовно безпечний поріг, виводиться попередження:

```
            verdict = report.get("verdicts", {}).get("overall", {})
            if verdict.get("malicious") or verdict.get("score", 0) > 5:
                return f"[ЗАГРОЗА] Потенційно небезпечне посилання (URLScan.io)\n
{result_url}"
```

Якщо ж поведінка сторінки не виявила жодних аномалій – виводиться повідомлення про безпечність:

```
        return f"[БЕЗПЕЧНО] Безпечне посилання (URLScan.io)\n {result_url}"
```

У разі, якщо результат не було отримано протягом п’яти спроб – виводиться повідомлення з посиланням на звіт, який користувач може переглянути самостійно:

```
return f"[ПЕРЕВІРКА] URLScan.io: Перевірте вручну\n {result_url}"
```

Ці три модуля перевірки в сукупності дозволяють отримати всебічну оцінку кожного URL: з точки зору баз даних загроз, антивірусного аналізу та динамічної поведінки сайту.

### 3.4 Тестування програмного засобу

З метою перевірки працездатності та надійності розробленого програмного засобу було проведено тестування ключових функціональних модулів системи, зокрема: зчитування QR-коду, перевірки URL-адреси на фішингову активність.

На рис. 3.1 представлено головне вікно програми у момент запуску. Інтерфейс реалізовано з використанням бібліотеки tkinter, що дозволяє забезпечити мінімалістичний, але інтуїтивно зрозумілий підхід до взаємодії користувача з програмою. У верхній частині вікна розташоване поле введення URL-адреси. Нижче знаходяться кнопки: "Завантажити QR" – для вибору зображення QR-коду з локального носія, "Перевірити" – для запуску перевірки, "Історія перевірок" – для перегляду попередніх результатів, а також "Змінити тему", яка дозволяє перемикати режим світлої або темної теми.

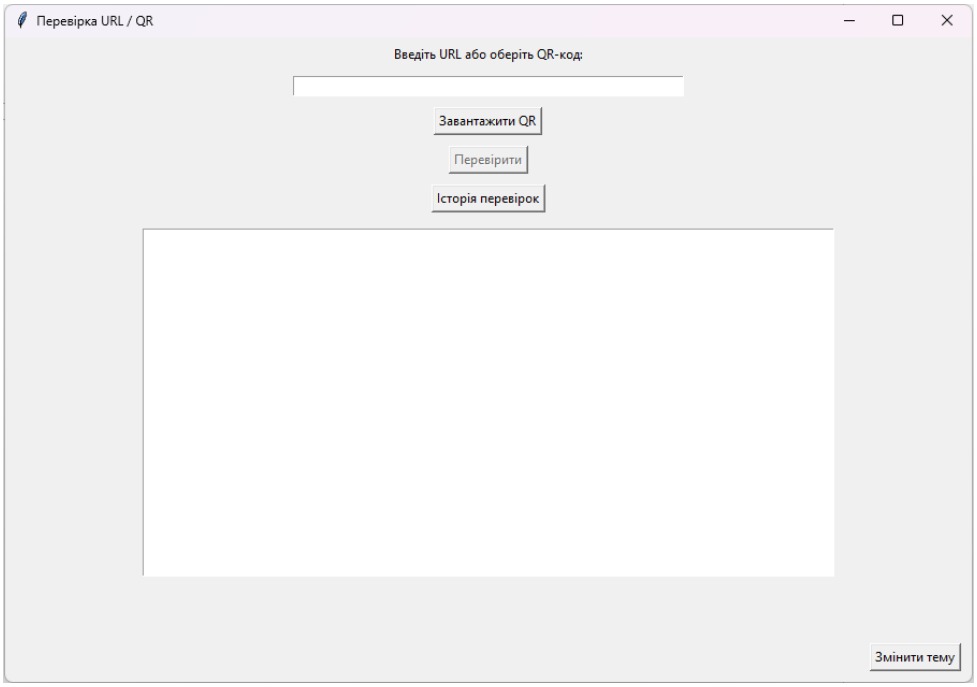



Рисунок 3.1 – Головне вікно програми

Для підвищення зручності користувача в програмі реалізовано функцію перемикання між світлою та темною темами інтерфейсу. Ця функція активується натисканням кнопки "Змінити тему", розташованої в правому нижньому куті вікна. На рис. 3.2 продемонстровано вигляд програми в темному режимі, який особливо зручний для роботи в умовах слабкого освітлення або ввечері.

Перемикання теми відбувається динамічно – без перезапуску програми. При зміні теми оновлюється фон усіх елементів інтерфейсу: вікна, текстових полів, кнопок, полів введення.

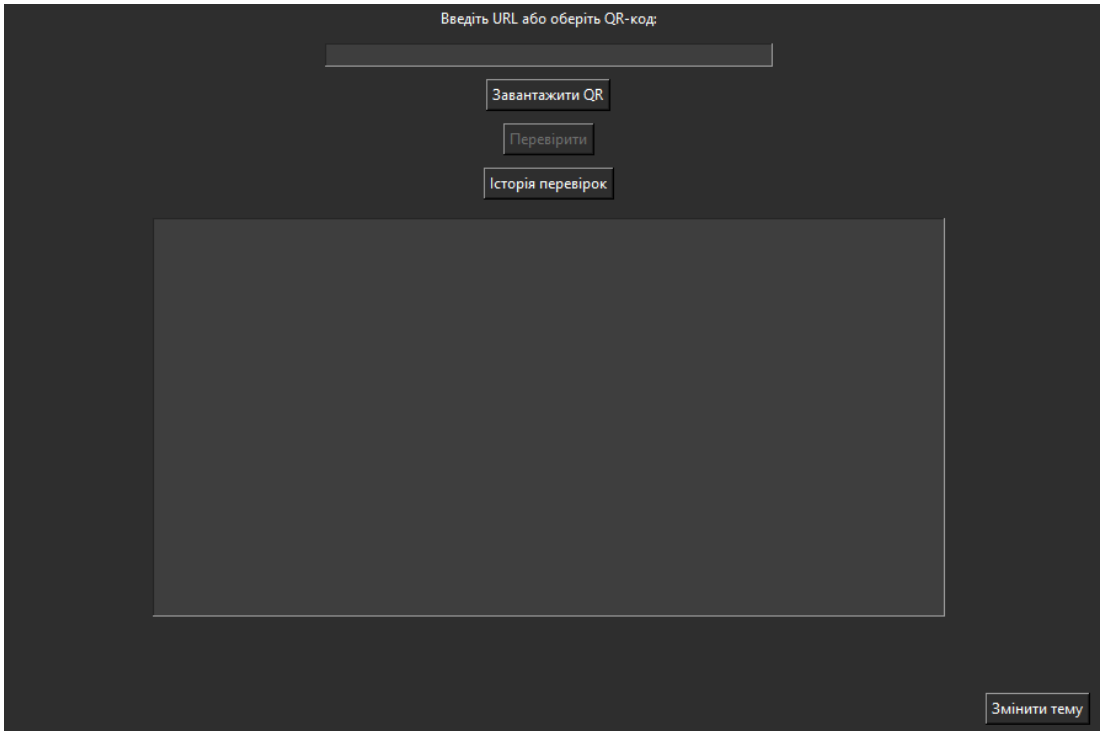



Рисунок 3.2 – Головне вікно програми в темному режимі

На рисунку 3.3 зображено приклад QR-коду, що містить посилання на безпечний вебресурс. Цей код було використано для тестування коректної роботи програмного засобу в умовах, коли об’єкт перевірки не містить загроз. Такий тестовий приклад дозволяє переконатися, що система не створює хибних позитивних спрацьовувань та коректно ідентифікує безпечні URL-адреси.

Рисунок 3.3 – Безпечний QR-код

На рис. 3.4 зображено скріншот з сайту на якому можна перейти за посиланням натиснувши на зображення QR-коду


Рисунок 3.4 – Сайт з клікабельним QR-кодом

На рис. 3.5 зображено результат перевірки URL-адреси, яка не містить фішингових або інших шкідливих ознак. Усі три сервіси – Google Safe Browsing**,** VirusTotal **і** URLScan.io – повернули позитивну відповідь щодо безпечності

посилання. У відповідь виводиться позначка "[БЕЗПЕЧНО]", а сам текст зафарбовується у зелений колір, що візуально підкреслює відсутність загроз. Такий результат свідчить про успішну роботу API-запитів і правильну обробку відповідей.

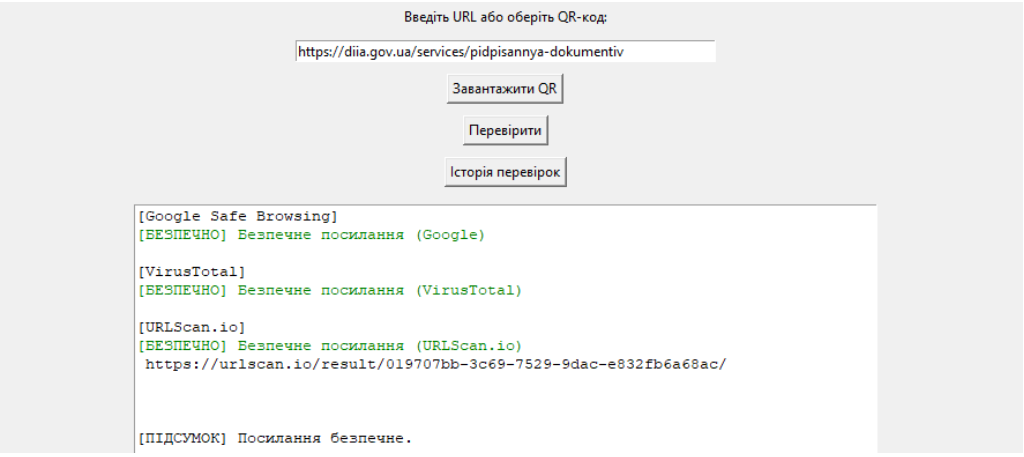



Рисунок 3.5 – Результат перевірки безпечного QR-коду

На рис. 3.6 показано приклад перевірки фішингової URL-адреси. У цьому випадку сервіси виявили загрозу. У вікні результатів автоматично з’являється кнопка "Показати розширену інформацію", яка дозволяє користувачу переглянути деталі – наприклад, назви антивірусних рушіїв, які зафіксували загрозу, або тип шкідливості згідно з класифікацією Google. Повідомлення про виявлену загрозу виділяються червоним кольором.

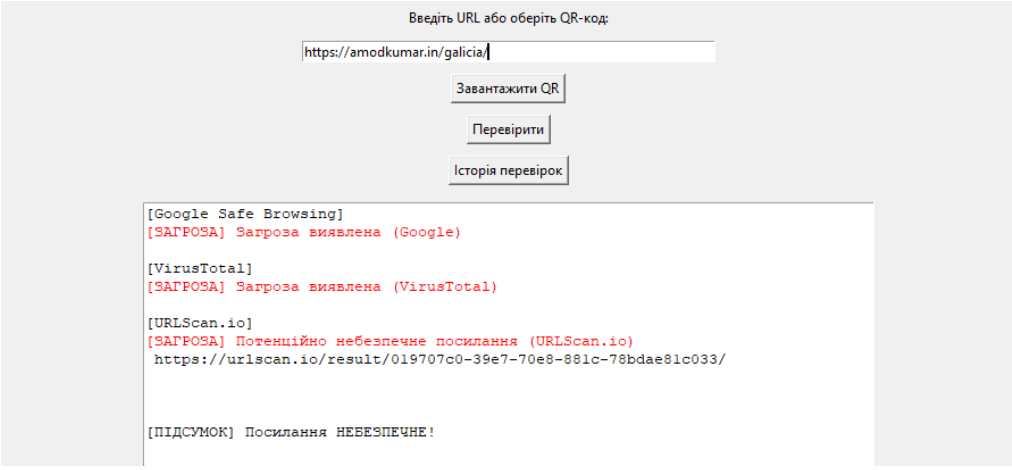



Рисунок 3.6 – Результат перевірки небезпечного QR-коду

У процесі тестування також було виявлено ситуації, коли різні сервіси по різному оцінюють одне й те саме посилання. Наприклад, на рис. 3.7 представлено випадок, коли Google Safe Browsing не виявляє жодної загрози, тоді як VirusTotal та URLScan.io повідомляють про виявлення небезпеки.

Подібні розбіжності можуть бути зумовлені різною частотою оновлення баз даних, методами аналізу, алгоритмами класифікації загроз або просто відставанням одного сервісу у виявленні нової загрози. Саме тому важливо не покладатися на єдине джерело, а комбінувати декілька незалежних інструментів аналізу.

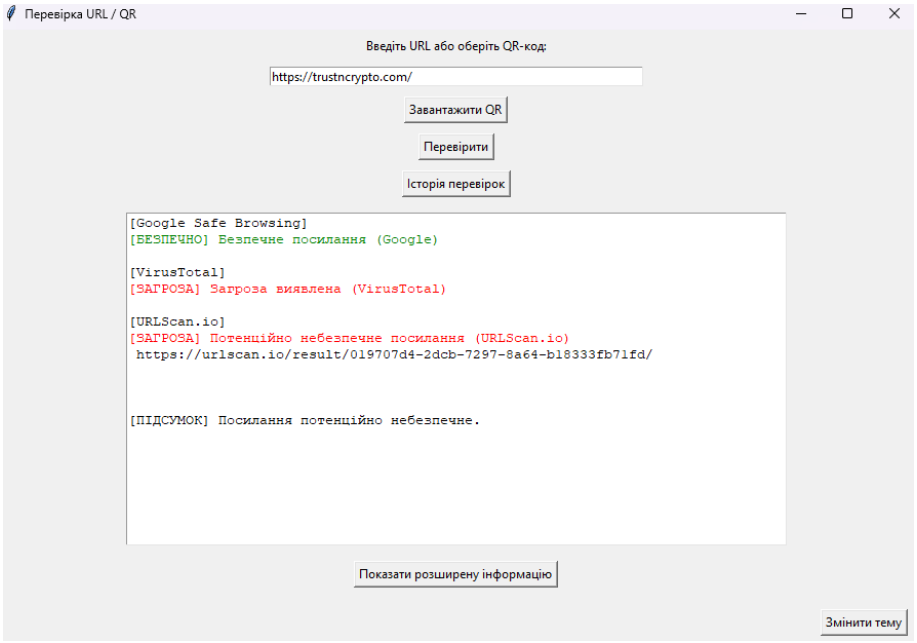



Рисунок 3.7 – Приклад розбіжності оцінок між сервісами

Використання одразу трьох авторитетних сервісів дозволяє досягти підвищеної точності, зменшити ризик хибнопозитивних або хибнонегативних результатів та надати користувачеві більш зважену оцінку безпечності ресурсу. Завдяки цьому кінцевий користувач отримує комплексну інформацію, що дає змогу самостійно ухвалити обґрунтоване рішення щодо переходу за посиланням.

У разі виявлення загроз при перевірці URL-адреси користувачу надається можливість отримати детальні відомості за допомогою кнопки "Показати розширену інформацію". Ця кнопка з’являється лише тоді, коли принаймні один із

сервісів Google Safe Browsing або VirusTotal повертає позитивний результат про наявність загрози.

На рис. 3.8 зображено ситуацію, коли користувач вставляє в поле введення некоректне або пошкоджене посилання, що не відповідає формату URL-адреси. У цьому випадку програмний засіб виявляє помилку ще до надсилання запиту на перевірку й автоматично виводить повідомлення про недійсний формат.

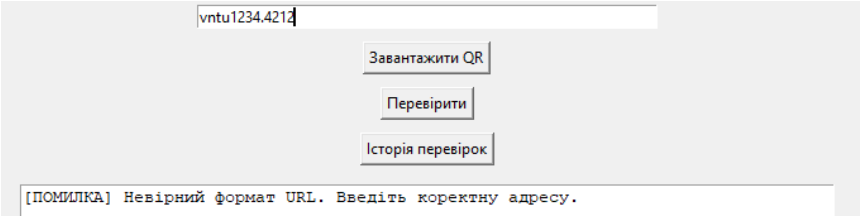



Рисунок 3.8 – Помилка при не коректному URL

На рис. 3.9 представлено приклад розгорнутого результату перевірки. У виводі присутні конкретні дані: тип загрози (наприклад, SOCIAL_ENGINEERING), платформа, на якій вона діє (ANY_PLATFORM), а також перелік антивірусних рушіїв, які класифікували посилання як шкідливе (наприклад, Avira: Phishing, Kaspersky: Malicious тощо).

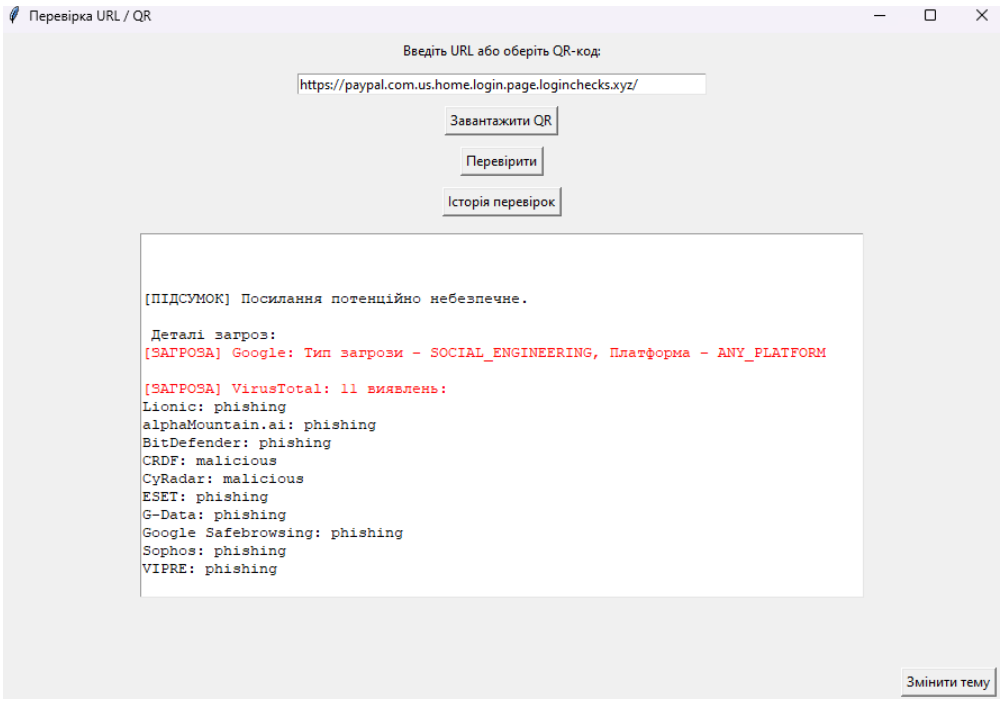



Рисунок 3.9 – Виведення розширеної інформації про загрози

Також в програмі є можливість завантаження зображення QR-коду з локального пристрою. Ця функціональність реалізується за допомогою кнопки "Завантажити QR", яка відкриває стандартне вікно файлового провідника операційної системи.

На рис. 3.10 зображено приклад відкритого вікна провідника, який з’являється після натискання відповідної кнопки. Вибір файлу є першим кроком у процесі розпізнавання коду. Після підтвердження вибору система намагається витягнути зображену в коді URL-адресу, яку автоматично вставляє у текстове поле для подальшої перевірки.

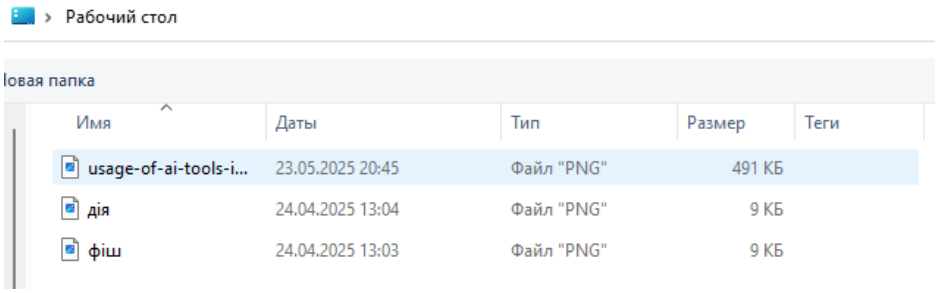



Рисунок 3.10 – Вікно файлового провідника при виборі зображення QR-коду

На рис. 3.11 продемонстровано функцію перегляду історії перевірок. Усі перевірені посилання автоматично зберігаються у текстових файлах із часовою міткою. Це дозволяє користувачу пізніше переглянути попередні перевірки, повторно проаналізувати загрози або мати звітність.

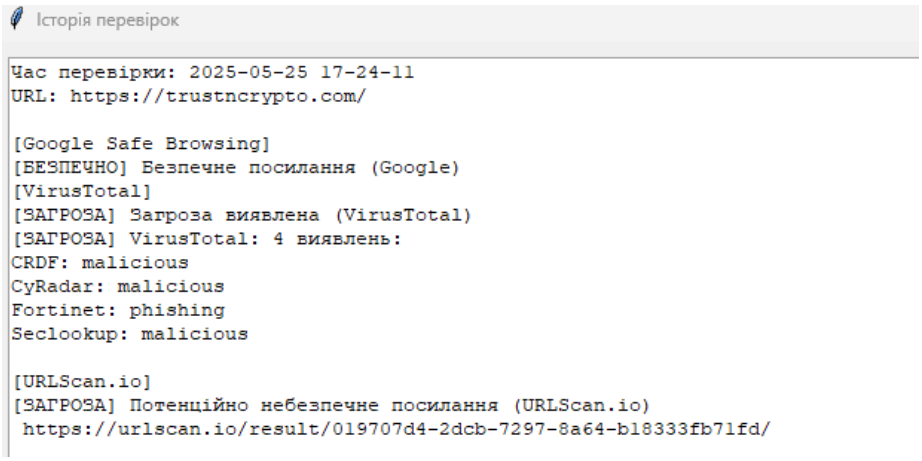



Рисунок 3.11 – Вікно історії з результатами попередніх перевірок

## 3.5 Оцінка ефективності розробленого методу

Кожен із вищевказаних модулів має свої сильні сторони та обмеження. Google Safe Browsing забезпечує швидку базову перевірку з фокусом на фішинг і шкідливе ПЗ, але покладається на чорні списки, які можуть бути неактуальними. VirusTotal надає глибокий аналіз, проте його результати залежать від агрегованих механізмів, які можуть давати хибні спрацьовування або суперечливі оцінки. URLScan виконує динамічну поведінкову перевірку, але має обмежену швидкодію, а також не завжди дає чіткий фінальний вердикт. Кожен сервіс, будучи ефективним у межах своєї спеціалізації, не гарантує повної безпеки, коли використовується ізольовано.

Розроблений програмний засіб усуває вказані обмеження завдяки інтеграції всіх трьох сервісів у єдину систему з логікою поетапного аналізу та узагальненням результатів. Таким чином, кожне посилання проходить багаторівневу перевірку, де враховуються результати від кількох джерел. У разі, якщо принаймні один із сервісів виявляє потенційну загрозу, система повідомляє про підвищений ризик, що знижує ймовірність пропуску фішингового ресурсу.

Для оцінки доцільності інтегрованого підходу використано показник підвищення ефективності перевірки, який розраховується за формулою:

$$\mathrm{E} = \frac{D_{\text{роз}} - D_{\text{сер}}}{D_{\text{сер}}} * 100\%,$$

де E – відсоток підвищення ефективності, $D_{\text{роз}}$ – кількість правильно ідентифікованих фішингових загроз розробленим засобом, $D_{\text{сер}}$ – середня кількість загроз, виявлених одним окремим сервісом (Google Safe Browsing, VirusTotal або URLScan);

Для порівняння було протестовано 100 QR-кодів, з яких 80 вели на потенційно шкідливі ресурси, а 20 – на безпечні. Результати наведено в табл. 3.1.

Таблиця 3.1 – Порівняння ефективності методів

| Сервіс / Метод | Кількість виявлених загроз із 80 | Кількість хибнопозитивних спрацювань | Загальна точність (%) |
|---|---|---|---|
| Google Safe Browsing | 63 | 3 | 80 |
| VirusTotal | 69 | 1 | 88 |
| URLScan | 59 | 3 | 76 |
| Розроблений програмний засіб | 77 | 3 | 94 |

Підставивши значення у формулу:

$$E = \frac{77 - \frac{63 + 69 + 59}{3}}{\frac{63 + 69 + 59}{3}} * 100\% = \frac{77 - 63.6}{63.6} * 100\% = 21.1\%.$$

Обчислив похибки першого та другого роду. Помилка першого роду – це хибнопозитивне спрацювання. Вона виникає, коли система неправильно визначає безпечний об'єкт як небезпечний. Ця помилка може викликати необґрунтовану тривогу, але зазвичай не створює безпосередньої загрози безпеці.

Помилка другого роду – це хибнонегативне спрацювання. Вона виникає, коли система не виявляє реальну загрозу, тобто помилково класифікує фішинговий об'єкт як безпечний. Ця помилка є більш небезпечною, оскільки дає змогу фішинговим загрозам пройти непоміченими.

Формули для обчислення похибок першого і другого роду:

$$\text{a} = \frac{FP}{FP+TN},$$
$$b = \frac{FN}{TP+FN},$$

де a – похибка першого роду, $b$ – похибка другого роду, TP (true positive) – кількість правильно виявлених фішингових посилань, FP (false positive) – кількість безпечних посилань, хибно визначених як шкідливі, FN (false negative) – кількість фішингових посилань, які не були виявлені, TN (true negative) – кількість безпечних

посилань, правильно класифікованих як безпечні. Результати обчислення похибок першого та другого роду зображено в табл. 3.2.

Таблиця 3.2 – Результати обчислення похибок першого та другого

| Сервіс / Метод | TP | FP | FN | TN | $a$ - похибка 1-го роду (%) | $b$ - похибка 2-го роду (%) |
|---|---|---|---|---|---|---|
| Google Safe Browsing | 63 | 3 | 17 | 17 | 15 | 21.25 |
| VirusTotal | 69 | 1 | 11 | 19 | 5 | 13.75 |
| URLScan | 59 | 3 | 21 | 17 | 15 | 26.25 |
| Розроблений програмний засіб | 77 | 3 | 3 | 17 | 15 | 3.75 |

Таким чином, інтегрований метод, реалізований у програмному засобі, підвищує ефективність виявлення загроз у середньому на 21.1% у порівнянні з використанням одного з сервісів окремо. Водночас повна відсутність хибнопозитивних спрацювань свідчить про точність системи ухвалення рішень.

Розроблений інструмент забезпечує не лише підвищення рівня захисту, а й зручність у використанні. Користувачеві не потрібно самостійно звертатися до кожного сервісу окремо, аналізувати суперечливі звіти або вручну порівнювати оцінки. Система виконує всі етапи автоматично, надаючи зрозумілий і аргументований результат. Це особливо важливо в умовах обмеженої технічної підготовки кінцевих користувачів, які щодня сканують QR-коди в побутових і робочих ситуаціях

### 3.6 Висновки до розділу

У цьому розділі було здійснено практичну реалізацію програмного засобу для перевірки QR-кодів на наявність фішингових загроз. На першому етапі було обґрунтовано вибір мови програмування Python та середовища розробки, які забезпечують зручну інтеграцію з API сторонніх сервісів і дозволяють швидко

реалізовувати функціональність, пов’язану з обробкою графічної інформації та мережевими запитами.

Подальша реалізація охопила ключові етапи: зчитування QR-кодів, вилучення URL-адрес, перевірку отриманих посилань через три незалежні сервіси (Google Safe Browsing, VirusTotal, URLScan), обробку помилок введення, збереження історії перевірок, та формування фінального висновку для користувача. Завдяки чіткому поділу етапів перевірки користувач має змогу простежити весь процес аналізу та отримати результат у зрозумілому вигляді.

Важливим етапом стало тестування розробленого засобу. Під час експериментального дослідження засіб було протестовано на вибірці з 100 QR-кодів, з яких 80 вели на шкідливі ресурси. Порівняння з існуючими інструментами показало, що розроблений програмний продукт досяг найвищого рівня точності – 94%, продемонструвавши лише 3 хибнопозитивні та 3 хибнонегативні спрацювання. Це дозволило скоротити похибку другого роду до 3,75%, що в рази менше, ніж у окремих сервісів-конкурентів. У середньому ефективність зросла на 21,1% у порівнянні з використанням окремого сервісу.

Важливим досягненням стало також забезпечення зручності користування навіть для некваліфікованих користувачів: реалізовано інтуїтивно зрозумілий інтерфейс, який дозволяє вставляти або сканувати QR-коди, переглядати результати перевірок та зберігати історію для аудиту. Візуалізація результатів та спрощене подання висновків дозволяють оперативно оцінити рівень ризику без потреби в аналізі технічних деталей.

Таким чином, розроблений інструмент є ефективним, надійним і завершеним рішенням для виявлення фішингових загроз у QR-кодах. Його багаторівневий підхід, що поєднує кілька джерел аналізу, дозволяє знизити ймовірність як помилкового пропуску загроз, так і зайвого блокування безпечних посилань. Це робить його корисним як для індивідуального користувача, так і для впровадження в корпоративні системи кіберзахисту.

# ВИСНОВКИ

У межах бакалаврської кваліфікаційної роботи було розроблено метод та програмний засіб для перевірки QR-кодів на наявність фішингових загроз, що є актуальним завданням у контексті стрімкого зростання цифрових загроз, зокрема атак типу QR-фішинг. В сучасному інформаційному середовищі, де користувачі активно взаємодіють з QR-кодами в оплаті, автентифікації, рекламі та інших сервісах, зловмисники дедалі частіше використовують ці коди для доставки шкідливих посилань, здатних викрасти конфіденційні дані або скомпрометувати пристрої.

Програмний засіб підтримує декодування QR-кодів, автоматичну перевірку URL-адрес кількома незалежними сервісами, формування короткого або розгорнутого звіту, збереження результатів перевірки в історію та зручне виведення висновків для користувача.

Під час тестування система продемонструвала високу надійність при обробці реальних прикладів QR-кодів, включаючи посилання з потенційною фішинговою активністю. Завдяки інтеграції кількох джерел даних та логіці обробки результатів, програма здатна коректно ідентифікувати як явно небезпечні ресурси, так і ті, що викликають сумніви, надаючи користувачеві можливість ухвалити обґрунтоване рішення щодо подальших дій. Була проведена кількісна оцінка ефективності розробленого методу, яка продемонструвала його перевагу порівняно з окремим використанням сервісів Google Safe Browsing, VirusTotal та URLScan. Згідно з результатами тестування, інтегрований підхід, реалізований у програмному засобі, забезпечив підвищення точності виявлення загроз на 21.1% у порівнянні з середнім результатом по кожному сервісу окремо.

Розроблений програмний засіб є ефективним рішенням для персонального або організаційного використання з метою зниження ризику взаємодії з фішинговими ресурсами. Його можна застосовувати у навчальних закладах, на підприємствах, у сфері електронної комерції та інформаційної безпеки.

# СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ


1. Звіт про тенденції загроз фішингу. URL: https://top-ai.com.ua/news/zvit-pro-tendencziyi-zagroz-fishyngu/? (дата звернення: 27.04.2025)
2. The State of Phishing 2023. URL: https://slashnext.com/wp-content/uploads/2023/10/SlashNext-The-State-of-Phishing-Report-2023.pdf (дата звернення: 27.04.2025)
3. Phishing Attacks Rise: ThreatLabz 2024 Phishing Report. URL: https://www.zscaler.com/blogs/security-research/phishing-attacks-rise-58-year-ai-threatlabz-2024-phishing-report? (дата звернення: 27.04.2025)
4. Understanding QR Code Phishing (QRishing). URL: https://medium.com/it-security-in-plain-english/understanding-qr-code-phishing-qrishing-2ab6c79ce9ba (дата звернення: 27.04.2025)
5. What is Quishing or QR Code Phishing? URL: https://www.qrcodechimp.com/what-is-quishing-or-qr-code-phishing/ (дата звернення: 27.04.2025)
6. Machine Learning Aided Static Malware Analysis: A Survey and Tutorial. URL: https://arxiv.org/pdf/1808.01201 (дата звернення: 04.05.2025)
7. Повний посібник з аналізу QR-кодів. URL: https://uk.qrcodechimp.com/qr-code-analytics-guide/ (дата звернення: 04.05.2025)
8. Detecting Quishing Attacks with Machine Learning Techniques Through QR Code Analysis. URL: https://arxiv.org/pdf/1808.01201 (дата звернення: 10.05.2025)
9. Quishing (QR Phishing). URL: https://www.proofpoint.com/us/threat-reference/quishing? (дата звернення: 10.05.2025)
10. On the (In)security of Google Safe Browsing. URL: https://www.inrialpes.fr/planete/people/amkumar/papers/gsb-security.pdf? (дата звернення: 17.05.2025)
11. Exploring the VirusTotal Dataset | An Analyst's Guide to Effective Threat Research. URL: https://blog.virustotal.com/2024/08/VT-S1-EffectiveResearch.html (дата звернення: 17.05.2025)
12. PhishTank FAQ - How does PhishTank work? URL: https://www.phishtank.com/faq.php (дата звернення: 17.05.2025)

13. Threat Actor UAC-0056 Targeting Ukraine with Fake Translation Software. URL: https://www.sentinelone.com/blog/threat-actor-uac-0056-targeting-ukraine-with-fake-translation-software/ (дата звернення: 17.05.2025)
14. ALPHV Ransomware: Analyzing the BlackCat After Change Healthcare Attack. URL: https://www.picussecurity.com/resource/blog/alphv-ransomware? (дата звернення: 17.05.2025)
15. OpenAI says China-linked group tried to phish its employees. URL: https://fortune.com/2024/10/09/openai-china-phishing-employees-hacker-attempt/ (дата звернення: 17.05.2025)
16. Google Ads Used in $4M Crypto Phishing Scam. URL: https://blockchain.news/news/google-ads-used-in-4m-crypto-phishing-scam (дата звернення: 20.05.2025)
17. Insider Threats: Microsoft 365 Account Hijack. URL: https://www.darktrace.com/es/blog/phishing-from-the-inside-microsoft-365-account-hijack (дата звернення: 20.05.2025)
18. Making the world’s information safely accessible. URL: https://safebrowsing.google.com/? (дата звернення: 20.05.2025)
19. Google Developers - Safe Browsing Reference. URL: https://developers.google.com/safebrowsing/reference (дата звернення: 20.05.25)
20. A Deep Dive into VirusTotal: Characterizing and Clustering a Massive File Feed. URL: https://arxiv.org/pdf/2210.15973 (дата звернення: 20.05.2025)
21. URLscan.io Documentation. URL: https://urlscan.io/docs/ (дата звернення: 25.05.2025)
22. Шпаковатий В. О. , Лукічов В. В.. Засіб для перевірки QR-кодів на наявність фішингових загроз. Матеріали LIV науково-технічної конференції підрозділів ВНТУ, Вінниця, 24-27 березня 2025 р. Електрон. текст. дані. 2025. URL: https://conferences.vntu.edu.ua/index.php/all-fitki/all-fitki-2025/paper/view/23700. (дата звернення: 25.05.2025)

# ДОДАТКИ

Додаток А

# ПРОТОКОЛ ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ

Назва роботи: Засіб для перевірки QR-кодів на наявність фішингових загроз

Автор роботи: Шпаковатий Владислав Олександрович

Тип роботи: бакалаврська кваліфікаційна робота

Підрозділ кафедра захисту інформації ФІТКІ, група І БС-21б

Коефіцієнт подібності текстових запозичень, виявлених у роботі системою StrikePlagiarism **1, 67 %**

Висновок щодо перевірки кваліфікаційної роботи (відмітити потрібне)

√ Запозичення, виявлені у роботі, є законними і не містять ознак плагіату, фабрикації, фальсифікації. Роботу прийняти до захисту

☐ У роботі не виявлено ознак плагіату, фабрикації, фальсифікації, але надмірна кількість текстових запозичень та/або наявність типових розрахунків не дозволяють прийняти рішення про оригінальність та самостійність її виконання. Роботу направити на доопрацювання.

☐ У роботі виявлено ознаки плагіату та/або текстових маніпуляцій як спроб укриття плагіату, фабрикації, фальсифікації, що суперечить вимогам законодавства та нормам академічної доброчесності. Робота до захисту не приймається.

Експертна комісія:

Завідувач кафедри ЗІ д. т. н., проф. _____ Володимир ЛУЖЕЦЬКИЙ

Гарант освітньої програми «Безпека інформаційних та комунікаційних систем» к. т. н., доц. _____ Леонід КУПЕРШТЕЙН

Особа, відповідальна за перевірку _____ Валентина КАПЛУН

З висновком експертної комісії ознайомлений(-на)

Керівник _____ Віталій ЛУКІЧОВ

Здобувач _____ Владислав ШПАКОВАТИЙ

# Додаток Б

# Лістинг програмного засобу

## main.py

```
import requests
import time
import os
import re
from datetime import datetime
from urllib.parse import quote
from tkinter import Tk, Label, Button, Entry, Text, END, filedialog, DISABLED,
NORMAL, Menu
from PIL import Image
from pyzbar.pyzbar import decode

GOOGLE_API_KEY = "*****************************"
VT_API_KEY = "*******************************"
URLSCAN_API_KEY = "**********************"

latest_url = ""
google_brief = ""
vt_brief = ""
google_detailed = ""
vt_detailed = ""
urlscan_result = ""

def is_valid_url(url):
    pattern = re.compile(
         r'^(https?://)?'
         r'([a-zA-Z0-9.-]+\.[a-zA-Z]{2,})'
         r'(:\d+)?'
         r'(\/[^\s]*)?$'
    )
    return bool(pattern.match(url))

def extract_url_from_qr(image_path):
    try:
        image = Image.open(image_path)
        decoded = decode(image)
        for obj in decoded:
            data = obj.data.decode("utf-8")
            if data.startswith("http"):
                return data
    except Exception:
        return None
    return None

def check_google_safe_browsing(url, detailed=False):
    try:
        endpoint =
f"https://safebrowsing.googleapis.com/v4/threatMatches:find?key={GOOGLE_API_KEY}"
        headers = {'Content-Type': 'application/json'}
        payload = {
            "client": {"clientId": "url-checker", "clientVersion": "1.0"},
            "threatInfo": {
                "threatTypes": ["MALWARE", "SOCIAL_ENGINEERING",
"UNWANTED_SOFTWARE"],
                "platformTypes": ["ANY_PLATFORM"],
                "threatEntryTypes": ["URL"],
                "threatEntries": [{"url": url}]
            }
        }
```

```
        response = requests.post(endpoint, headers=headers, json=payload)
        response.raise_for_status()
        result = response.json()
        if result.get("matches"):
            if detailed:
                threat_info = result["matches"][0]
                threat_type = threat_info.get("threatType", "невідомо")
                platform_type = threat_info.get("platformType", "невідомо")
                return f"[ЗАГРОЗА] Google: Тип загрози – {threat_type}, Платформа
– {platform_type}"
            else:
                return "[ЗАГРОЗА] Загроза виявлена (Google)"
        else:
            return "[БЕЗПЕЧНО] Безпечне посилання (Google)"
    except Exception as e:
        return f"[ПОМИЛКА] Google: {str(e)}"

def check_virustotal(url, detailed=False):
    try:
        headers = {
            "x-apikey": VT_API_KEY,
            "Content-Type": "application/x-www-form-urlencoded"
        }
        encoded_url = quote(url, safe='')
        scan_response = requests.post(
            "https://www.virustotal.com/api/v3/urls",
            headers=headers,
            data=f"url={encoded_url}"
        )
        scan_response.raise_for_status()
        analysis_id = scan_response.json()["data"]["id"]
        report_url = f"https://www.virustotal.com/api/v3/analyses/{analysis_id}"

        for _ in range(6):
            time.sleep(5)
            report_response = requests.get(report_url, headers=headers)
            if report_response.status_code == 200:
                report_data = report_response.json()
                status = report_data["data"]["attributes"]["status"]
                if status == "completed":
                    malicious =
report_data["data"]["attributes"]["stats"]["malicious"]
                    if malicious == 0:
                        return "[БЕЗПЕЧНО] Безпечне посилання (VirusTotal)"
                    else:
                        if detailed:
                            results = report_data["data"]["attributes"]["results"]
                            details = []
                            for engine, res in results.items():
                                if res["category"] == "malicious":
                                    details.append(f"{engine}: {res['result']}")
                            return f"[ЗАГРОЗА] VirusTotal: {malicious}
виявлень:\n" + "\n".join(details[:10])
                        else:
                            return f"[ЗАГРОЗА] Загроза виявлена (VirusTotal)"
        return "[ПЕРЕВІРКА] VirusTotal: Аналіз триває..."
    except Exception as e:
        return f"[ПОМИЛКА] VirusTotal: {str(e)}"

def check_urlscan(url):
    try:
        headers = {
            "API-Key": URLSCAN_API_KEY,
```

```
            "Content-Type": "application/json"
        }
        payload = {"url": url, "visibility": "public"}
        response = requests.post("https://urlscan.io/api/v1/scan/",
headers=headers, json=payload)
        data = response.json()
        uuid = data.get("uuid")
        result_url = data.get("result")

        if not uuid:
            return "[ПОМИЛКА] URLScan.io: Не вдалося отримати ID"

        for _ in range(5):
            time.sleep(5)
            r = requests.get(f"https://urlscan.io/api/v1/result/{uuid}/",
headers=headers)
            if r.status_code == 200:
                report = r.json()
                verdict = report.get("verdicts", {}).get("overall", {})
                if verdict.get("malicious") or verdict.get("score", 0) > 5:
                    return f"[ЗАГРОЗА] Потенційно небезпечне посилання
(URLScan.io)\n {result_url}"
                return f"[БЕЗПЕЧНО] Безпечне посилання (URLScan.io)\n
{result_url}"
        return f"[ПЕРЕВІРКА] URLScan.io: Перевірте вручну\n {result_url}"
    except Exception as e:
        return f"[ПОМИЛКА] URLScan.io: {str(e)}"

def save_to_history(url, google_brief, vt_brief, google_detailed, vt_detailed,
urlscan_result):
    os.makedirs("history", exist_ok=True)
    now = datetime.now().strftime("%Y-%m-%d %H-%M-%S")
    filename = f"history/{now}.txt"
    with open(filename, "w", encoding="utf-8") as f:
        f.write(f"Час перевірки: {now}\n")
        f.write(f"URL: {url}\n\n")
        f.write(f"[Google Safe Browsing]\n{google_brief}\n")
        if "[ЗАГРОЗА]" in google_brief:
            f.write(google_detailed + "\n\n")
        f.write(f"[VirusTotal]\n{vt_brief}\n")
        if "[ЗАГРОЗА]" in vt_brief:
            f.write(vt_detailed + "\n\n")
        f.write(f"[URLScan.io]\n{urlscan_result}\n")

def show_history():
    history_window = Tk()
    history_window.title(" Історія перевірок")
    history_text = Text(history_window, width=100, height=30)
    history_text.pack(padx=10, pady=10)

    if not os.path.exists("history"):
        history_text.insert(END, "Історія порожня.")
        return

    files = sorted(os.listdir("history"), reverse=True)
    if not files:
        history_text.insert(END, "Історія порожня.")
        return

    for fname in files:
        with open(os.path.join("history", fname), "r", encoding="utf-8") as f:
            history_text.insert(END, f.read())
            history_text.insert(END, "\n" + "=" * 100 + "\n")
```

```
def apply_text_colors(text_widget):
    text_widget.tag_config("safe", foreground="green")
    text_widget.tag_config("threat", foreground="red")
    text_widget.tag_config("warning", foreground="orange")
    text_widget.tag_config("error", foreground="black")

    content = text_widget.get("1.0", END)
    for tag in text_widget.tag_names():
        text_widget.tag_remove(tag, "1.0", END)

    lines = content.split('\n')
    for i, line in enumerate(lines):
        start_index = f"{i + 1}.0"
        if "[БЕЗПЕЧНО]" in line:
            text_widget.tag_add("safe", start_index, f"{i + 1}.end")
        elif "[ЗАГРОЗА]" in line:
            text_widget.tag_add("threat", start_index, f"{i + 1}.end")
        elif "[ПЕРЕВІРКА]" in line:
            text_widget.tag_add("warning", start_index, f"{i + 1}.end")
        elif "[ПОМИЛКА]" in line:
            text_widget.tag_add("error", start_index, f"{i + 1}.end")

def validate_input(*args):
    if entry.get().strip():
        check_button.config(state=NORMAL)
    else:
        check_button.config(state=DISABLED)

def analyze_url(url):
    global latest_url
    results.delete(1.0, END)
    show_more_btn.pack_forget()

    if not url.startswith("http"):
        url = "http://" + url

    if not is_valid_url(url):
        results.insert(END, "[ПОМИЛКА] Невірний формат URL. Введіть коректну
адресу.\n")
        apply_text_colors(results)
        return

    latest_url = url
    results.insert(END, "[Google Safe Browsing]\n[ПЕРЕВІРКА] Перевірка...\n\n")
    apply_text_colors(results)
    root.after(100, lambda: update_google(url))

def update_google(url):
    global google_brief, google_detailed
    google_brief = check_google_safe_browsing(url, detailed=False)
    google_detailed = check_google_safe_browsing(url, detailed=True)
    results.delete(1.0, END)
    results.insert(END, f"[Google Safe Browsing]\n{google_brief}\n\n")
    results.insert(END, "[VirusTotal]\n[ПЕРЕВІРКА] Перевірка...\n")
    apply_text_colors(results)
    root.after(100, lambda: update_virustotal(url))

def update_virustotal(url):
    global vt_brief, vt_detailed
    vt_brief = check_virustotal(url, detailed=False)
    vt_detailed = check_virustotal(url, detailed=True)
    content = results.get(1.0, END)
```

```
    results.delete(1.0, END)
    results.insert(END, content.replace("[ПЕРЕВІРКА] Перевірка...", vt_brief))
    results.insert(END, "[URLScan.io]\n[ПЕРЕВІРКА] Перевірка...\n\n")  # Додано
\n\n для подвійного проміжку
    apply_text_colors(results)
    root.after(100, lambda: update_urlscan(url))

def update_urlscan(url):
    global urlscan_result
    urlscan_result = check_urlscan(url)
    content = results.get(1.0, END)
    results.delete(1.0, END)
    results.insert(END, content.replace("[ПЕРЕВІРКА] Перевірка...",
urlscan_result))
    apply_text_colors(results)

    if "[ЗАГРОЗА]" in google_brief or "[ЗАГРОЗА]" in vt_brief:
        show_more_btn.pack(pady=5)

    danger_count = 0
    for result in [google_brief, vt_brief, urlscan_result]:
        if "[ЗАГРОЗА]" in result:
            danger_count += 1

    summary = "\n[ПІДСУМОК] "
    if danger_count == 0:
        summary += "Посилання безпечне."
    elif danger_count < 3:
        summary += "Посилання потенційно небезпечне."
    else:
        summary += "Посилання НЕБЕЗПЕЧНЕ!"

    results.insert(END, summary + "\n")
    apply_text_colors(results)

    save_to_history(url, google_brief, vt_brief, google_detailed, vt_detailed,
urlscan_result)

def show_detailed_info():
    results.insert(END, "\n Деталі загроз:\n")
    if "[ЗАГРОЗА]" in google_brief:
        results.insert(END, google_detailed + "\n\n")
    if "[ЗАГРОЗА]" in vt_brief:
        results.insert(END, vt_detailed + "\n")
    show_more_btn.pack_forget()
    apply_text_colors(results)

def toggle_theme():
    global current_theme

    if current_theme == "light":
        root.configure(bg="#2e2e2e")
        for widget in root.winfo_children():
            if isinstance(widget, (Label, Button)):
                widget.configure(bg="#2e2e2e", fg="white")
            elif isinstance(widget, Entry):
                widget.configure(bg="#3e3e3e", fg="white",
insertbackground="white")
            elif isinstance(widget, Text):
                widget.configure(bg="#3e3e3e", fg="white",
insertbackground="white")
        current_theme = "dark"
    else:
```

```
        root.configure(bg="SystemButtonFace")
        for widget in root.winfo_children():
            if isinstance(widget, (Label, Button)):
                widget.configure(bg="SystemButtonFace", fg="black")
            elif isinstance(widget, Entry):
                widget.configure(bg="white", fg="black", insertbackground="black")
            elif isinstance(widget, Text):
                widget.configure(bg="white", fg="black", insertbackground="black")
        current_theme = "light"

def choose_qr():
    path = filedialog.askopenfilename(
        title="Оберіть зображення QR-коду",
        filetypes=[("Image Files", "*.png;*.jpg;*.jpeg;*.bmp")]
    )
    if path:
        url = extract_url_from_qr(path)
        if url:
            entry.delete(0, END)
            entry.insert(0, url)
            check_button.config(state=NORMAL)
        else:
            results.delete(1.0, END)
            results.insert(END, "[ПОМИЛКА] QR-код не розпізнано.")
            apply_text_colors(results)

root = Tk()
root.geometry("900x600")
current_theme = "light"
root.title(" Перевірка URL / QR")

Label(root, text="Введіть URL або оберіть QR-код:").pack(pady=5)
entry = Entry(root, width=60)
entry.pack(pady=5)
entry.bind("<KeyRelease>", validate_input)

context_menu = Menu(root, tearoff=0)
context_menu.add_command(label="Вставити", command=lambda: entry.insert(END,
root.clipboard_get()))
entry.bind("<Button-3>", lambda e: context_menu.tk_popup(e.x_root, e.y_root))

Button(root, text="Завантажити QR", command=choose_qr).pack(pady=5)
check_button = Button(root, text="Перевірити", command=lambda:
analyze_url(entry.get()), state=DISABLED)
check_button.pack(pady=5)
Button(root, text="Історія перевірок", command=show_history).pack(pady=5)
theme_button = Button(root, text="Змінити тему", command=toggle_theme)
theme_button.place(relx=1.0, rely=1.0, anchor="se", x=-10, y=-10)  # 10 пікселів
від правого нижнього кута


results = Text(root, width=80, height=20)
results.pack(pady=10)

results_context_menu = Menu(root, tearoff=0)
results_context_menu.add_command(label="Копіювати", command=lambda:
root.clipboard_append(results.selection_get()))

def show_results_context_menu(event):
    try:
        results_context_menu.tk_popup(event.x_root, event.y_root)
    finally:
        results_context_menu.grab_release()
```

```
results.bind("<Button-3>", show_results_context_menu)

results.tag_config("safe", foreground="green")
results.tag_config("threat", foreground="red")
results.tag_config("warning", foreground="orange")
results.tag_config("error", foreground="black")

show_more_btn = Button(root, text="Показати розширену інформацію",
command=show_detailed_info)

root.mainloop()
```

**Додаток В**

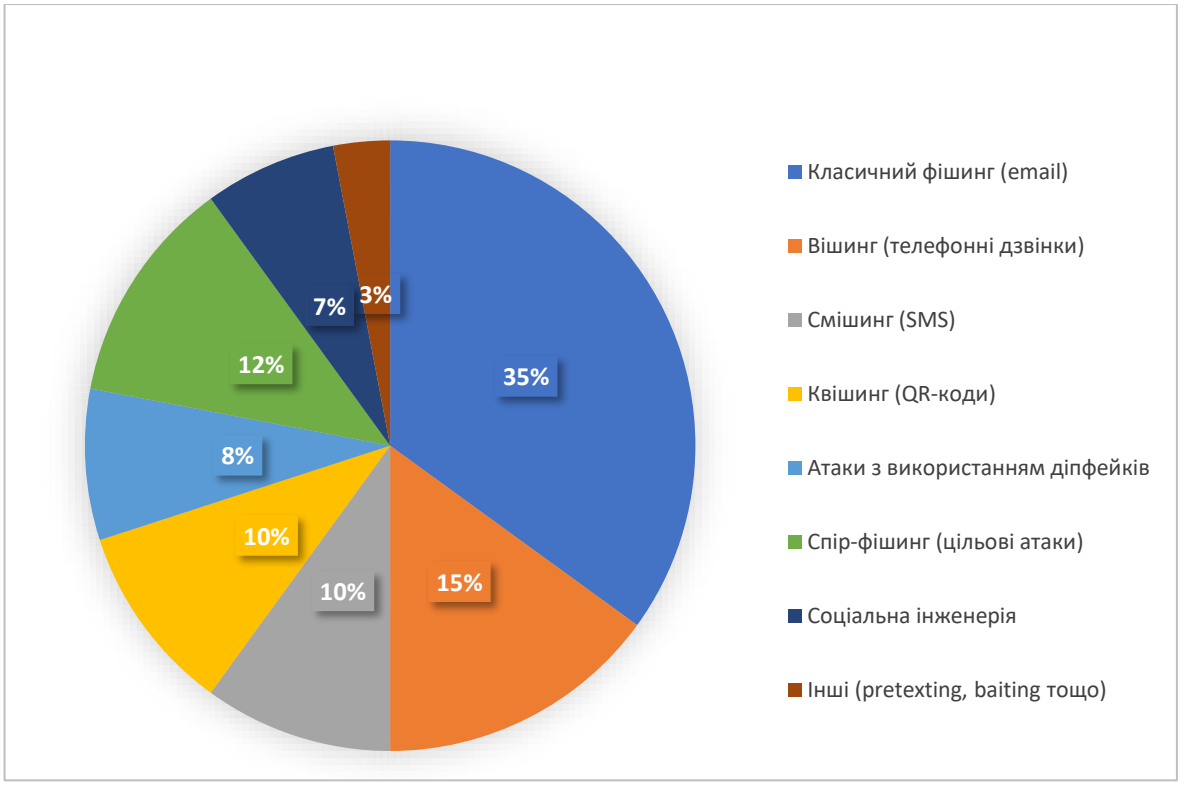
# ІЛЮСТРАТИВНА ЧАСТИНА

## ЗАСІБ ДЛЯ ПЕРЕВІРКИ QR-КОДІВ НА НАЯВНІСТЬ ФІШИНГОВИХ ЗАГРОЗ

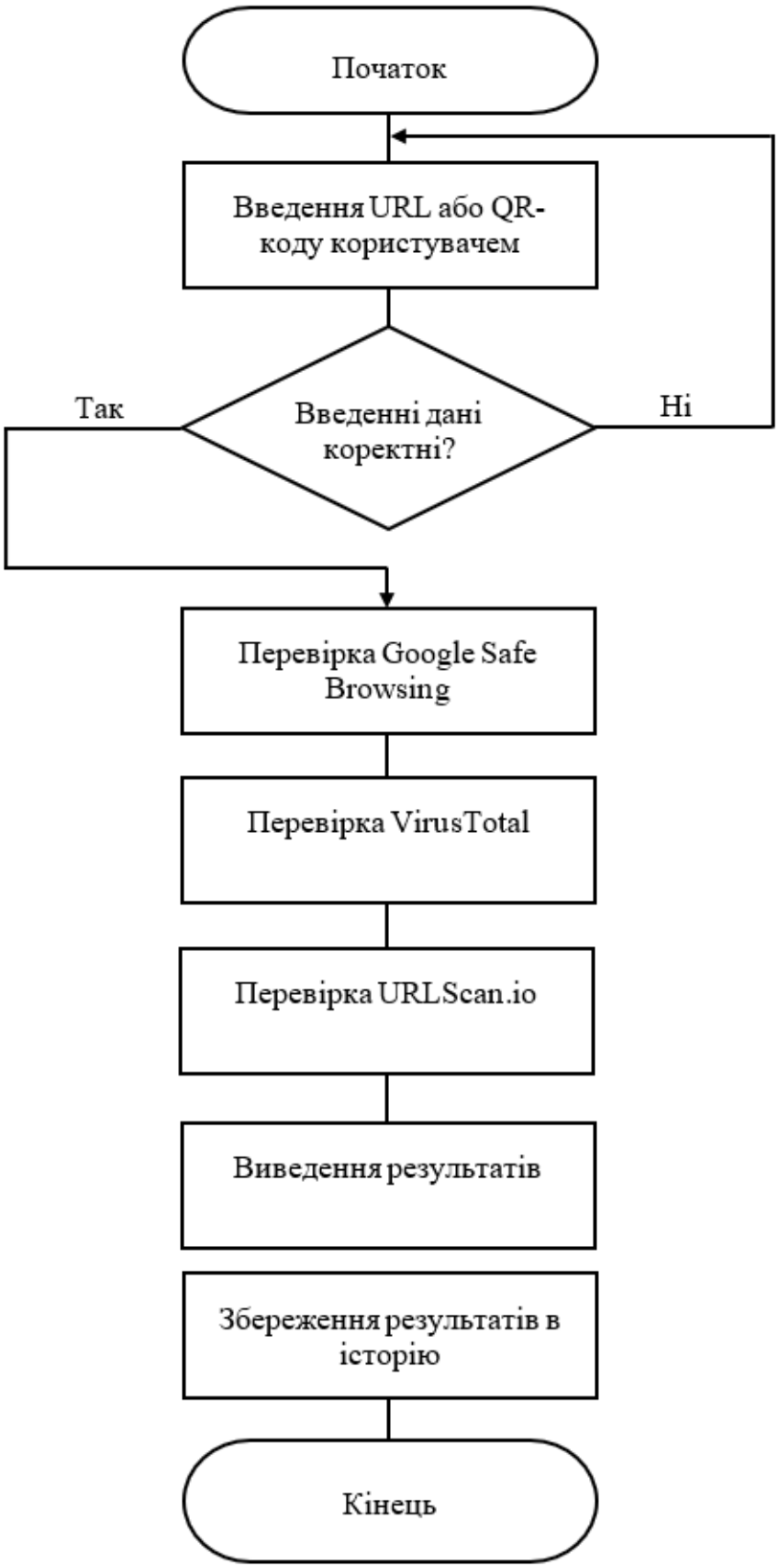
Виконав: студент 4 курсу групи 1БС-21б
спеціальності 125 Кібербезпека
Владислав ШПАКОВАТИЙ

Керівник: к. т. н., доцент, доцент каф. ЗІ
Віталій ЛУКІЧОВ
«12» червня 2025 р.

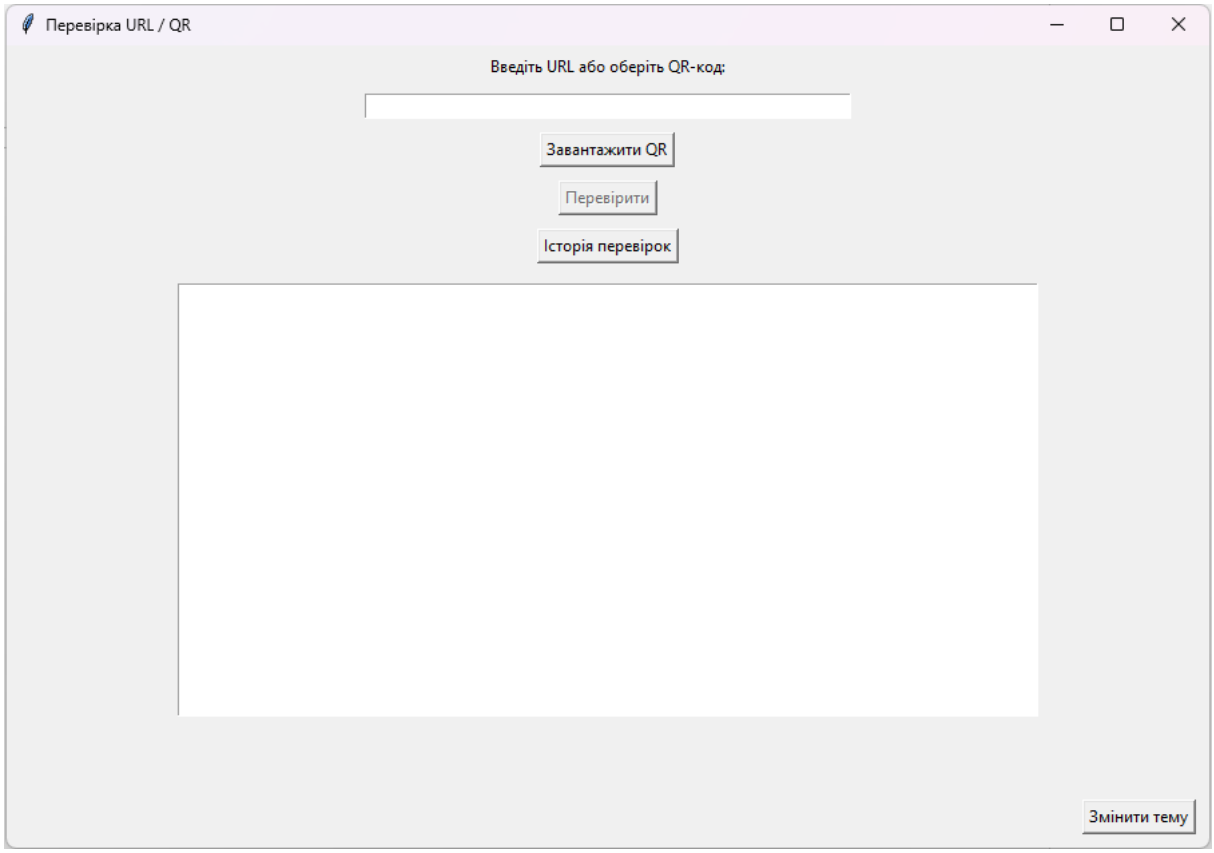
# ДІАГРАМА ВІДСОТКОВИХ ЗНАЧЕНЬ ФІШИНГОВИХ АТАК ЗА ТИПАМИ



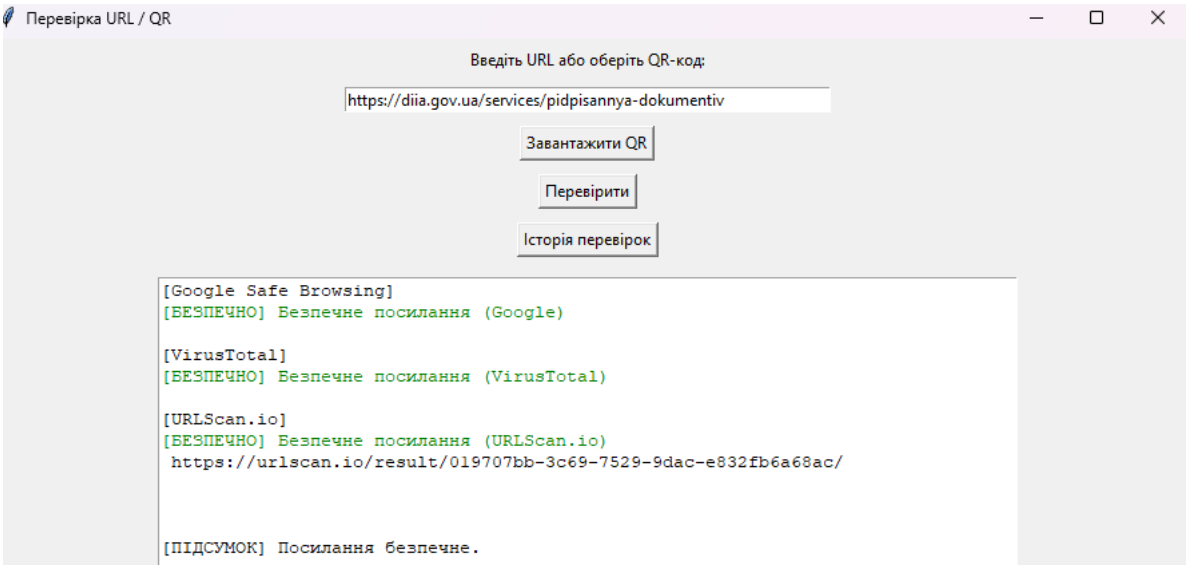
# БЛОК СХЕМА РОБОТИ ПРОГРАМНОГО ЗАСОБУ



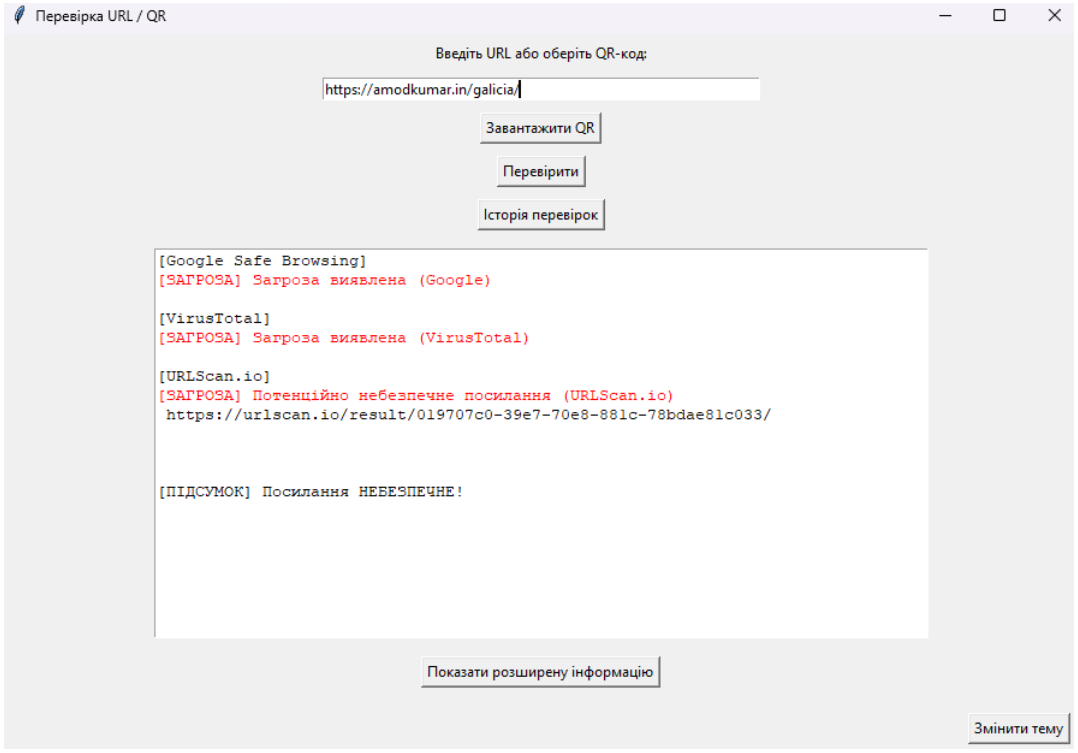
# ГОЛОВНЕ ВІКНО ПРОГРАМИ



# РЕЗУЛЬТАТ ПЕРЕВІРКИ БЕЗПЕЧНОГО QR-КОДУ



# РЕЗУЛЬТАТ ПЕРЕВІРКИ НЕБЕЗПЕЧНОГО QR-КОДУ



## ПОРІВНЯННЯ ЕФЕКТИВНОСТІ МЕТОДІВ

| Сервіс / Метод | Кількість виявлених загроз із 80 | Кількість хибнопозитивних спрацювань | Загальна точність (%) |
|---|---|---|---|
| Google Safe Browsing | 63 | 3 | 80 |
| VirusTotal | 69 | 1 | 88 |
| URLScan | 59 | 3 | 76 |
| Розроблений програмний засіб | 77 | 3 | 94 |

## РЕЗУЛЬТАТИ ОБЧИСЛЕННЯ ПОХИБОК ПЕРШОГО ТА ДРУГОГО

| Сервіс / Метод | TP | FP | FN | TN | $a$ - похибка 1-го роду (%) | $b$ - похибка 2-го роду (%) |
|---|---|---|---|---|---|---|
| Google Safe Browsing | 63 | 3 | 17 | 17 | 15 | 21.25 |
| VirusTotal | 69 | 1 | 11 | 19 | 5 | 13.75 |
| URLScan | 59 | 3 | 21 | 17 | 15 | 26.25 |
| Розроблений програмний засіб | 77 | 3 | 3 | 17 | 15 | 3.75 |