ЛЬВІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ІВАНА ФРАНКА

__Факультет прикладної математики та інформатики__

__Кафедра кібербезпеки__

Курсова робота

Автоматизована система виявлення фішингових сайтів

Студентки _3_ курсу, групи __ПМК-32с_____

напряму підготовки _____кібербезпека_____

_____Гнатик _Соломія_Богданівна_____

(прізвище та ініціали)

Керівник _ас_Грицишин_Остап_Орестович__

(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Національна шкала_____

Кількість балів:_____Оцінка: ECTS

Львів – 2025

Зміст

ВСТУП

Інформаційна безпека є однією з найактуальніших проблем сучасного цифрового світу, де щоденно обробляються величезні обсяги даних та відбуваються численні онлайн-взаємодії. Одним із найбільш розповсюджених і небезпечних видів кіберзлочинності є фішингові атаки.

Фішинг — це вид шахрайства, що полягає у створенні фальшивих вебресурсів або повідомлень, які імітують легітимні сервіси з метою обману користувача для отримання конфіденційних даних, таких як логіни, паролі, номери банківських карток, особисті ідентифікаційні відомості. Зловмисники маскуються під надійні організації — банки, популярні інтернет-сервіси, державні установи, щоб викликати довіру жертв і змусити їх добровільно надати доступ до важливої інформації. Особливістю фішингу є активне використання соціальної інженерії, тобто психологічного впливу на користувачів для введення їх в оману.

За останні роки масштаби фішингових атак значно зросли. За даними різних аналітичних центрів, кількість таких випадків збільшується на десятки відсотків щороку, а збитки, завдані жертвам, оцінюються в мільярди доларів на світовому рівні. Небезпека фішингу полягає не лише у фінансових втратам — часто втрачаються особисті дані, що може призвести до викрадення особистості, зловживань у соціальних мережах, втрати конфіденційної інформації організацій. Особливо вразливими є користувачі, які не мають достатньої кіберграмотності, а також працівники компаній, які можуть стати жертвами цілеспрямованих атак на корпоративні системи.

У зв’язку з цим, розробка автоматизованих систем виявлення фішингових сайтів набуває особливої актуальності. Вони дозволяють оперативно ідентифікувати потенційно небезпечні ресурси, мінімізуючи ризики втрати персональних даних, фінансових ресурсів і репутації. Використання сучасних методів аналізу URL, структури вебсторінок, сертифікатів безпеки та технологій машинного навчання відкриває нові можливості для підвищення точності і швидкості виявлення фішингових загроз.

Мета дослідження: розробка автоматизованої системи виявлення фішингових сайтів, яка забезпечить ефективний аналіз характеристик вебресурсів та дозволить у режимі реального часу попереджати користувачів про можливу небезпеку.

Актуальність теми полягає в тому, що з розвитком інтернет-технологій і збільшенням кількості користувачів мережі зростає й кількість фішингових атак, що робить необхідним створення сучасних цифрових рішень для їх виявлення та запобігання. Автоматизація процесу моніторингу і аналізу значно підвищує ефективність протидії цим загрозам у порівнянні з традиційними методами.

Об’єкт дослідження: процес застосування автоматизованих технологій для розпізнавання фішингових сайтів шляхом комплексного аналізу технічних характеристик та вмісту .

Предмет дослідження: розробка алгоритмів і програмного забезпечення для автоматичного визначення фішингових ресурсів та виявлення можливих недоліків і вдосконалення функціоналу.

Структура роботи: курсова робота складається зі вступу, п’яти розділів, висновку та списку використаних джерел.

РОЗДІЛ 1. ДОСЛІДЖЕННЯ ПРОБЛЕМНОЇ ОБЛАСТІ ТА МОЖЛИВОСТЕЙ СИСТЕМИ.

1.1 Концепція проєкту та цільова аудиторія

У сучасному цифровому світі ефективний захист користувачів від фішингових атак є надзвичайно важливим завданням для забезпечення кібербезпеки. Автоматизована система виявлення фішингових сайтів створена щоб допомогти своєчасно ідентифікувати потенційні загрози, мінімізуючи ризики втрати конфіденційної інформації та фінансових збитків.

Основна ідея проєкту полягає у створенні інтегрованої платформи, яка за допомогою комплексного аналізу URL, контенту та поведінкових ознак вебсайтів автоматично визначає, чи є ресурс фішинговим. Система забезпечує швидке реагування у реальному часі та надає користувачам зрозумілі попередження про можливі загрози.

Цільова аудиторія системи включає:

- Звичайних користувачів Інтернету, які бажають захистити свої персональні дані під час роботи в мережі;
- ІТ-спеціалістів та системних адміністраторів, відповідальних за безпеку корпоративних інформаційних систем;
- Компанії, що надають послуги з кібербезпеки, які можуть інтегрувати розроблену систему в свої продукти для автоматичного виявлення і блокування фішингових ресурсів;
- Державні установи, які займаються кібербезпекою та контролем за захистом персональних даних громадян.

Для підвищення зручності користування та оперативного інформування передбачена інтеграція з Telegram-ботом, який дозволить надсилати користувачам миттєві сповіщення про виявлені фішингові сайти, а також отримувати від них запити на перевірку підозрілих посилань у будь-який час.

Отже, автоматизована система виявлення фішингових сайтів орієнтована на забезпечення високого рівня кібербезпеки за допомогою доступного, оперативного та інтерактивного інструменту, що допомагає користувачам уникати шахрайських ресурсів в Інтернеті.

1.2 Аналіз наявних рішень та технологічних підходів.

У сфері кібербезпеки існує велика кількість інструментів і сервісів, які можуть автоматично виявляти фішингові сайти. Однак багато з них мають певні недоліки, пов’язані з точністю аналізу, швидкістю реакції або складністю інтеграції в існуючі системи.

Одним із поширених рішень є сервіс VirusTotal — це популярний онлайн-сервіс для аналізу файлів і URL-адрес на наявність шкідливого ПЗ, у тому числі фішингових сайтів. Плюси VirusTotal: широкий спектр антивірусних сканерів, швидкий аналіз, безкоштовний доступ і велика база даних загроз. Мінуси: іноді можуть бути хибні спрацьовування, залежність від оновлень антивірусних баз і відсутність глибокого аналізу поведінки сайту.

Іншим прикладом є PhishTank URL Checker — це безкоштовний онлайн-сервіс для перевірки вебпосилань на фішингові загрози, який базується на колективній базі даних, яку підтримує спільнота користувачів.Плюси PhishTank: висока оперативність оновлення бази даних, можливість внесення нових підозрілих сайтів користувачами, а також відкритість і прозорість інформації.Мінуси: залежність від внеску спільноти, що може призводити до затримок у виявленні нових загроз, а також обмежена глибина аналізу, оскільки перевірка базується переважно на співставленні URL, а не на комплексному оцінюванні характеристик сайтів.

Також існує система Google Transparency Report — Safe Browsing Site Status — це сервіс від Google, який дозволяє перевірити, чи є вебсайт безпечним або позначеним як потенційно шкідливий, зокрема фішинговий.Плюси: швидке та надійне оновлення даних завдяки великій мережі Google, інтеграція з браузером Chrome для автоматичного блокування небезпечних сайтів, простий інтерфейс для перевірки URL, а також безкоштовний доступ.Мінуси: сервіс більше орієнтований на загальну безпеку вебресурсів і може не виявляти всі новітні або спеціалізовані фішингові атаки, також відсутня глибока аналітика чи пояснення причин блокування.

Отже, більшість існуючих інструментів орієнтовані на загальний захист і не завжди забезпечують гнучкість та масштабованість, необхідні для широкого впровадження в різних сферах. Створена автоматизована система виявлення фішингових сайтів прагне поєднати високу точність, адаптивність та інтеграцію з сучасними інформаційними платформами, що робить її ефективним рішенням для забезпечення безпеки користувачів у цифровому середовищі.

РОЗДІЛ 2. ФУНКЦІОНАЛ І ОБМЕЖЕННЯ СИСТЕМИ

Функціональні вимоги визначають основні можливості застосунку, які необхідно реалізувати для забезпечення його повної працездатності. Вони описують, що саме система повинна виконувати, які дії доступні користувачам і як застосунок реагує на їхні запити.

Користувачі мають змогу вводити URL-адреси для перевірки на наявність фішингових ознак та отримувати детальний аналіз з безпековим балом і статусом ресурсу — чи є він безпечним, підозрілим або фішинговим. Крім того, вони можуть переглядати загальну статистику сервісу, що включає загальну кількість виконаних перевірок та число виявлених фішингових сайтів. В системі передбачена можливість повідомляти про потенційно шкідливі URL-адреси, які, на думку користувача, були неправильно оцінені або відсутні у базах даних. Також користувачі мають доступ до інформаційних розділів, таких як "Як це працює" та "FAQ", і можуть вільно переміщатися між основними сторінками сайту за допомогою зручного меню навігації.

Нефункціональні вимоги описують характеристики системи, що впливають на якість її роботи, продуктивність, безпеку і зручність використання. Застосунок має працювати стабільно і без збоїв навіть при великій кількості одночасних запитів, а також коректно обробляти помилки, що можуть виникнути через некоректні URL або проблеми з зовнішніми сервісами. Безпека даних є пріоритетом: всі запити і передача інформації між клієнтом і сервером мають бути захищеними, наприклад, за допомогою HTTPS, а система повинна бути стійкою до поширених типів веб-атак. Логи безпеки повинні фіксувати підозрілі дії для подальшого аналізу. Продуктивність системи забезпечує швидке завантаження сторінок і отримання результатів перевірки URL, що не має перевищувати 3–5 секунд при середній швидкості інтернет-з’єднання, а кешування даних із використанням Redis мінімізує час відповіді на повторні запити.

Система спроєктована з урахуванням масштабованості, що дозволяє розширювати її функціонал у майбутньому без необхідності кардинальної перебудови архітектури. Зручність користування забезпечується зрозумілий інтерфейс, який дає змогу швидко вводити URL і отримувати результати. Для сумісності застосунок підтримує основні браузери, такі як Google Chrome, Mozilla Firefox, Microsoft Edge і Safari.

Важливою складовою є підтримка та обслуговування продукту, яка передбачає регулярне оновлення баз даних фішингових сайтів і алгоритмів їх виявлення без зупинки роботи сервісу, а також наявність детального логування, що сприяє легкому моніторингу і налагодженню системи.

2.1 Сценарії використання (Use Cases)

Сценарії використання детально описують, як різні типи користувачів взаємодіють із системою для досягнення конкретних цілей, демонструючи очікувану поведінку застосунку "PhishGuard" у відповідь на дії користувача.

Передбачається, що користувач має доступ до Інтернету, і система "PhishGuard" функціонує у штатному режимі. Процес починається з того, що користувач відкриває головну сторінку застосунку, де бачить спеціальне поле для введення URL. Користувач вводить або вставляє підозрілий URL у це поле, після чого система негайно виконує клієнтську валідацію формату адреси. Якщо формат є некоректним, користувач отримує миттєве повідомлення про помилку, і кнопка для запуску перевірки залишається неактивною, запобігаючи відправці неправильних даних. У разі коректного введення, користувач натискає кнопку "Перевірити", що ініціює показ індикатора завантаження на інтерфейсі, інформуючи про активний процес аналізу.

Фронтенд надсилає запит до PhishGuard.API. Бекенд, отримавши запит, спершу звертається до кешу Redis, щоб перевірити, чи немає вже збережених результатів для цього URL. Якщо URL знайдено в кеші, система оперативно повертає кешовані дані, значно прискорюючи відповідь. Якщо ж URL не є кешованим, PhishGuard.API запускає комплексний аналіз, залучаючи власну бібліотеку PhishDetector. Цей аналіз включає перевірку URL за чорними списками, детальний аналіз домену, оцінку SSL-сертифіката, перевірку DNS-записів та аналіз HTML-контенту сторінки на наявність підозрілих елементів. Результати кожної перевірки та фінальний безпековий бал з відповідним статусом (безпечний, підозрілий, фішинговий) фіксуються у базі даних MongoDBAtlas. Після завершення аналізу, PhishGuard.API повертає ці результати клієнтській частині. Фронтенд, у свою чергу, приховує індикатор завантаження та відображає отримані дані в чіткому та інтуїтивно зрозумілому форматі, використовуючи кольорове кодування для швидкого сприйняття статусу безпеки. На основі наданої інформації користувач може прийняти обґрунтоване рішення щодо подальших дій з перевіреним URL.

Варто зазначити, що у разі некоректного введення URL, фронтенд, завдяки JavaScript, одразу інформує про помилку формату, тримаючи кнопку "Перевірити" неактивною. Якщо ж PhishGuard.API зіткнеться з внутрішньою помилкою під час обробки запиту, на фронтенді відобразиться загальне повідомлення про неможливість виконати перевірку, при цьому деталі помилки будуть зафіксовані у серверних лог-файлах. Крім того, якщо користувач перевищить встановлений ліміт запитів, PhishGuard.API поверне HTTP-статус 429 (Too Many Requests), і фронтенд сповістить про тимчасове блокування можливості подальших перевірок.

РОЗДІЛ 3. СТРУКТУРА ЗАСТОСУНКУ ТА ВИБІР ТЕХНОЛОГІЙ

3.1 Архітектура системи

Проєкт "PhishGuard" є застосунком для перевірки URL-адрес на наявність фішингу. Цей підхід дозволяє забезпечити ефективне управління компонентами системи, покращити її масштабованість і спростити процес розробки та тестування. В даному випадку функціональні частини програми розділені на окремі компоненти, що дозволяє краще організувати розподілену обробку запитів і масштабувати кожен компонент залежно від навантаження.

Застосунок складається з кількох основних компонентів, що взаємодіють між собою через REST API. Кожен з компонентів виконує конкретну роль у загальній архітектурі.

Основними компонентами є:

- PhishGuard.API — сервіс, який відповідає за обробку запитів на перевірку URL, виконання аналізу на фішинг за допомогою різних алгоритмів, збереження результатів перевірок та надання статистичних даних. Побудований на фреймворку Flask.
- Front (HTML, CSS, JavaScript) — клієнтська частина, яка взаємодіє з користувачем і забезпечує інтерфейс для взаємодії з усіма сервісами системи.

PhishGuard.API

Сервіс PhishGuard.API є критично важливим для виявлення фішингових загроз і побудований на легкому та гнучкому фреймворку Flask. Він отримує URL-адреси від користувачів та виконує їх всебічний аналіз, що включає перевірку за низкою параметрів:

- Перевірка за чорними списками: URL порівнюється з відомими базами фішингових сайтів.
- Аналіз домену: Перевіряється вік домену, наявність у ньому символів, що вводять в оману
- Перевірка SSL-сертифіката: Аналізується наявність та валідність SSL-сертифіката, а також термін його дії.
- DNS-записи: Перевіряються DNS-записи домену для виявлення підозрілих конфігурацій.
- Контент сторінки: Аналізується HTML-контент сторінки на наявність підозрілих елементів, таких як скрипти переадресації, форми для введення облікових даних, а також ключові слова, пов'язані з фішингом.

Усе це забезпечується через використання власної бібліотеки PhishDetector, яка агрегує результати цих перевірок та формує загальний "безпековий" бал (final_score) та визначає, чи є URL фішинговим (is_phishing). Результати кожної перевірки зберігаються в базі даних MongoDBAtlas, що дозволяє відстежувати історію сканувань та надавати статистичні дані. Сервіс також надає API для повідомлення про підозрілі URL.

Для оптимізації продуктивності та захисту від зловмисних запитів у PhishGuard.API інтегровані наступні технології:

- Redis: Використовується як бекенд для кешування (Flask-Caching), що значно прискорює повторні запити та зменшує навантаження на основні алгоритми аналізу. Крім того, Redis застосовується для обмеження частоти запитів (Flask-Limiter), забезпечуючи стабільну роботу сервісу та запобігаючи DDoS-атакам.
- Система логування: Застосунок активно використовує вбудовані механізми логування Python. Ведуться два основні типи логів:
  - Загальні лог-файли: Фіксують інформацію про нормальну роботу застосунку, виконання запитів, успішні операції та потенційні попередження.
  - Лог-файли безпеки: Призначені для запису підозрілих дій, спроб несанкціонованого доступу, аномалій у запитах та інших подій, що можуть свідчити про загрозу безпеці. Це допомагає в моніторингу та швидкому реагуванні на інциденти.

Front (HTML, CSS, JavaScript)

Клієнтська частина застосунку, що реалізована за допомогою HTML, CSS та JavaScript, забезпечує зручний інтерфейс для користувачів. Всі запити до серверного компонента PhishGuard.API здійснюються через REST API, що дозволяє отримувати необхідну інформацію і забезпечувати взаємодію з користувачем.

Основними елементами інтерфейсу є:

- Поле для введення URL: Користувач може ввести URL-адресу, яку потрібно перевірити на фішинг.
- Відображення результатів перевірки: Після аналізу на сторінці відображаються детальні результати, включаючи статус безпеки (безпечний, підозрілий, фішинговий), безпековий бал, а також перелік виконаних перевірок з їхніми результатами.
- Загальна статистика: Користувач може переглядати загальну статистику використання сервісу, таку як загальна кількість сканувань та виявлених фішингів.

- Інформаційні розділи: Наявні секції "Як це працює" та "FAQ" для надання додаткової інформації про функціонал сервісу та відповіді на поширені запитання.

Використання HTML, CSS та JavaScript забезпечує швидке завантаження та високу продуктивність клієнтської частини, дозволяючи створити зрозумілий та зручний користувацький інтерфейс для перевірки URL-адрес на фішинг.

3.2 Технологічний стек та середовище розробки

Для розробки PhishGuard було обрано сучасний технологічний стек, який забезпечує ефективність та зручність у подальшій підтримці.

Мова програмування: Python — завдяки широким можливостям для розробки алгоритмів машинного навчання, обробки даних та інтеграції з веб-сервісами.

Аналіз даних: Для перевірки URL та інших характеристик сайтів використовуються різні методи аналізу, зокрема синтаксичний аналіз, перевірка доменних імен, SSL-сертифікатів, заголовків HTTP та інших технічних ознак.

База даних: MongoDBAtlas — вибрана через гнучкість у зберіганні як структурованих, так і неструктурованих даних, що є важливим для роботи з різноманітною інформацією про сайти.

Інтерфейс користувача: Веб-інтерфейс реалізований за допомогою HTML, CSS та JavaScript з використанням стандартних бібліотек, що забезпечує простий і зручний доступ до функцій системи.

Інтеграція: Для оперативного оповіщення користувачів про потенційні загрози передбачена інтеграція з Telegram-ботом, що дозволяє отримувати повідомлення та взаємодіяти із системою через месенджер.

Середовище розробки: Використовується PyCharm з розширеннями для Python, і Git для контролю версій.

3.2.1 Деталізація бібліотек та фреймворків

Проєкт "PhishGuard" базується на ефективному наборі технологій, обраних за їхню функціональність та здатність забезпечити швидку розробку та надійну роботу системи виявлення фішингу.

Основні технології бекенду :

- **Flask:** Легкий фреймворк для створення API, що обробляє запити та керує логікою системи PhishGuard.API.
- **Pymongo:** Драйвер для взаємодії з базою даних **MongoDB**, де зберігаються результати перевірок URL та повідомлення про фішинг.
- **Flask-Caching:** Розширення Flask, що використовує **Redis** для швидкого кешування результатів аналізу. Це значно прискорює повторні перевірки вже відомих URL.
- **Flask-Limiter:** Захищає API від надмірного навантаження, обмежуючи кількість запитів від кожного користувача. Також використовує **Redis** для відстеження лімітів.
- **Requests:** Бібліотека для надсилання HTTP-запитів до зовнішніх джерел. Вона використовується для завантаження чорних списків, отримання вмісту веб-сторінок та здійснення запитів до сервісів Whois.
- **BeautifulSoup4 (bs4):** Допомагає аналізувати HTML-код веб-сторінок для виявлення підозрілих елементів та ключових слів.
- **tldextract:** Використовується для точного виділення домену та піддомену з URL, що важливо для аналізу схожості доменів.
- **python-whois:** Дозволяє отримувати інформацію про реєстрацію домену (вік, власник), яка є одним з показників фішингу.
- **dnspython:** Виконує DNS-запити для перевірки налаштувань домену, допомагаючи виявити аномалії.
- **Certifi:** Забезпечує безпечне HTTPS-з'єднання при роботі із зовнішніми сервісами.
- **Dotenv:** Дозволяє безпечно завантажувати конфігураційні дані (паролі, ключі) з файлу .env.
- **Logging:** Вбудований модуль Python для запису всіх важливих подій системи (інформація, попередження, помилки, події безпеки), що є ключовим для моніторингу та налагодження.

Основні технології фронтенду (HTML/CSS/JavaScript):

- **HTML:** Формує структуру веб-сторінок застосунку, включаючи поля вводу, кнопки, блоки результатів та навігацію.
- **CSS:** Відповідає за зовнішній вигляд інтерфейсу – кольори, шрифти, розташування елементів та адаптацію для різних пристроїв.
- **JavaScript:** Забезпечує інтерактивність: обробляє дії користувача (наприклад, введення URL), відправляє запити до бекенду, динамічно оновлює сторінку та реалізує анімації (як для FAQ).
- **Font Awesome:** Надає іконки для покращення візуального сприйняття інтерфейсу.

3.3 Модель Бази даних

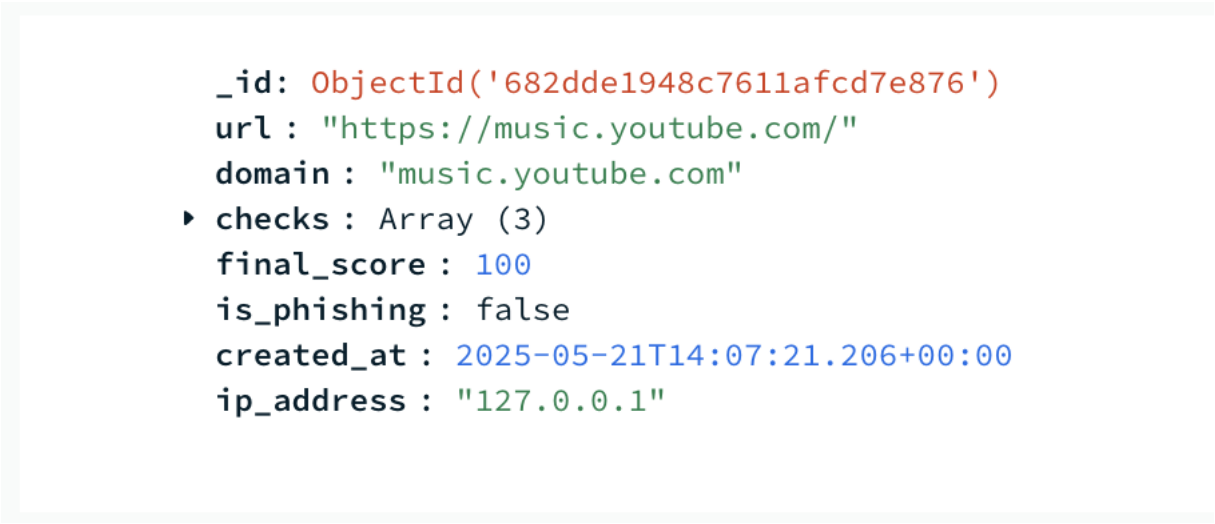

Рис. 3.3. Структура колекції scan_results в MongoDBAtlas

Результати аналізу URL зберігаються в колекції scan_results у MongoDB Atlas у вигляді об'єктів ScanResult. Ці об'єкти містять такі ключові поля:

- url: URL, який був перевірений.
- domain: Домен перевіреного URL.
- final_score: Загальний безпековий бал.
- is_phishing: Булеве значення, що вказує, чи є URL фішинговим.
- checks: Список детальних перевірок, кожна з яких містить description, details, result (наприклад, 'pass', 'warning', 'fail') та score.
- created_at: Час створення запису.

3.4 Роль Redis у кешуванні та лімітуванні

PhishGuard.API активно використовує **Redis** для оптимізації продуктивності та забезпечення стабільності. Одним з ключових напрямків його застосування є кешування. Коли система аналізує URL, що вже був перевірений раніше, або коли надходять повторні запити на той самий ресурс, Redis дозволяє миттєво повернути попередні результати аналізу, уникнувши виконання повного циклу складних перевірок. Це значно скорочує час відповіді для користувачів та зменшує навантаження на серверні ресурси, такі як процесорний час та зовнішні мережеві запити до сервісів Whois або DNS. Такий підхід забезпечує високу швидкість функціонування застосунку, особливо при зростаючому потоці запитів.

Окрім кешування, Redis є фундаментальним компонентом для механізму лімітування частоти запитів. Це життєво необхідно для захисту PhishGuard.API від потенційних зловмисних дій, таких як DDoS-атаки або надмірне споживання ресурсів окремими клієнтами. Завдяки Redis, система може відстежувати кількість запитів від кожної унікальної IP-адреси протягом

певного часового інтервалу. У разі перевищення встановленого ліміту, подальші запити від цього джерела тимчасово блокуються. Це гарантує справедливий розподіл обчислювальних ресурсів між усіма користувачами та підтримує високий рівень доступності та стійкості сервісу в умовах значного навантаження.

3.5 Використання лог-файлів для моніторингу та безпеки

Ефективне логування є невід'ємною складовою забезпечення роботи та надійності PhishGuard.API. Застосунок інтегрований з потужними механізмами логування, які ретельно фіксують усі важливі події, що відбуваються в системі. Це дозволяє мати повне уявлення про функціонування сервісу та оперативно реагувати на будь-які аномалії.

Ведеться ведення двох основних категорій лог-файлів. Перша категорія – це загальні лог-файли, що документують повсякденну роботу застосунку. Тут записуються інформаційні повідомлення про успішне виконання запитів, початок та завершення аналізу URL, взаємодію з базою даних, а також будь-які попередження або некритичні помилки, що виникають у процесі роботи. Ці записи є незамінними для налагодження, моніторингу продуктивності та аналізу загального стану системи.

Друга, і не менш важлива, категорія – це лог-файли безпеки. Вони призначені для виявлення та фіксації подій, які можуть свідчити про потенційні загрози. Сюди включаються записи про спрацювання лімітера запитів, спроби доступу до неіснуючих ресурсів, підозрілі формати вхідних даних або інші аномалії, що відхиляються від нормального поведінкового патерну. Моніторинг цих логів є критично важливим для оперативного виявлення спроб злому, сканування вразливостей або інших зловмисних дій, що дозволяє вчасно вживати заходів для захисту PhishGuard та його користувачів. Ретельне ведення обох типів логів забезпечує повний аудит системних подій та підтримує високий рівень безпеки застосунку.

РОЗДІЛ 4. ІНТЕРАКТИВНІ МОЖЛИВОСТІ ТА ІНТЕРФЕЙС

4.1 Візуальна структура користувацького інтерфейсу

Інтерфейс автоматизованої системи виявлення фішингових сайтів розроблено з акцентом на простоту, інтуїтивність та зручність використання для широкого кола користувачів — від звичайних інтернет-користувачів до фахівців з кібербезпеки. Головною метою є забезпечення швидкого та зрозумілого доступу до основної функціональності системи — перевірки веб-адрес на наявність фішингових ознак.

На головній сторінці центральним елементом є поле вводу URL-адреси, куди користувачі можуть вводити або вставляти підозрілі посилання. Це поле супроводжується підказкою, яка допомагає уникнути помилок при введенні. Поруч розташована кнопка запуску перевірки з чітким маркуванням, яка активується лише за умови коректного формату URL. Після запуску аналізу поряд із кнопкою з’являється індикатор завантаження, що інформує про виконання процесу перевірки.Результати перевірки виводяться у вигляді блоку під полем вводу. Вони містять інформацію про безпеку URL, оцінку ризику та опис виявлених підозрілих ознак, таких як відсутність SSL-сертифіката або невідповідність доменного імені. Для покращення сприйняття даних використовується кольорове кодування: зелений означає безпеку, жовтий — потенційний ризик, а червоний — високий ризик. Користувачі також можуть завантажити детальний звіт або скопіювати інформацію.

Навігаційне меню, розміщене у верхній або бічній частині інтерфейсу, містить посилання на розділи “Головна”, “API”, “Про нас”, “Статистика” та “Контакти”. Воно має мінімалістичний дизайн, щоб не відволікати від основної роботи.

Дизайн інтерфейсу виконано у сучасному мінімалістичному стилі з нейтральною палітрою кольорів і контрастними акцентами, що підкреслюють важливі елементи. Для кращого розуміння складних функцій реалізовано інформаційні підказки, а також передбачено легкий доступ до довідкових матеріалів про фішинг та рекомендації щодо безпеки в інтернеті. Крім того, користувачі можуть повідомляти про помилки або пропонувати покращення, що сприяє постійному вдосконаленню системи.

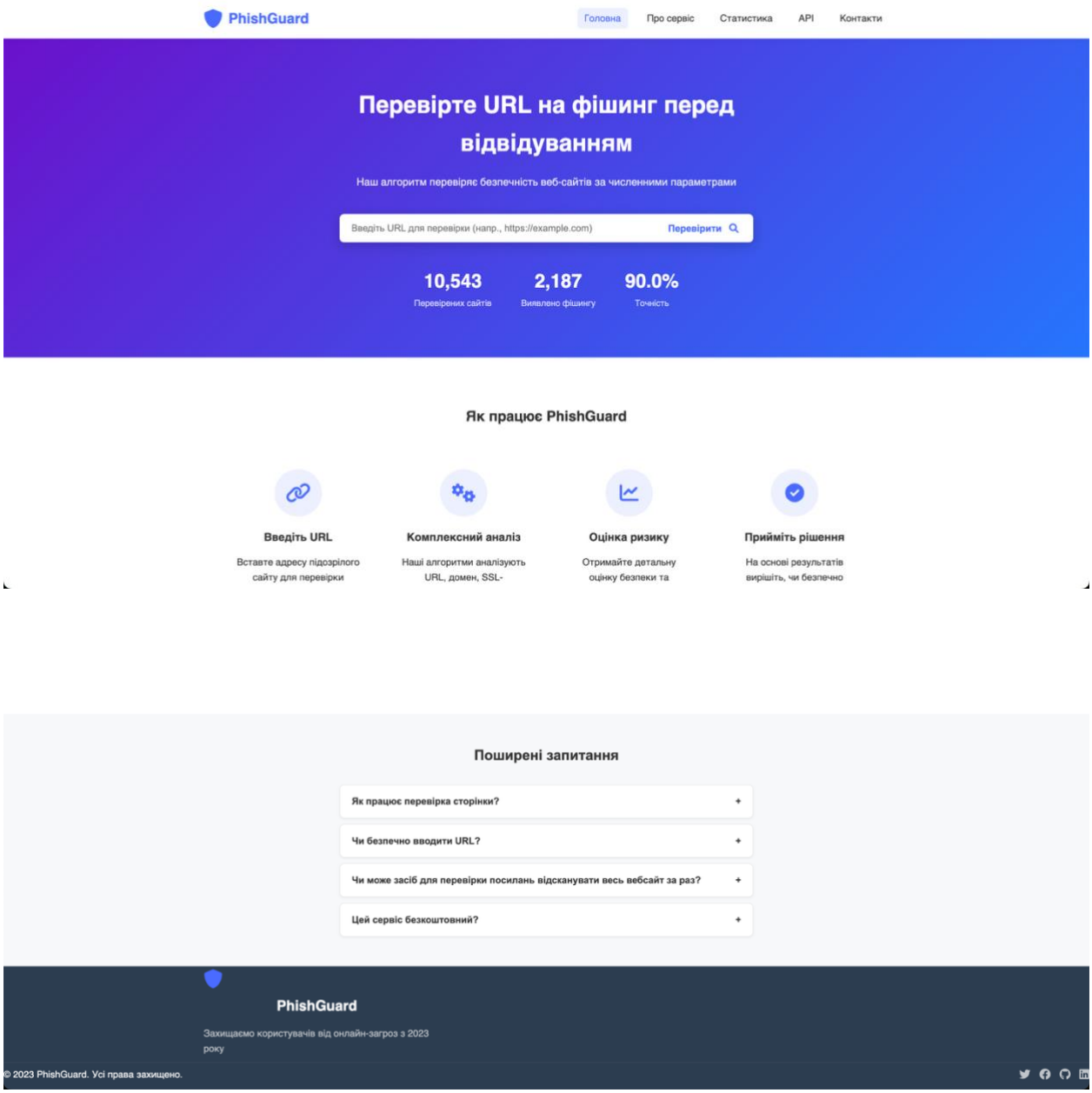

Рис 4.1 Вигляд інтерфейсу користувача

4.2 Дії користувача та обробка подій

Взаємодія користувача зі системою починається з введення URL у спеціальне поле. Система здійснює перевірку формату посилання ще до початку аналізу, і у випадку помилки відразу ж відображає відповідне повідомлення, що допомагає уникнути зайвих дій. Після коректного введення активується кнопка “Перевірити”. Натискання цієї кнопки запускає процес аналізу, під час якого на екрані відображається індикатор завантаження, що інформує користувача про активність системи. Після завершення перевірки користувачу виводиться докладний звіт із висновком щодо безпеки ресурсу, а також деталями виявлених ознак фішингу . Користувач може одразу

скопіювати результати або зберегти їх у вигляді файлу для подальшого використання. Якщо потрібно, можна повторити перевірку іншого URL. Крім того, користувач має можливість самостійно натиснути кнопку “Повідомити про помилку”, завдяки якій це посилання буде додано до бази даних фішингових ресурсів, що допомагає покращити загальну безпеку системи і попереджати інших користувачів про потенційну загрозу.

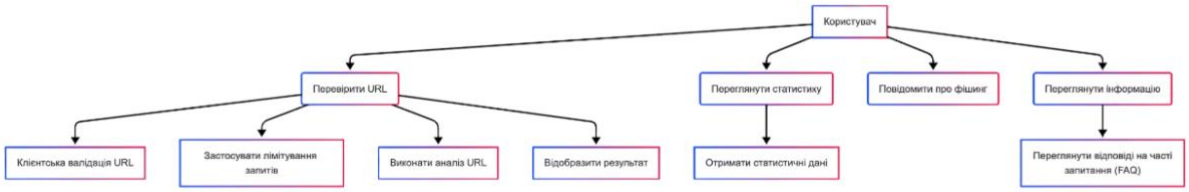

Рис. 4.2. Діаграма сценаріїв використання

РОЗДІЛ 5. ЕФЕКТИВНІСТЬ ТА ПЕРЕВАГИ ЗАСТОСУНКУ

5.1 Ручне тестування: практичні випробування на різних типах URL

Окрім формальних методів тестування, важливим етапом перевірки працездатності та ефективності системи PhishGuard було проведення ручного тестування. Цей етап передбачав інтерактивну перевірку функціональності системи шляхом введення різних типів URL-адрес та аналізу отриманих результатів. Метою було оцінити, наскільки зрозумілим є інтерфейс, як система реагує на коректні та некоректні дані, і наскільки точно вона ідентифікує фішингові загрози в реальних умовах, а також перевірити ефективність механізму лімітування запитів.

Під час ручного тестування використовувалися такі категорії посилань. До них належали загальновідомі легітимні сайти, для перевірки того, що система коректно ідентифікує безпечні ресурси і не видає хибних спрацювань. Були протестовані URL таких ресурсів, як google.com, wikipedia.org, youtube.com та інші популярні веб-сайти. У всіх випадках система коректно визначала їх як "безпечні", що підтверджує надійність базового аналізу. Також використовувалися відомі фішингові сайти, посилання на фішингові ресурси, взяті з публічних баз даних та архівів (наприклад, з Blacklists, що використовуються в PhishDetector, або з сервісів на кшталт PhishTank). Це дозволило перевірити швидкість та точність виявлення вже відомих загроз. Система демонструвала високу ефективність у ідентифікації цих посилань як "фішингових", часто з високим безпековим балом, що вказує на наявність багатьох ознак ризику.

Крім того, тестувалися підозрілі URL з незвичайними символами або структурою. До них входили посилання, що містять нестандартні домени (наприклад, з IDN homograph атаками), IP-адреси замість доменних імен, численні піддомени або інші ознаки, що можуть бути використані шахраями. Наприклад, http://192.168.1.1/login.php або https://www.paypal.com.login.example.com. У цих випадках система коректно позначала такі URL як "підозрілі" або "фішингові", залежно від виявлених факторів ризику (наприклад, відсутність SSL для IP-адреси, або використання субдоменів для приховування справжнього домену). Вводилися також неіснуючі або некоректні URL, тобто посилання, які не відповідають стандартним форматам (наприклад, htp:/example.com або justtext), а також неіснуючі домени. Система ефективно обробляла такі випадки, видаючи повідомлення про некоректний формат URL на клієнтській стороні або повертаючи відповідні помилки з бекенду (наприклад, HTTP 400 Bad Request, якщо запит був некоректним, або показуючи низький бал безпеки з причиною "ресурс недоступний"). Окремо перевірялися URL з різними SSL-сертифікатами, тобто сайти як з дійсними, так і з простроченими, самопідписаними або відсутніми SSL-сертифікатами. Система точно

відображала статус SSL-сертифіката як один з факторів у перевірці, що є важливою ознакою довіри до ресурсу.

Під час ручного тестування особлива увага приділялася візуальному відображенню результатів, а саме тому, наскільки зрозумілими є безпековий бал та статус, і чи допомагає кольорове кодування швидко оцінити ризик.

Також оцінювалася швидкість відповіді, з якою система обробляє запити та повертає результати, особливо для нових, некешованих URL. Проводилася перевірка обробки помилок, а саме коректності повідомлень про помилки та поведінки системи у випадках непередбачених ситуацій (наприклад, відсутність доступу до зовнішніх сервісів).

Ключовим аспектом тестування також була перевірка механізму лімітування запитів (Rate Limiting). Ця функція, реалізована за допомогою Flask-Limiter та Redis, забезпечує захист системи. З однієї IP-адреси (наприклад, локальної машини розробника) послідовно надсилалася велика кількість запитів на перевірку URL, що перевищувала встановлений ліміт (наприклад, 10 запитів на хвилину, як це налаштовано в конфігурації RATE_LIMIT_PER_MINUTE). Спостерігалося, що після досягнення ліміту система повертала HTTP-статус 429 Too Many Requests. На фронтенді при цьому з'являлося відповідне повідомлення про тимчасове блокування можливості подальших перевірок, і пропонувалося спробувати пізніше. Це підтвердило, що Redis ефективно відстежує кількість запитів з кожної IP-адреси та застосовує встановлені обмеження, запобігаючи перевантаженню сервісу та потенційним DDoS-атакам. Після закінчення періоду лімітування, доступ до перевірок автоматично відновлювався.

5.2 Визначення потенційних можливостей для подальшого вдосконалення

Проєкт PhishGuard має великий потенціал для розвитку, і на майбутнє планується впровадження важливих функцій, які дозволять зробити додаток ще більш ефективним, зручним та багатофункціональним інструментом у боротьбі з фішингом.

Основні напрямки подальшого розвитку включають:

1. **Розширення інтеграції з Telegram-ботом:** Наразі Telegram-бот дозволяє перевіряти URL. У майбутньому планується значно розширити його функціональність. Це може включати:

a) **Прийом файлів:** Надання користувачам можливості завантажувати підозрілі файли (наприклад, PDF, DOCX, ZIP-архіви) безпосередньо через Telegram-бот для їхньої перевірки. Це дозволить аналізувати файли на наявність вбудованих шкідливих посилань, макросів або іншого шкідливого вмісту, а також перевіряти їхні хеші за базами відомих шкідливих програм.
b) **Двосторонній зв'язок:** Додавання можливості для бота надсилати користувачам попередження про нові, виявлені фішингові загрози, або про оновлення статусу раніше перевірених URL.
c) **Розширена статистика:** Надання користувачам персоналізованої статистики через бота, наприклад, кількість перевірених ними URL, виявлені загрози та інше.

2. **Інтеграція перевірки файлів на фішинг/шкідливий вміст:** Одним з ключових напрямків є розробка та впровадження модуля для аналізу файлів. Цей функціонал дозволить перевіряти не лише URL, а й файли, які часто використовуються у фішингових кампаніях для доставки шкідливого програмного забезпечення. Це може включати:

a) **Аналіз метаданих файлів:** Вилучення інформації про походження файлу, автора, дати створення/модифікації.
b) **Перевірка хешів:** Порівняння хешів завантажених файлів з базами даних відомих шкідливих програм (наприклад,VirusTotal).
c) **Статичний аналіз:** Розбір вмісту файлів (наприклад, PDF, Office документів) на наявність аномалій, вбудованих об'єктів, підозрілих посилань або скриптів.

3. **Впровадження елементів машинного навчання (Machine Learning) у PhishDetector:** Хоча поточний PhishDetector базується на правилах та евристиках, інтеграція машинного навчання значно підвищить точність виявлення та адаптивність до нових видів атак. Це може включати:

a) **Класифікація URL:** Навчання моделі на великих наборах даних фішингових та легітимних URL для автоматичного визначення їхнього типу. Можна використовувати ознаки, вже вилучені PhishDetector
b) **Аналіз контенту сторінок:** Аналізу тексту на веб-сторінці, виявлення підозрілих фраз, термінів, що вказують на фішинг.
c) **Предиктивний аналіз:** Розробка моделей, які можуть прогнозувати ймовірність фішингу на основі поведінкових ознак (наприклад, швидкість реєстрації домену, аномальна активність).

4. **Розширення джерел чорних списків та інтеграція з Threat Intelligence платформами:**

a) **Додавання нових баз:** Включення додаткових публічних та приватних чорних списків для підвищення покриття відомих загроз.
b) **Інтеграція з Threat Intelligence:** Автоматичний імпорт даних про нові загрози з платформ кіберрозвідки (наприклад, Censys, VirusTotal), що дозволить системі бути в курсі найактуальніших фішингових кампаній.

5. **Розширений користувацький інтерфейс та звітність:**

a) **Детальна звітність:** Надання користувачеві більш детальних звітів про перевірку URL, включаючи пояснення кожного фактора ризику та рекомендації щодо дій.
b) **Візуалізація даних:** Покращення візуалізації статистики та трендів виявлення фішингу.
c) **Історія перевірок:** Можливість для зареєстрованих користувачів зберігати історію своїх перевірок та отримувати сповіщення про зміну статусу раніше перевірених URL.

Ці потенційні можливості розвитку підкреслюють масштабованість архітектури PhishGuard та її адаптивність до мінливого ландшафту кібербезпеки, що дозволить системі залишатися ефективним інструментом у боротьбі з фішингом.

ВИСНОВКИ

У процесі розробки курсової роботи було успішно створено та реалізовано застосунок PhishGuard, орієнтований на ефективне виявлення та попередження фішингових загроз в мережі Інтернет. Застосунок надає користувачам можливість інтерактивної перевірки URL-адрес, що включає комплексний аналіз їхніх характеристик та візуалізацію результатів. Основною метою програми є підвищення рівня кібербезпеки користувачів завдяки оперативній та точній ідентифікації потенційно небезпечних веб-ресурсів.

Важливими перевагами застосунку PhishGuard є його **доступність** (як веб-сервіс), **швидкість реагування** на запити та **персоналізований підхід** до оцінки ризиків для кожного перевіреного URL.

Особливу увагу було приділено зручності інтерфейсу, що дозволяє користувачам без труднощів вводити URL та інтерпретувати результати перевірки. Чітке візуальне відображення безпекового балу та статусу (безпечний, підозрілий, фішинговий) із застосуванням кольорового кодування забезпечує миттєве розуміння рівня загрози. Крім того, архітектура програми, що використовує Flask**,** MongoDB та Redis**,** гарантує її стабільну роботу, масштабованість та здатність обробляти значну кількість запитів, а також ефективно протидіяти перевантаженням завдяки механізму лімітування запитів.

У майбутньому планується подальше вдосконалення застосунку, додавання нових функцій, які дозволять зробити процес захисту ще більш комплексним та інтерактивним. Це включає розширення функціоналу Telegram-бота для прямої взаємодії з користувачами та, що особливо важливо, впровадження модуля для перевірки файлів на фішинг та шкідливий вміст. Цей функціонал дозволить аналізувати не лише посилання, а й потенційно небезпечні файли, що часто використовуються у фішингових атаках, значно посилюючи захист. Також розглядається інтеграція елементів машинного навчання для підвищення точності виявлення нових типів загроз.

Узагальнюючи**,** PhishGuard є потужним і корисним інструментом для виявлення та запобігання фішинговим атакам, а його подальше вдосконалення може розширити його функціональність до комплексного рішення з кібербезпеки, що включатиме перевірку різних типів контенту та активну взаємодію з користувачами.

Усі матеріали цієї курсової роботи, включаючи вихідний код застосунку, доступні на платформі GitHub.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Grinberg M. Flask Web Development: Developing Web Applications with Python. — O'Reilly Media, 2018. — 254 с.
2. Офіційна документація MongoDB - MongoDB Docs. MongoDB Manual [Електронний ресурс].
   Режим доступу: https://www.mongodb.com/docs/manual/
3. Джерело чорного списку - URLhaus: Malware URL Database [Електронний ресурс].
   Режим доступу: https://urlhaus.abuse.ch/downloads/text_recent/
4. Джерело чорного списку -OpenPhish: Public Phishing Feed [Електронний ресурс].
   Режим доступу: https://raw.githubusercontent.com/openphish/public_feed/refs/heads/main/feed.txt
5. Офіційна документація Redis - Redis Labs. Redis Documentation [Електронний ресурс].
   Режим доступу: https://redis.io/docs/
6. Документація бібліотеки requests - Requests: HTTP for Humans. Requests Documentation
   [Електронний ресурс].
   Режим доступу: https://requests.readthedocs.io/en/master/
7. Документація Beautiful Soup - Beautiful Soup Documentation. Beautiful Soup 4.4.0 documentation
   [Електронний ресурс].
   Режим доступу:https://www.crummy.com/software/BeautifulSoup/bs4/doc/
8. Репозиторій проєкту “PhishGuard” на GitHub
   [Електронний ресурс]. Режим доступу: https://github.com/Solyaaa/PhishGuard.git
9. Офіційна документація Flask - Flask documentation
   [Електронний ресурс]
   Режим доступу: https://flask.palletsprojects.com/en/stable/
10. Онлайн-редактор для діаграми - Mermaid. *Mermaid Live Editor* [Електронний ресурс]
    Режим доступу: https://mermaid.live/