Вінницький національний технічний університет

Факультет менеджменту та інформаційної безпеки

Кафедра менеджменту та безпеки інформаційних систем

БАКАЛАВРСЬКА ДИПЛОМНА РОБОТА

на тему:

Розробка програми для виявлення прихованої інформації у зображеннях інтернет-ресурсів

Виконав: студент 4 курсу, гр.1КІТС-20б
спеціальності 125– Кібербезпека
Освітня програма – Кібербезпека
інформаційних технологій та систем
(шифр і назва напряму підготовки, спеціальності)

Горохов А. В.
(прізвище та ініціали)

Керівник: к.т.н., доц., зав. каф. МБІС
Карпінець В. В.
(прізвище та ініціали)
«____» _____ 2024 р.

Рецензент: проф. каф. ОТ
Захарченко С. М.
(прізвище та ініціали)
«____» _____ 2024 р.

Допущено до захисту
Голова секції УБ кафедри МБІС
д.т.н., проф. Яремчук Ю.Є.
“_____”_____2024р.

Вінниця ВНТУ – 2024

Вінницький національний технічний університет
Факультет менеджменту та інформаційної безпеки
Кафедра менеджменту та безпеки інформаційних систем

Рівень вищої освіти І-й (бакалаврський)
Галузь знань 12 – Інформаційні технології
Спеціальність 125 – Кібербезпека
Освітньо-професійна програма - Кібербезпека інформаційних технологій та систем

ЗАТВЕРДЖУЮ
Голова секції УБ, кафедра МБІС

д.т.н., проф. Яремчук Ю.Є.
" 22 " березня 2024 р.

З А В Д А Н Н Я
на бакалаврську дипломну роботу студенту
Горохову Артему Володимировичу
(прізвище, ім`я, по-батькові)

1. Тема роботи Розробка програми для виявлення прихованої інформації у зображеннях інтернет-ресурсів.

Керівник роботи Карпінець Василь Васильович к.т.н.,доц., зав. каф. МБІС
(прізвище, ім`я, по-батькові, науковий ступінь, вчене звання)
затверджені наказом вищого навчального закладу від "11" березня 2024року
№ 80
2. Термін подання студентом роботи 5 червня 2024р
3. Вихідні дані до роботи Електронні джерела, наукові статті по темі, що описують тему бакалаврської дипломної роботи.
4. Зміст текстової частини:
Для досягнення мети було поставлено наступні задачі: дослідити актуальність вибраної теми; здійснити аналіз стеганографічних методів приховування інформації у зображення; проаналізувати принцип роботи цих методів; розробити програмний додаток; реалізувати процес виявлення прихованої інформації; протестувати розроблений додаток;
5. Перелік ілюстративного матеріалу (з точним зазначенням обов`язкових креслень)
Діаграма алгоритму роботи користувача із розробленим програмним додатком.
6. Консультанти розділів роботи.

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Розділ I	Карпінець В. В. к.т.н., доц., зав. Каф. МБІС		
Розділ II	Карпінець В. В. к.т.н., доц., зав. Каф. МБІС		
Розділ III	Карпінець В. В. к.т.н., доц., зав. Каф. МБІС		

7.Дата видачі завдання 22 березня 2024 р.

КАЛЕНДАРНИЙ ПЛАН

№	Назва та зміст етапу	Термін виконання		Примітка
		початок	закінчення	
1	Аналіз предметної області	26.03.2024	11.04.2024	Виконано
2	Аналіз методів розв'язання задачі	13.04.2024	21.04.2024	Виконано
3	Постановка задач розробки веб-сервісу	22.04.2024	29.04.2024	Виконано
4	Дослідження інформаційних джерел	30.04.2024	02.05.2024	Виконано
5	Аналіз варіантів роботи модуля	03.05.2024	08.05.2024	Виконано
6	Розробка алгоритму роботи	09.05.2024	16.05.2024	Виконано
7	Реалізація програмного модулю	16.05.2024	21.05.2024	Виконано
8	Тестування програмного модулю	2105.2024	22.05.2024	Виконано
9	Оформлення матеріалів до захисту БДР	22.05.2024	27.05.2024	Виконано

Студент _____ Гороховський І. В.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Карпінець В. В.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Бакалаврська дипломна робота складається з 81 сторінок формату А4, на яких є 31 рисунків, список використаних джерел містить 35 найменувань.

Метою роботи є розробка програми для виявлення прихованої інформації в інтернет ресурсах, використання.

У першому розділі роботи досліджено актуальність розробки програми, здійснено аналіз існуючих проблем.

У другому розділі роботи здійснено розробку структури та функціоналу додатку, що передбачає просте застосування і роботу для швидкого виявлення прихованої інформації в зображенні; застосування в програмі Stegdetect, зчитування інформації і декодування UTF - 8; здійснено вибір та обґрунтування обраних засобів програмування для реалізації розробки.

У третьому розділі роботи здійснено розробку графічного інтерфейсу програмного засобу, описано ключові елементи програмної реалізації, описано інструкцію користувача для роботи з ресурсом, здійснено тестування розробленої програми.

Ключові слова: програма, прихована інформація, зображення.

ANNOTATION

The bachelor's thesis consists of 81 A4 pages, featuring 31 figures, and includes a list of sources containing 35 titles.

The aim of the thesis is to develop a program for detecting hidden information on internet resources. The first chapter examines the relevance of developing the program and analyzes existing problems.

In the second chapter, the structure and functionality of the application are developed, aiming for easy application and quick detection of hidden information in images. The chapter discusses the implementation of StegDetect, reading information, and decoding UTF-8, and provides a rationale for the chosen programming tools for the development.

The third chapter involves the development of the graphical interface of the software, describes the key elements of the software implementation, provides a user manual for working with the resource, and details the testing of the developed program.

Keywords: program, hidden information, images.

ЗМІСТ

ВСТУП

Актуальність З настанням інформаційної ери все більше людей використовують мобільні пристрої для спілкування, роботи та творчості. Це приносить великі зручності в наше життя та роботу. Але з кожним разом збільшуються проблеми безпеки. Наприклад, особиста конфіденційність підвергається прослуховуванню, поширенню, крадіжці робіт з авторським правом і так далі.

У вирішенні цих проблем приділяється значна увага захисту конфіденційності та авторських прав за допомогою приховання інформації. Приховування інформації означає, що секретна інформація приховується в кришечному зображенні, використовуючи деякі характеристики цього зображення, і під час передачі кришечного зображення детектором не виявляються аномалії, щоб стего-зображення можна було безпечно передати отримувачу. Отримувач витягує секретну інформацію за допомогою певного алгоритму для реалізації секретного зв'язку. Серед них секретною інформацією може бути текст, зображення і так далі.

Під час процесу приховування зазвичай метод полягає в перетворенні секретної інформації в послідовність бітів і приховуванні бітової інформації в кришечному носії. Крім того, зображення може бути безпосередньо приховане в іншому зображенні, або секретна інформація може відповідати словнику відображення, і інформація, що містить об'єкти відображення, може бути передана отримувачу. Стеганографія є засобом прихованого спілкування і має велике значення для національної безпеки та військових справ. Однак стеганографія також використовується людьми з нечесними намірами під час захисту безпеки мережевого зв'язку. Насправді, приховання інформації також використовується в шпигунстві, терористичних актах, злочинності та інших активностях в останні роки.

У таких умовах ефективний нагляд за стеганографією та запобігання її зловживанням або незаконному застосуванню стало нагальною потребою військових та безпекових відомств різних країн. Таким чином, стеганаліз здобув широку увагу та розвиток в розвитку приховання інформації. **Метою роботи є** розробка програми виявлення прихованої інформації у зображеннях інтернет-ресурсів.

Задачами дослідження є

- аналіз актуальності дослідження обраної галузі та дослідження існуючих аналогів у вигляді програм для виявлення прихованої інформації;
- вибір та обґрунтування обраних засобів програмування для розробки програми.
- розробка структури та функціоналу програми
- проектування та розробка інтерфейсу користувача
- реалізація програмного засобу
- тестування розробки та аналіз отриманих даних

Об’єкт дослідження: програма для виявлення прихованої інформації у зображеннях інтернет-ресурсів**.**

Предмет дослідження: процес розробки програма для виявлення прихованої інформації у зображеннях інтернет-ресурсів.

Практична цінність: розроблено програмний продукт, який являє собою програму в якій задіяна бібліотека для парсингу, аналізи HTML-коду, готовий програмний інструмент Stegdetect для виявлення прихованої інформації в інтернет ресурсах.

1.ЗАГАЛЬНИЙ АНАЛІЗ ЗАСТОСУВАННЯ ПРОГРАМИ ВИЯВЛЕНННЯ ПРИХОВАНОЇ ІНФОРМАЦІЇ У ЗОБРАЖЕННЯХ ІНТЕРНЕТ-РЕСУРСІВ

В даному розділі здійснимо аналіз обраної галузі, а самме визначимо актуальність дослідження та розробки впровадження інформаційних технологій для надання захисних послуг.

1.1 Актуальність дослідження обраної галузі

Технологія приховування інформації має довгу історію. Спочатку люди використовували своє волосся для приховування секретної інформації, ховаючи секретні дані в своїй шкірі голови та чекаючи, поки волосся відросте, щоб передати військову інформацію.

У добу швидкого розвитку комп'ютерів сфера приховування інформації розвивалася стрімко. В галузі приховування інформації в зображеннях на ранніх етапах найбільш представницьким алгоритмом приховування інформації був алгоритм стеганографії з просторовим найменш значущим бітом (LSB). У цьому алгоритмі секретна інформація приховується в найменш значущому біті кожного пікселя, використовуючи нечутливість людського ока до кольору, для передачі секретної інформації. Для кольорових зображень, які зазвичай складаються з трьох каналів: червоного (R), зеленого (G) та синього (B), кожен з яких займає 8 біт, від 00x00 до 0Xff, стеганографія LSB передбачає модифікацію найменш значущого біта кольорового компонента RGB. Наприклад, для каналу R припустимо, що R(x, y) = 11011010, найменш значущим бітом є останній біт 0. Якщо прихований секретний біт дорівнює 1, найменш значущий біт 0 змінюється на 1, і остаточне значення R(x, y) = 11011011; якщо прихований секретний біт дорівнює 0, найменш значущий біт не модифікується, R(x, y) = 11011010. Ємність приховування в алгоритмі LSB стеганографії дуже вражаюча, але вона важко протистоїть статистичним характеристикам. Серед методів стеганографії в просторі секретна інформація приховується головним чином шляхом обчислення значень пікселів. Типові методи

включають заміну LSB [Wu, Wu, Tsai та ін. (2005)], відповідність LSB [Mielikainen (2006); Xia, Wang, Sun та ін. (2014); Xia, Wang, Sun та ін. (2016)], стеганографія зображень на багатобітній площині (MBPIS) [Nguyen, Yoon та Lee (2006)], алгоритм на основі гістограми [Li, Chen, Pan та ін. (2009)], кольорова палітра [Johnson і Jajodia (1998)], система числових баз MBNS [Zhang та Wang (2005)], модуляція індексу квантованих значень (QIM) [Chen і Wornell (2001)] та інші [1].

Для стеганографії в частотній області секретна інформація приховується головним чином шляхом модифікації деяких специфікованих частотних коефіцієнтів. Алгоритми перетворення включають дискретне косинусне перетворення (DCT), перетворення Фур'є, дискретне вейвлет-перетворення (DWT) та інші. Методи стеганографії зазвичай діляться на дві категорії: стеганографія JPEG [Westfeld (2001); Provos і Honeyman (2001); Sallee (2003); Provos і Honeyman (2003)] і стеганографія на основі дискретного вейвлет-перетворення [Al-Ataby і Al-Naima (2008); Yang і Deng (2006); Chen і Lin (2006); Talele і Keskar (2010)].

Для підвищення безпеки передачі секретної інформації запропоновані адаптивні стеганографічні алгоритми для зображень, такі як S-UNIWARD [Holub, Fridrich і Denemark (2014)], WOW [Holub і Fridrich (2012)], HUGO [Pevný, Filler і Bas (2010)]. Ці алгоритми вибирають області зі складною текстурою, встановлюючи поріг спотворення відповідно до спотворення вбудовування пікселів зображення. Як видно на рис. 1, тут представлено порівняння ефектів у HUGO, S-UNIWARD і WOW. З порівняння прихованих зображень і вихідних зображень важко візуально виявити аномалії прихованих зображень [2].

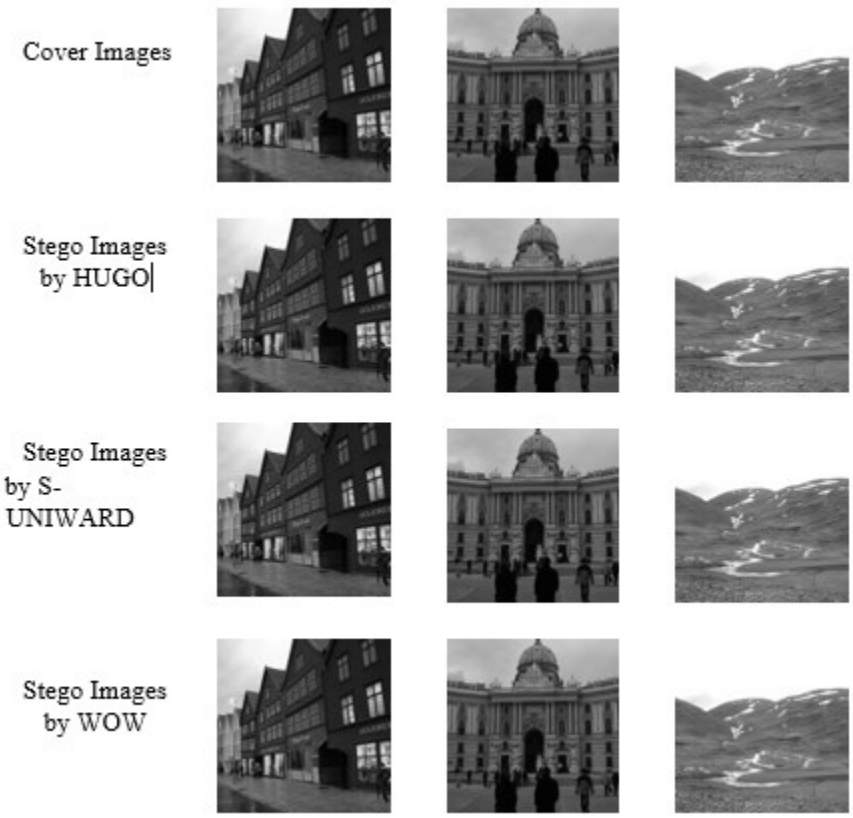


Рисунок 1.1 - Порівняння діаграм ефектів у HUGO, S-UNIWARD і WOW

GAN (генеративно-змагальна мережа) функціонує за рахунок протидії між генератором і дискримінатором, що дозволяє генератору створювати зображення, які можуть протистояти дискримінатору. У базовому застосуванні GAN, генератор імітує розподіл об'єктів, і кожен результат імітації подається на дискримінатор для класифікації на реальні або підроблені зображення. Якщо дискримінатор визначає, що зображення є згенерованим, цей результат передається назад генератору, який на основі цього зворотного зв'язку знову генерує розподіл зображення. Після багаторазового циклічного повторення цього процесу, в кінцевому підсумку генерується досить реалістичне зображення [3].

У цьому процесі ми знаходимо певну схожість між протистоянням генератора і дискримінатора та протистоянням між стеганографією і стеганалізом. Як показано на Рис. 1.2, процес стеганографії полягає у створенні стеганографічних зображень шляхом приховування секретного повідомлення у вихідних зображеннях за

допомогою алгоритму вбудовування. Для отримувача секретна інформація зі стеганографічних зображень вилучається за допомогою алгоритму вилучення; для стеганалізу, після того як стеганографічні зображення стають загальнодоступними, детектор визначає, чи є зображення вихідним або стеганографічним, визначаючи, чи містять вони секретне повідомлення.

У GAN генеративна мережа створює зображення шляхом введення шуму в генератор. Дискримінативна мережа подає підроблені та реальні зображення на дискримінатор, і дискримінатор дає результат, чи є зображення реальним, тобто визначає справжність згенерованого зображення.

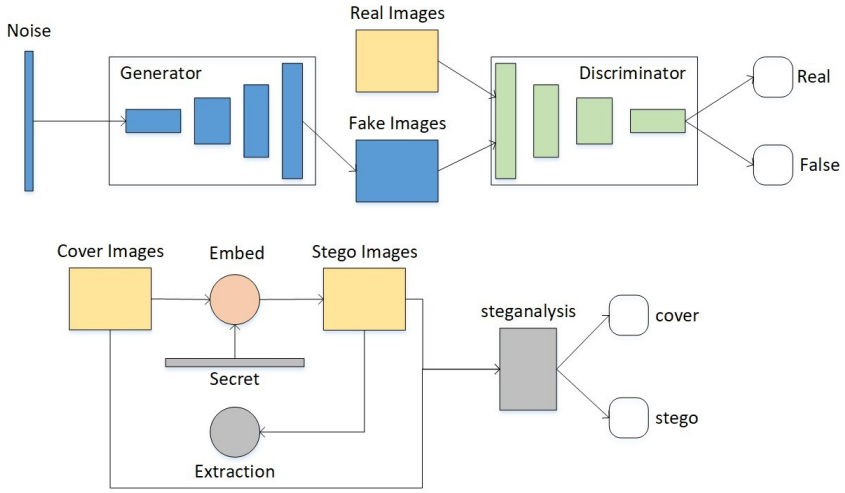


Рисунок 1.2 - Порівняння між стеганографією та відповідним стеганалізом і GAN

Після аналізу стає зрозуміло, що структура GAN повністю відповідає структурі стеганографії. Генеративна мережа відповідає стеганографії для створення стегозображення, а дискримінаційна мережа відповідає стеганалізу для визначення того, чи є зображення фальшивим (стегозображенням). Тому багато статей застосовують GAN до стеганографії [4]. Експериментально доведено, що поєднання стеганографії та GAN робить процес стеганографії більш надійним, а отримане стегозображення більш прихованим і безпечним. Загальна функція оптимізації GAN показана в рівнянні (1).

$$\min_G \max_D V(D, G) = \mathbb{E}_{x \sim P_{data}(x)}[\log(D(x))] + \mathbb{E}_{z \sim P_z(z)}[\log(1 - D(G(z)))] \quad (1)$$

де x представляє вхідні дані, $P_z(z)$ – це шумова змінна, $P_{data}(x)$ – це реальні дані, а $D(x)$ представляє ймовірність того, що x походить від реальних даних, а не згенерованих.

Цей метод вперше був запропонований Гейзом та Данезісом [Hayes and Danezis (2017)]. Вони визначили трьохсторонню гру, де Аліса, Боб і Єва. Аліса та Боб намагаються приховати секретну інформацію в зображенні для таємного спілкування, а Єва прослуховує їхню розмову та визначає, чи містить вона секретну інформацію. У цьому процесі всі три сторони є нейронними мережами. Аліса виступає як генератор стеганографічного конструктора, Єва – як дискримінатор стеганалізу, а Боб – як екстрактор. Стегозображення, створене генератором, коригується відповідно до зворотного зв'язку від стеганалізу, тобто дискримінатора. Боб витягує інформацію з бітів зі стегозображення.

Волхонський та ін. [Volkhonskiy, Nazarov, Borisenko et al. (2017)] запропонували SGAN (Steganographic Generative Adversarial Networks), який головним чином додає дискримінатор, заснований на стеганалізі з використанням GAN. Як показано на Рис. 1.3, структура SGAN складається з генератора і двох дискримінаторів: дискримінатора і стеганализатора. Роль дискримінатора полягає в тому, щоб визначити, чи є зображення справжнім, а стеганализатор використовується для оцінки, чи містить зображення секретну інформацію. Загальна функція оптимізації показана в рівнянні (2), де функція оптимізації стеганализатора додається до функції оптимізації моделі GAN. Цей метод знижує рівень виявлення стеганалізу, роблячи приховування інформації більш безпечним. SGAN збільшив дискримінатор на основі DCGAN [Radford, Metz and Chintala (2015)]. Візуальні ефекти SGAN показані на Рис. 1.4.

$$\min_G \max_{D, S} V(D, S, G) = \alpha (\mathbb{E}_{x \sim P_{data}(x)}[\log(D(x))] + \mathbb{E}_{z \sim P_z(z)}[\log(1 - D(G(z)))]) +$$

\[(1 - \alpha) \mathbb{E}_{z \sim P_z(z)}[\log(S(Stego(G(z)))) + \log(1 - S(G(z)))] \quad (2) \]

де \(P_z(z) \) – це шумова змінна, \(P_{data}(x) \) – це реальне зображення, \(Stego(x) \) – це стегозображення, \(\alpha \) – це ваговий параметр, який використовується для контролю важливості функції втрат для \(D \) і \(S \) під час генерації зображення генератором \(G \).

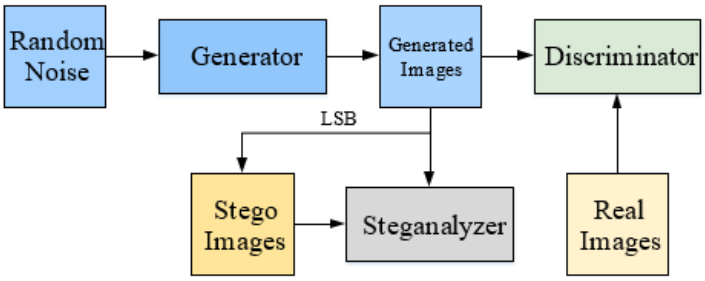


Рисунок 1.3 – Структура SGAN

Рисунок 1.4 – Візуальні ефекти SGAN

Тому розробка програм для виявлення прихованої інформації в інтернет-ресурсах є надзвичайно актуальною у сучасному цифровому світі [5]. Ось деякі ключові моменти, які підтверджують цю актуальність:

− Збільшення використання інтернет-ресурсів: За останні роки інтернет став невід'ємною частиною повсякденного життя, зростає кількість інформації, яка публікується в інтернеті, включаючи зображення.

− Загроза безпеці і приватності: Разом зі зростанням обсягів інформації в інтернеті зростає і загроза її неправомірного використання. Серед них є прихована інформація, яка може бути використана для шпигунства, розповсюдження неправдивої інформації, атак на приватність тощо.

− Застосування стеганографії: Стеганографія стає все більш поширеним методом для приховання інформації в мережі. Використання цього методу може мати негативні наслідки для безпеки та приватності користувачів.

− Необхідність ефективного контролю: З метою забезпечення безпеки мережевого середовища та захисту приватності користувачів необхідно мати ефективні засоби для виявлення та блокування прихованої інформації.

− Зростання кількості кіберзлочинності: Кіберзлочинність, така як крадіжка особистих даних, шахрайство, дестабілізація політичних процесів, стає все більш серйозною загрозою, і виявлення прихованої інформації може допомогти у запобіганні таких атак.

Рисунок 1.5 – Актуальність розробки програми

Розробка програм для виявлення прихованої інформації в інтернет-ресурсах надзвичайно актуальна в сучасному світі цифрових технологій. За останні роки інтернет став основним засобом спілкування, роботи та розваг для мільйонів людей по всьому світу. Це призвело до збільшення кількості інформації, яка публікується в мережі, включаючи зображення, текст, аудіо та відео.

Однак разом із зростанням обсягів інформації збільшуються і загрози безпеці. В інтернеті з'являються різноманітні види кіберзлочинності, включаючи крадіжку особистих даних, розповсюдження фейкових новин, а також порушення авторських прав. Багато з цих загроз пов'язані з використанням стеганографії та інших методів приховування інформації.

У цьому контексті розробка програм для виявлення прихованої інформації має велике значення. Такі програми дозволять ефективно виявляти приховані дані в інтернет-ресурсах, що допоможе забезпечити конфіденційність особистих даних користувачів, запобігти поширенню шкідливої інформації та боротьбі з кіберзлочинністю.

Крім того, з розвитком штучного інтелекту та машинного навчання, можна створити програми, які будуть автоматично виявляти приховані дані та аналізувати їх з метою запобігання можливим загрозам.

Такі програми також можуть сприяти збільшенню довіри до інтернет-ресурсів, підвищити рівень безпеки мережевого середовища та забезпечити захист авторських прав. Таким чином, розробка програм для виявлення прихованої інформації в інтернет-ресурсах є важливим напрямком роботи в області цифрової безпеки та захисту інформації [6].

1.2 Аналіз проблем

Хоча програми для виявлення прихованої інформації в інтернет-ресурсах мають значний потенціал у покращенні безпеки та захисті приватності, вони також можуть виникнути деякі проблеми:

− Можливість обхідних шляхів: Зловмисники можуть розвивати нові методи приховання інформації, щоб уникнути виявлення за допомогою програм. Це вимагатиме постійного оновлення програм та алгоритмів, що призводить до боротьби в зачіпці.

− Проблеми з приватністю інформації: Сама програма може стати об'єктом атаки для зловмисників, які намагаються отримати доступ до конфіденційних даних або використати програму для недобросовісних цілей.

− Потреба в експертному підтримці: Використання програм для виявлення прихованої інформації може вимагати наявності кваліфікованих фахівців для аналізу результатів та вжиття відповідних заходів у разі виявлення загроз.

Помилкова тривожність може призвести до надмірної реакції на невеликі аномалії, що спричинить незручності для користувачів та може викликати втрату довіри до програми. Неоднозначне трактування результатів може ускладнити співпрацю між різними сторонами та унеможливити ефективну реакцію на виявлені загрози.

Поява нових методів обходу може підірвати ефективність програм та вимагати постійного оновлення алгоритмів і стратегій виявлення. Значна кількість трафіку, що обробляється програмою, може призвести до збоїв у мережевій швидкості та викликати проблеми з її продуктивністю.

Отже, важливо усвідомлювати, що разом з перевагами, програми для виявлення прихованої інформації можуть викликати ряд проблем, які потребують уважного

розгляду та вирішення для забезпечення їх ефективності та безпеки, таким чином, хоча програми для виявлення прихованої інформації можуть бути корисними інструментами для захисту мережі, їхнє впровадження також потребує уважного розгляду можливих проблем і врахування відповідних заходів безпеки.

Щоб уникнути можливих проблем з програмами для виявлення прихованої інформації, важливо дотримуватися наступних кроків:

- Забезпечити безпечне та надійне програмне забезпечення, вибираючи перевірені рішення з відкритим вихідним кодом або від надійних виробників.
- Регулярно оновлювати програми, щоб мати доступ до останніх версій з виправленими помилками та новими функціями безпеки.
- Проводити регулярні аудити безпеки для оцінки ефективності програми та виявлення можливих слабких місць.

Загалом, важливо поєднувати технологічні та організаційні заходи для мінімізації ризиків та забезпечення безпеки програм для виявлення прихованої інформації.

1.3 Аналіз подібних програм по виявленню прихованої інформації

Ось деякі з відомих програм для виявлення прихованої інформації (steganalysis tools), які використовуються в галузі цифрової безпеки:

- **StegDetect**: Це програмне забезпечення для виявлення стеганографічних контейнерів в зображеннях
- **StegExpose**: Цей інструмент аналізує зображення на наявність в них прихованої інформації, що може бути використана для стеганографії.
- **OpenStego**: Використовується для виявлення прихованої інформації, яка може бути вбудована в зображення.

Ці програми розроблені з метою виявлення стеганографічних технік, які використовуються для приховування інформації у різних типах мультимедійних файлів. Вони допомагають інформаційним та безпековим службам виявляти та аналізувати потенційні загрози, пов'язані з використанням стеганографії в мережевих комунікаціях.

StegDetect

StegDetect є одним з популярних інструментів для виявлення стеганографії в цифрових зображеннях. Основним призначенням StegDetect є знаходження прихованих даних, які можуть бути вбудовані в зображення за допомогою різних технік стеганографії. Ось детальніша інформація про цей інструмент:

Основні функції та можливості StegDetect:

− Виявлення стеганографічних методів: StegDetect аналізує зображення на предмет наявності прихованої інформації, яка може бути вбудована через різні методи стеганографії, такі як Least Significant Bit (LSB) і інші алгоритми вбудовування даних.

− Підтримка різних форматів зображень: Інструмент підтримує різні формати файлів зображень, такі як JPEG, PNG, BMP тощо, що робить його універсальним для використання в різних умовах.

− Використання наукових методів: StegDetect базується на наукових дослідженнях і алгоритмах для виявлення відмінностей у зображеннях, що можуть вказувати на наявність стеганографії. Він аналізує статистичні характеристики зображень та їхні цифрові сліди для виявлення можливих змін, внесених стеганографією.

− Користувальницький інтерфейс: Інструмент має зручний інтерфейс, що дозволяє користувачам з легкістю завантажувати та аналізувати зображення на предмет наявності прихованої інформації. Він здатний генерувати звіти про результати аналізу.

Застосування в дослідженнях та криміналістиці: StegDetect використовується у наукових дослідженнях, криміналістиці та сферах цифрової безпеки для виявлення та аналізу стеганографічних атак.

StegDetect є потужним інструментом для виявлення прихованої інформації в цифрових зображеннях, який допомагає виявляти потенційні загрози та зберігати безпеку в мережевому середовищі. Його застосування важливо для запобігання кібератак та збереження конфіденційності даних [7].

StegExpose

StegExpose є іншим потужним інструментом для аналізу та виявлення стеганографії в цифрових зображеннях. Основним призначенням цього інструмента є ідентифікація прихованої інформації, яка може бути вбудована в зображення з метою прихованого обміну інформацією або інших кримінальних цілей. Ось детальніша інформація про StegExpose:

Основні функції та можливості StegExpose:

− Аналіз різних типів зображень: StegExpose підтримує аналіз різних форматів зображень, таких як JPEG, PNG, BMP та інші, що дозволяє йому виявляти приховану інформацію незалежно від типу файлу.

− Виявлення методів стеганографії: Інструмент використовує різні алгоритми для виявлення стеганографічних технік, включаючи аналіз цифрових слідів, статистичні характеристики та інші методи для виявлення змін у зображеннях.

− Графічний інтерфейс користувача: StegExpose має інтуїтивно зрозумілий інтерфейс, що дозволяє користувачам завантажувати та аналізувати зображення з легкістю. Він також забезпечує зручний спосіб перегляду результатів аналізу.

− Використання в криміналістиці та безпеці: StegExpose є популярним інструментом у сфері цифрової криміналістики, де використовується для дослідження та розслідування кримінальних справ, пов'язаних зі стеганографією та цифровою безпекою.

− Аналіз цифрових слідів: Інструмент використовує аналіз цифрових слідів у зображеннях для виявлення навмисно внесених змін, що можуть свідчити про наявність стеганографії.

StegExpose є важливим інструментом для виявлення стеганографії у цифрових зображеннях, що використовується у сфері кібербезпеки та цифрової криміналістики. Його можливості аналізу та виявлення дозволяють ідентифікувати приховані дані і сприяють підвищенню безпеки та захисту інформації в цифровому середовищі [8].

OpenStego

OpenStego є відомим інструментом для стеганографії, який дозволяє користувачам приховувати секретні повідомлення всередині цифрових зображень. Ця програма є відкритою та безкоштовною для використання, що робить її доступною для широкого кола користувачів. OpenStego використовує методи стеганографії для вбудовування інформації в зображення таким чином, щоб ці зміни були практично непомітними для людського ока.

Основна мета OpenStego полягає в забезпеченні безпечного приховування даних, що може бути корисним для захисту конфіденційної інформації або для передачі секретних повідомлень. Програма підтримує вбудовування повідомлень у формати зображень, такі як BMP і PNG, і дозволяє користувачам витягувати приховані дані з цих зображень.

Крім того, OpenStego має можливість використання цифрових водяних знаків для захисту авторських прав на зображення. Ця функція дозволяє вбудовувати інформацію про авторські права без видимого впливу на якість зображення.

OpenStego також забезпечує високу надійність і безпеку приховування інформації. Програма використовує шифрування для додаткового захисту даних, що приховуються. Це означає, що навіть якщо хтось зможе виявити приховану інформацію, їм все одно буде потрібно зламати шифр для доступу до неї [9].

Однією з ключових переваг OpenStego є його простота використання. Інтерфейс програми є інтуїтивно зрозумілим, що дозволяє користувачам легко вбудовувати та витягувати приховані повідомлення без необхідності глибоких технічних знань.

Рисунок 1.6 – Зовнішній вигляд програми OpenStego

Розглянемо приклад приховування інформації з допомогою даного додатку, візьмемо текстовий документ, напишемо там деяку інформацію і приховаємо її.

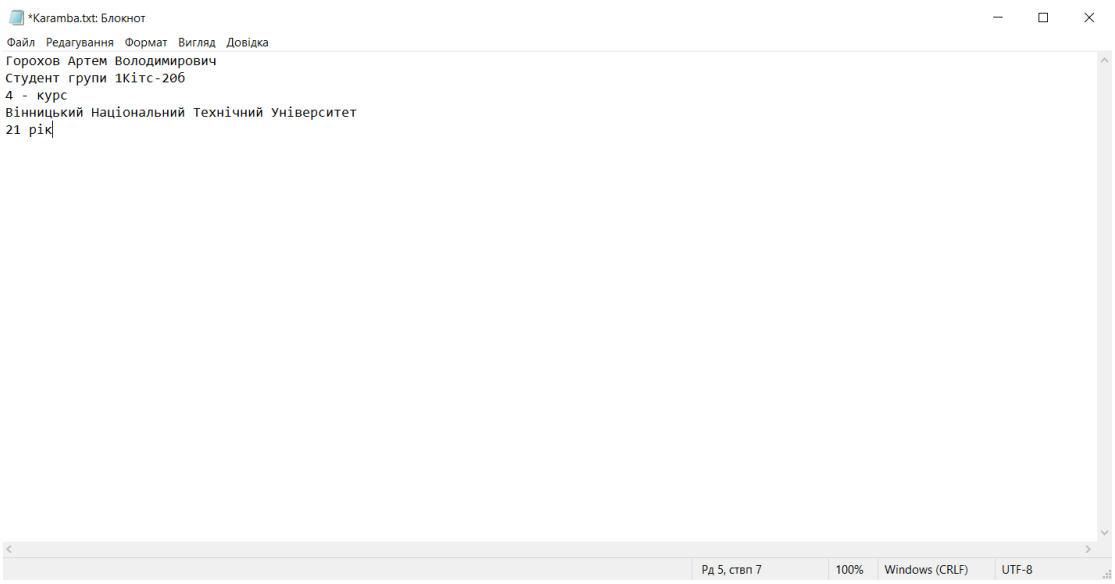


Рисунок 1.7 Умовний прихований текст

Наступним кроком буде скачування або створення зображення формату .bmp, .gif, .jpeg, .jpg, .png або .wbmp.

Рисунок 1.8 – Зображення в форматі .png

Далі користуючись програмою обираємо файл який ми хочемо приховати в іншому файлі, тобто наш текст та отримаємо такий результат.

Рисунок 1.9 – Результат приховування інформації

Як результат, якщо придивитись до правої частини рисунку, то видно що він висвітлений. Це наглядний приклад як працює приховування тексту або інших файлів у зображені. Як наслідок в деяких місцях змінюється зображення, через це для

передачі інформації в якості зображення обирають чорно білі фото, бо в них важче неозброєним оком розгледіти відмінності.

1.4 Висновок до розділу і постановка задачі

На основі проведеного аналізу можна зробити кілька ключових висновків:

− Важливість приховування інформації: Технологія приховування інформації має довгу історію і використовується для забезпечення конфіденційності та безпеки передачі даних. Сучасні методи стеганографії дозволяють приховувати інформацію у зображеннях, що робить її майже невидимою для людського ока.

− Розвиток технологій: З появою та розвитком генеративно-змагальних мереж (GAN) можливості стеганографії значно розширилися. GAN дозволяють створювати реалістичні зображення, що важко відрізнити від справжніх, що робить приховування інформації ще більш ефективним.

− Протистояння стеганографії та стеганалізу: Структура GAN, яка включає генератор і дискримінатор, аналогічна структурі стеганографії та стеганалізу. Це дозволяє застосовувати GAN для створення більш надійних стегозображень, а також для їх виявлення [10].

Постановка задачі

Для розробки програми виявлення прихованої інформації у зображеннях інтернет-ресурсів слід врахувати наступні завдання:

− Розробка або використання алгоритму виявлення: На основі аналізу розробити або використати вже існючий алгоритм виявлення прихованої інформації у зображеннях.

− Інтеграція з веб-інтерфейсом: Створити інтерфейс користувача для інтеграції з веб-ресурсами. Це дозволить автоматично перевіряти зображення на наявність прихованої інформації в реальному часі.

− Тестування та оптимізація: Провести тестування програми на різних наборах даних для оцінки її ефективності та надійності. Врахувати можливість помилкових спрацювань і налаштувати алгоритм для мінімізації таких випадків.

Розробка такої програми сприятиме підвищенню рівня безпеки в інтернеті та допоможе виявляти приховану інформацію, що може використовуватися для незаконних або зловмисних цілей [8].

2.ПРОЕКТУВАННЯ ПРОГРАМИ ВИЯВЛЕННЯ ПРИХОВАНОЇ ІНОФРМАЦІЇ У ЗОБРАЖЕННЯХ ІНТЕРНЕТ РЕСУРСІВ

В даному розділі буде описано проектування програми виявлення прихованої інформації у зображеннях інтернет-ресурсів, що буде включати в себе ось такі етапи: створення блок схеми алгоритму виконная завдання, процесу обробки і виявлення зображень, створення зручного інтерфейсу для користувача.

Також на основі обраних засобів захисту та поставленої мети роботи буде здійснено вибір засобів програмування та його обґрунтування для подальшої практичної реалізації спроектованого додатку.

2.1 Вибір методу стеганографічного аналізу для розробленого додатку

В програмі для виявлення прихованої інформації у зображеннях з інтернет-ресурсів можна використовувати кілька методів стеганографічного аналізу, щоб підвищити точність і ефективність виявлення [12]. Ось деякі з них:

1.Аналіз статистичних властивостей зображень:

- Цей метод базується на виявленні змін у статистичних властивостях зображень, таких як гістограма інтенсивності пікселів або коефіцієнти Фур'є-аналізу. Стеганографічні методи часто змінюють ці властивості, що дозволяє їх виявити [13].

2. Метод розрізнення найменших значущих бітів (LSB):

- Цей метод полягає у виявленні змін у найменш значущих бітах пікселів зображення. Стеганографія LSB змінює ці біти, що може бути виявлено за допомогою відповідних алгоритмів аналізу [14].

3.Методи на основі гібридного аналізу:

- Ці методи комбінують кілька технік для виявлення стеганографічного вмісту, таких як поєднання статистичного аналізу та аналізу найменш значущих бітів [15].

4. Методи з використанням машинного навчання:

- Алгоритми машинного навчання можуть бути натреновані на великій кількості зображень з відомим стеганографічним вмістом та без нього. Після навчання такі моделі можуть ефективно класифікувати нові зображення [16].

5. Аналіз на основі частотного домену:

- У цьому підході використовується дискретне косинусне перетворення (DCT), дискретне хвильове перетворення (DWT) або інші перетворення для аналізу змін у частотному домені зображень. Стеганографічні методи часто вносять зміни в ці коефіцієнти, що дозволяє їх виявити [17].

6. Аналіз на основі фрактальних властивостей:

- Цей метод базується на аналізі фрактальних властивостей зображення, таких як самоподібність на різних масштабах. Зміни, викликані стеганографічним вмістом, можуть вплинути на ці властивості [18].

7. Методи з використанням нейронних мереж і глибокого навчання:

- Глибокі нейронні мережі можуть бути застосовані для виявлення складних патернів у зображеннях, що вказують на наявність стеганографічного вмісту. Такі моделі можуть автоматично виявляти приховані дані, навіть якщо вони використовують складні методи приховування [19].

Всі ці методи можуть бути інтегровані у вашу програму для підвищення її здатності до виявлення прихованої інформації в зображеннях. Комбінація кількох методів дозволить значно покращити ефективність виявлення і зменшити кількість хибнопозитивних або хибнонегативних результатів.

Тому саме для нашого додатку для виявлення прихованої інформаціїї було обрано StegDetect.

StegDetect є підходящим вибором для нашої програми виявлення прихованої інформації у зображеннях з інтернет-ресурсів з кількох причин. Ось детальний аналіз, чому саме цей інструмент є ідеальним для нашого завдання.

Висока точність і надійність

StegDetect спеціально розроблений для виявлення стеганографічного вмісту в JPEG-зображеннях. JPEG є одним з найпоширеніших форматів зображень в інтернеті, тому фокус StegDetect на цьому форматі робить його дуже корисним для нашої програми. Використовуючи лінійний дискримінантний аналіз (LDA) та інші методи, StegDetect забезпечує високу точність виявлення прихованої інформації, що є критичним для нашого завдання [20].

Підтримка різних методів стеганографії

StegDetect може виявляти кілька різних стеганографічних методів, включаючи jsteg, jphide, Invisible Secrets, OutGuess, F5 та інші. Ця багатофункціональність дозволяє програмі бути ефективною у виявленні різних типів прихованої інформації, що підвищує її універсальність і надійність.

Автоматизація процесу

Однією з ключових переваг StegDetect є його здатність автоматично аналізувати зображення на наявність прихованої інформації. Це означає, що наша програма може автоматично сканувати великі обсяги зображень з інтернет-ресурсів без необхідності ручної перевірки кожного зображення. Така автоматизація значно економить час і ресурси.

Легкість інтеграції

StegDetect розроблений таким чином, що його легко інтегрувати в інші програми. Це дозволяє без великих зусиль включити його в нашу програму для аналізу

зображень з інтернет-ресурсів. Використовуючи StegDetect як бібліотеку або командний рядок, ми можемо швидко адаптувати його для наших потреб.

Ефективність і продуктивність

StegDetect ефективно працює з великими обсягами даних, забезпечуючи високу продуктивність навіть при аналізі великої кількості зображень. Це важливо для нашої програми, оскільки інтернет-ресурси можуть містити тисячі або навіть мільйони зображень, які потрібно проаналізувати.

Відкритість і підтримка спільноти

StegDetect має відкритий код і підтримується спільнотою дослідників, що забезпечує постійне оновлення та вдосконалення інструмента. Це гарантує, що наша програма завжди буде використовувати найсучасніші методи виявлення стеганографічного вмісту.

StegDetect — це інструмент для виявлення прихованої інформації в JPEG-зображеннях, який використовує методи стеганографічного аналізу. Він призначений для автоматичного аналізу зображень на наявність прихованого вмісту, що робить його особливо корисним для безпеки інформації та цифрової криміналістики [21].

Основні принципи роботи StegDetect:

− Виявлення за допомогою статистичного аналізу

StegDetect використовує статистичні методи для виявлення стеганографічного вмісту в JPEG-файлах. Основна ідея полягає у виявленні відхилень у статистичних властивостях зображень, які можуть бути викликані стеганографічними методами. Стеганографія змінює характеристики зображення, і ці зміни можуть бути виявлені за допомогою спеціальних алгоритмів.

Лінійний дискримінантний аналіз (LDA)

Одним з ключових методів, що використовуються в StegDetect, є лінійний дискримінантний аналіз (LDA). LDA використовується для класифікації зображень на підставі їх статистичних властивостей. Він дозволяє відрізняти зображення з прихованим вмістом від звичайних зображень на основі аналізу ймовірнісних розподілів [22].

Підтримка різних стеганографічних методів

StegDetect може виявляти декілька різних стеганографічних методів, включаючи:

- jsteg

- jphide

- Invisible Secret

- OutGuess

- F5

Це дозволяє інструменту бути універсальним і ефективним при роботі з різними типами стеганографічних алгоритмів.

Використання сигнатур

Для виявлення прихованої інформації StegDetect використовує сигнатури стеганографічних методів. Кожен метод стеганографії залишає унікальні сліди або сигнатури в зображенні, які можуть бути виявлені за допомогою аналізу.

Процес роботи StegDetect

1.Аналіз зображення: StegDetect завантажує JPEG-зображення і аналізує його на наявність прихованого вмісту. Інструмент проводить аналіз пікселів, коефіцієнтів JPEG та інших властивостей зображення.

2.Виявлення аномалій: Використовуючи статистичний аналіз та LDA, StegDetect виявляє аномалії у властивостях зображення, які можуть вказувати на наявність стеганографічного вмісту.

3.Ідентифікація методу стеганографії: Якщо виявлено аномалії, StegDetect намагається ідентифікувати конкретний метод стеганографії, який використовувався для приховування інформації.

4.Звітування: Після аналізу StegDetect генерує звіт, який містить інформацію про виявлені аномалії та можливі методи стеганографії, які могли бути використані.

Використання StegDetect в програмі

StegDetect можна легко інтегрувати в програму для автоматичного аналізу зображень з інтернет-ресурсів. Це можна зробити через виклики командного рядка або використовуючи бібліотеки, якщо вони доступні. Після інтеграції програма зможе автоматично сканувати зображення, виявляти прихований вміст та генерувати відповідні звіти.

В цілому, StegDetect є потужним і надійним інструментом для виявлення прихованої інформації в JPEG-зображеннях, що робить його ідеальним вибором для нашої програми.

StegDetect є корисним інструментом для виявлення стеганографії, оскільки він використовує ряд просунутих методів аналізу, які роблять його ефективним і надійним. Детальний розгляд роботи StegDetect включає декілька ключових аспектів, які забезпечують його високу продуктивність у виявленні прихованої інформації.

Детальніше про методи, які використовує StegDetect

Лінійний дискримінантний аналіз (LDA)

StegDetect застосовує лінійний дискримінантний аналіз для класифікації зображень. LDA дозволяє відокремити зображення, які містять приховану

інформацію, від тих, що її не містять, базуючись на статистичних відмінностях у їхніх характеристиках. Це забезпечує високу точність і мінімізує кількість хибнопозитивних результатів.

Аналіз сигнатур

Кожен стеганографічний метод залишає унікальні сліди або сигнатури в зображенні. StegDetect порівнює ці сигнатури з базою відомих стеганографічних методів для визначення, чи містить зображення приховану інформацію, і який метод був використаний [23].

Використання кількох методів виявлення

StegDetect може ідентифікувати кілька стеганографічних методів, таких як jsteg, jphide, Invisible Secrets, OutGuess та F5. Це дозволяє інструменту бути універсальним і забезпечувати високу точність у різних сценаріях.

Інтеграція StegDetect у програму:

Інтерфейс командного рядка

StegDetect можна використовувати через інтерфейс командного рядка, що дозволяє легко інтегрувати його у вашу програму. Це дає змогу автоматично аналізувати великі обсяги зображень без необхідності ручного втручання.

Автоматизація процесу

Програму можна налаштувати для автоматичного сканування зображень з інтернет-ресурсів. Це забезпечує безперервний моніторинг та виявлення прихованої інформації, що є важливим для забезпечення безпеки даних.

Переваги StegDetect

1.Точність і надійність: Завдяки використанню LDA та аналізу сигнатур, StegDetect забезпечує високу точність виявлення прихованої інформації.

2.Універсальність: Підтримка різних стеганографічних методів робить його корисним для широкого спектра застосувань.

3.Автоматизація: Можливість автоматичного аналізу зображень значно економить час і ресурси.

4.Легкість інтеграції: Використання інтерфейсу командного рядка дозволяє легко інтегрувати StegDetect у різні програмні рішення.

Таким чином, StegDetect є потужним інструментом для виявлення стеганографічної інформації у JPEG-зображеннях. Його точність, універсальність і можливість автоматизації роблять його ідеальним вибором для нашої програми, що аналізує зображення з інтернет-ресурсів на наявність прихованої інформації [24].

2.2 Розробка структури та функціоналку програми

Основною метою даної розробки, являється можливість користувачам отримувати дані про те чи знаходяться в картинках приховані русерси та для їхньої перевірки.

Програма для виявлення прихованої інформації у зображеннях інтернет-ресурсів має складну структуру, яка включає кілька ключових компонентів для ефективного функціонування.

Основні частини програми включають:

− Модуль збору даних: Цей модуль відповідає за збір зображень з інтернет-ресурсів. Він використовується для парсингу веб-сторінок і соціальних мереж для автоматичного завантаження зображень у великій кількості.

−Попередня обробка зображень: Призначений для форматування і нормалізації зображень перед аналізом. Цей модуль може включати конвертацію зображень у стандартні формати, зміну розмірів та нормалізацію піксельних значень.

−Модуль аналізу зображень: Основний компонент програми, який використовує різноманітні методи для виявлення стеганографічних даних. Включає аналіз

просторової області (LSB алгоритми) і частотної області (DCT, DWT), а також застосування методів машинного навчання для покращення точності аналізу [25].

− Модуль візуалізації та звітності: Відповідає за представлення результатів аналізу користувачеві. Цей модуль генерує детальні звіти і візуалізації, що допомагають користувачеві легко ідентифікувати підозрілі зображення та аномалії.

− Модуль безпеки та конфіденційності: Включає заходи для захисту персональних даних користувачів та збереження конфіденційності під час обробки зображень інтернет-ресурсів.

Ця структура дозволяє програмі ефективно виконувати завдання з виявлення стеганографічної інформації, забезпечуючи високу точність та безпеку при аналізі зображень, що поширюються через мережу.

Виходячи з поданої структури, користувачу достатньо запустити програму та вставити у поле аналізу посилання на сайт або окрему картинку в браузері. Програма для сканування зображень в інтернет-ресурсах на наявність прихованої інформації має за мету забезпечити повноцінне виявлення стеганографічних даних, які можуть бути приховані у зображеннях.

Цей інструмент використовується для автоматизованого збору зображень з веб-сайтів, їх попередньої обробки та аналізу з метою виявлення незаконної або прихованої інформації.

Основні функції програми включають збір зображень з різних джерел інтернету, таких як веб-сторінки і соціальні мережі, а також їхню передобробку, включаючи зміну форматів і масштабування для подальшого аналізу. Сам аналіз зображень проводиться з використанням різноманітних методів, включаючи аналіз просторової області (наприклад, LSB алгоритми) та частотної області (за допомогою DCT або DWT) [26].

Після завершення аналізу програма надає користувачеві звіти з результатами, що містять інформацію про виявлені аномалії чи підозрілі зображення, а також можливості для подальшого детального аналізу та обробки.

Таким чином, програма для виявлення прихованої інформації у зображеннях інтернет-ресурсів представляє собою потужний інструмент для забезпечення кібербезпеки та виявлення потенційних загроз у мережі, тому розглянемо кроки, етапи які проходить зображення перед тим як ми отримаємо результат.

Загальний функціонал програми описується таким чином:

- Отримання URL зображення отримане з введеної веб сторінки.
- Завантаження зображення за цими URL.
- Аналіз зображення на наявність прихованих файлів та тексту.
- Відображення результатів аналізу у вигляді списку URL, стану прихованих даних і прихованого тексту у вікні програми.

Далі будуть охарактеризовані наступні кроки які виконує програма:

Крок 1 Ініціалізація аналізу:

Крок 1.1 Після натискання кнопки 'button 1' , текстове поле, список та текстова область очищуються, і встановлюється текст мітки 'label 5' на Початок аналізу

Крок 1.2 Програма отримує URL з тестового поля 'textBox 1''

Крок 2 Отримання URL зображень:

Крок 2.1 Викликається асинхронний метод ''GetImageUrlsFromWebsite', який повертає список URL зображень з введеного веб-сайту.

Крок 2.2 Ці URL додаються до списку ''sitBox1'.

Крок 3 Завантаження та аналіз зображень:

Крок 3.1 Завантажується зображення за допомогою асинхронного методу ''DownloadIamge'.

Крок 3.2 Якщо зображення успішно завантажене (''bitmap != null').

Крок 3.3 Відображається зображення у компоненті 'pictureBox1'.

Крок 3.4 Виконується аналіз зображення на наявність прихованих файлів за допомогою методу ''AnalyzeImageForHiddenFiles'.

Крок 3.6 Виконується витягування прихованого тексту за допомгою методу ''ExtractHiddenText'.

Крок 3.7 Результати аналізу (URL зображення, наявність прихваоних даних, прихований текст) додаються до текстової області 'richTextBox1'.

Крок 3.8 Мітка 'label5'. оновлюється, вказуючи на поточне аналізоване зображення.

Крок 4 Завершення аналізу:

Крок 4.1 Після завершення аналізу всіх зображень, мітка 'lable5' оновлюється на 'Аналіз завержено'.

Загалом усі функції і процеси які відбуваються під час роботи програми, приховані від користувача

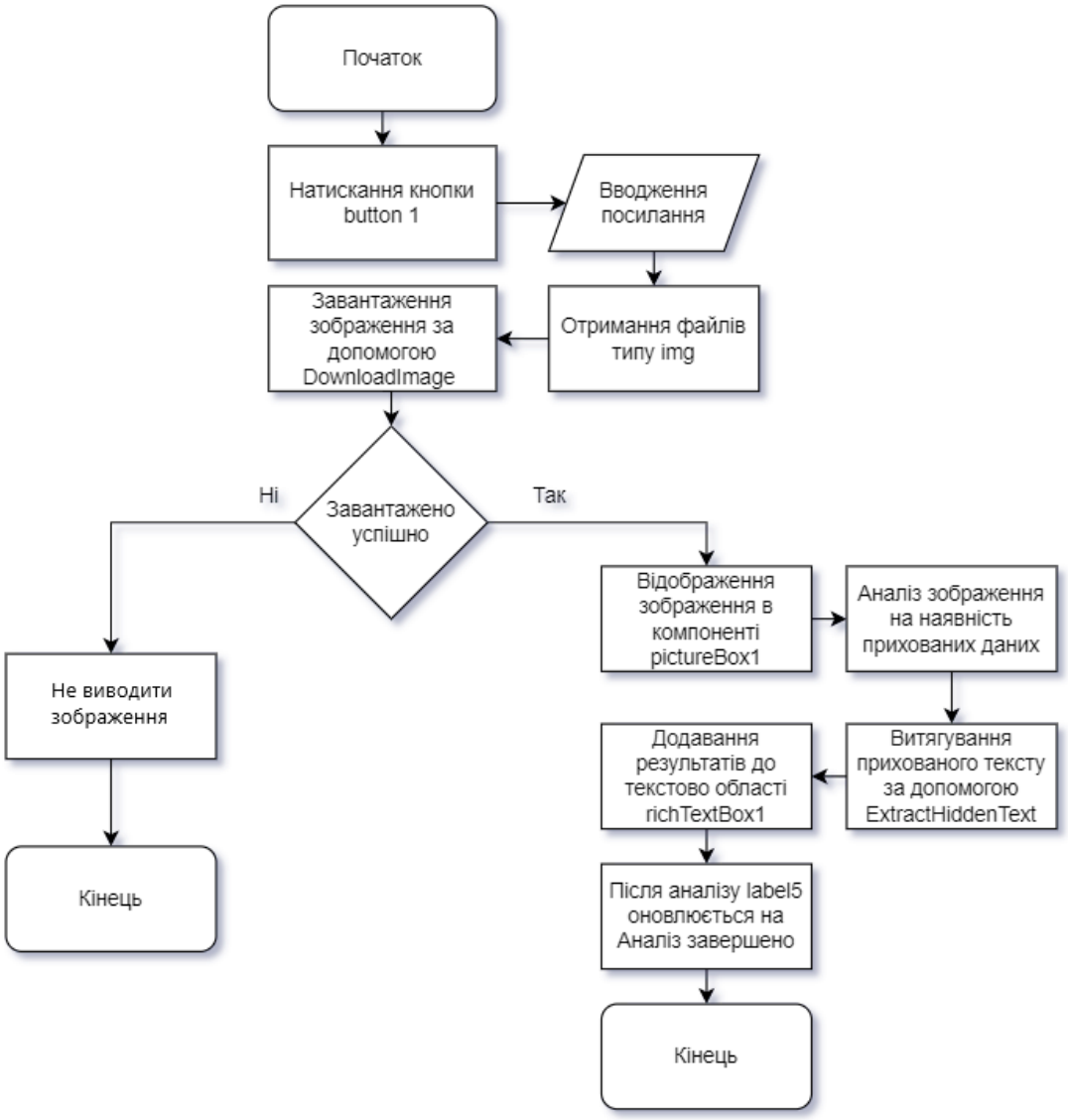


Рисунок 2.1 – Алгоритм роботи розробленого додатку

2.3 Розробка графічного інтерфейсу програмної розробки

Розробка графічного інтерфейсу програмної розробки має вирішальне значення для забезпечення зручності та ефективності взаємодії користувача з програмою. Графічний інтерфейс користувача (GUI) надає візуальні елементи, такі як кнопки, меню, форми та інші компоненти, які дозволяють користувачам легко взаємодіяти з програмою без потреби вивчати складні команди або синтаксис [27].

По-перше, графічний інтерфейс спрощує використання програми, роблячи її доступнішою для користувачів з різним рівнем технічних знань. Це особливо важливо для програм, які призначені для широкого кола користувачів або виконують складні функції. Зрозумілий і інтуїтивно зрозумілий інтерфейс дозволяє користувачам швидко освоїти програму і ефективно її використовувати.

По-друге, добре розроблений графічний інтерфейс підвищує продуктивність роботи користувача. Зручні елементи керування та організація інтерфейсу дозволяють користувачам виконувати свої завдання швидше і з меншими зусиллями. Це зменшує кількість помилок і покращує загальний досвід використання програми.

Розробка графічного інтерфейсу також включає в себе створення привабливого дизайну, який сприяє позитивному сприйняттю програми. Зовнішній вигляд і зручність використання інтерфейсу можуть значно вплинути на загальне враження користувача від програми, що є важливим фактором для її успішного впровадження і використання.

Таким чином, розробка графічного інтерфейсу програмної розробки є ключовим етапом, який забезпечує зручність, ефективність і привабливість програми для користувачів, сприяючи її успішному впровадженню та використанню.

Графічний інтерфейс не лише полегшує взаємодію користувача з програмою, але й забезпечує її функціональність на вищому рівні. Він дозволяє структурувати інформацію та функції таким чином, щоб вони були логічно організовані і легко

доступні. Це особливо важливо для складних програм, які містять велику кількість функцій і налаштувань, таких як програми для виявлення прихованої інформації у зображеннях.

Завдяки графічному інтерфейсу, розробники можуть створювати інтерактивні елементи, які дозволяють користувачам виконувати різноманітні дії, наприклад, завантажувати зображення для аналізу, налаштовувати параметри пошуку прихованої інформації, переглядати результати та отримувати детальну інформацію про знайдені приховані дані. Інтерфейс дозволяє зробити всі ці дії інтуїтивно зрозумілими і простими у виконанні.

Крім того, графічний інтерфейс забезпечує можливість візуалізації даних. Для програми, що аналізує зображення, важливо надавати результати у зручному для сприйняття форматі. Наприклад, можна відображати зображення до і після аналізу, виділяти області з прихованою інформацією, надавати графічні звіти і діаграми, які пояснюють результати аналізу. Це дозволяє користувачам швидко і легко розуміти результати роботи програми.

Нарешті, графічний інтерфейс є важливим інструментом для забезпечення зворотного зв'язку від користувачів. Інтерактивні елементи інтерфейсу можуть бути налаштовані для збору відгуків і пропозицій від користувачів, що дозволяє розробникам постійно покращувати програму на основі реальних потреб і побажань користувачів.

Отже, розробка графічного інтерфейсу програмної розробки - це не лише створення привабливого дизайну, але й забезпечення зручності, функціональності, адаптивності і можливості візуалізації даних, що робить програму більш ефективною і привабливою для користувачів.

Крім забезпечення зручності використання, графічний інтерфейс допомагає створювати більш інтуїтивно зрозумілий користувацький досвід, що є важливим

фактором для успішного впровадження програми на ринок. Інтуїтивний інтерфейс дозволяє знизити криву навчання для нових користувачів, зменшуючи час, необхідний для освоєння основних функцій програми. Це особливо актуально для програм, які орієнтовані на широкий спектр користувачів, включаючи тих, хто не має глибоких технічних знань.

У процесі розробки графічного інтерфейсу важливо дотримуватися принципів ергономічного дизайну, що включають у себе оптимальне розташування елементів керування, використання зрозумілих іконок, а також забезпечення консистентності у всьому інтерфейсі. Це сприяє покращенню користувацького досвіду і зменшує ймовірність помилок під час використання програми.

Забезпечення зручності для користувачів також включає підтримку локалізації та доступності. Програма повинна мати можливість працювати на різних мовах, відповідно до потреб користувачів у різних регіонах світу. Крім того, важливо враховувати потреби користувачів з обмеженими можливостями, забезпечуючи підтримку технологій для читання з екрану, висококонтрастного режиму та інших засобів, що підвищують доступність програми.

Враховуючи всі ці аспекти, можна зробити висновок, що розробка графічного інтерфейсу є критично важливою складовою успішного проекту. Вона забезпечує не лише зручність і функціональність, але й сприяє безпеці, доступності і продуктивності програми, що в кінцевому підсумку покращує досвід користувачів і сприяє успішному впровадженню і використанню програмного продукту.

Загальний вигляд програми вміщує в себе абюсолютно всі потрібні деталі для легкого користування, має не складний у розумінні інтерфейс.

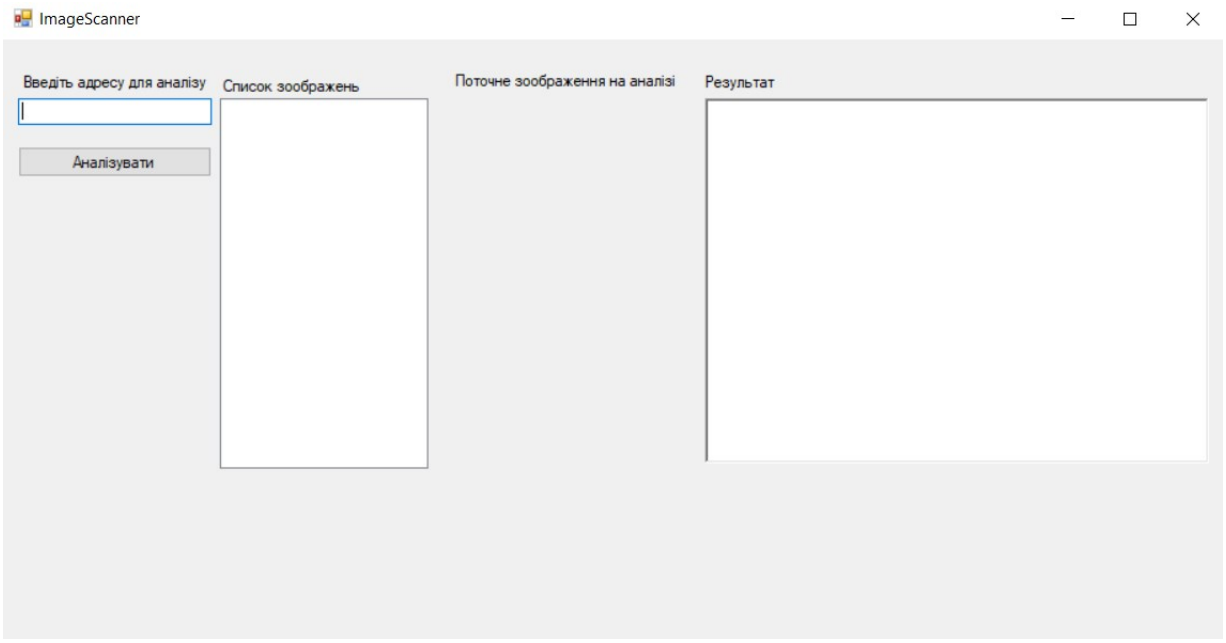


Рисунок 3.1 – Загальний вигляд програми

Як видно на рис. 3.1, компоненти гармонійно розташовані та не заважають один одному.

2.4 Висновки до розділу

В даному розділі були розглянуті існуючі методи виявлення прихованої інформації, розроблено алгоритм витягу зображення з інтернет-ресурсу та подальший його аналіз, також створений графічний інтерфейс для зручного користування.

3.ПРАКТИЧНА РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ РОЗРОБЛЕНОЇ ПРОГРАМИ

В даному розділі буде описано основні етапи практичної реалізації та тестування розробленої програми для виявлення прихованої інформації в інтернет ресурсах. Зокрема, спроектовано та описано розробку графічного користувацького інтерфейсу, програмну реалізацію розробленої програми, наведено інструкцію користувача для роботи з додатком, проведено тестування розробленого програмного продукту.

3.1 Вибір програмного застосунку та мови програмування

Виходячи із поставленої мети та задач роботи, для розробки програми виявлення прихованої інформації у зображеннях інтернет-ресурсів було обрано такі засоби програмування:

Мова програмування C# для розробки програми виявлення прихованої інформації у зображеннях інтернет-ресурсів, C# є відмінним варіантом з кількох причин:

- Широкі можливості для розробки Windows-орієнтованих додатків: C# є основною мовою програмування для платформи .NET, що значно спрощує розробку десктопних додатків для операційних систем Windows. Оскільки багато користувачів Інтернету використовують Windows, C# дозволяє легко інтегрувати програму в їхнє середовище.

- Можливості для розробки веб-додатків: C# також використовується для розробки веб-додатків з використанням ASP.NET та інших веб-технологій, що може бути корисним для інтеграції програми з інтернет-ресурсами [28].

- Широкі бібліотеки і фреймворки: Екосистема .NET має величезну кількість сторонніх бібліотек і фреймворків, які полегшують розробку і підтримку програмного забезпечення.

− Простота використання та сучасність: C# є сучасною мовою програмування зі зручним синтаксисом та широкими можливостями підтримки об'єктно-орієнтованого програмування, асинхронних операцій та інших сучасних парадигм.

− Інтеграція з іншими технологіями Microsoft: Якщо програма має інтегруватись з іншими продуктами Microsoft, такими як SQL Server, Azure або Office, то використання C# дозволяє з легкістю інтегрувати ці сервіси.

Обираючи C# для розробки програми виявлення прихованої інформації у зображеннях, ми отримуємо мову програмування з потужними інструментами, що дозволяють ефективно і швидко реалізувати функціональність програми, забезпечуючи при цьому високий рівень надійності та підтримки [29].

Обираючи мову програмування C# для розробки програми виявлення прихованої інформації у зображеннях інтернет-ресурсів, важливо врахувати кілька ключових аспектів.

C# є частиною платформи .NET, що забезпечує широкі можливості для розробки різноманітних застосунків, включаючи десктопні програми для Windows, веб-додатки, служби, мобільні додатки та інші типи програм. Його синтаксис і особливості спрощують роботу з об'єктно-орієнтованим програмуванням, що є важливим для структурування складних програмних рішень.

Окрім того, C# має потужну інтегровану середу розробки (IDE) - Visual Studio, яка надає розширені можливості для розробки, налагодження і підтримки програм. Visual Studio підтримує широкий набір інструментів для автоматизації рутинних завдань, включаючи рефакторинг коду, управління версіями, інтеграцію з системами контролю версій і тестуванням.

Ще однією перевагою C# є його підтримка асинхронних операцій, що дозволяє програмам ефективно взаємодіяти з мережами та іншими віддаленими ресурсами без

блокування головного потоку виконання. Це особливо важливо для програм, які обробляють великі обсяги даних, такі як зображення у великих інтернет-ресурсах.

Крім того, C# має активну спільноту розробників і велику кількість відкритих бібліотек, які допомагають з різними завданнями, включаючи обробку зображень і аналіз даних. Це робить мову привабливою для розробки програм з високим рівнем функціональності і надійності.

Вибір C# для розробки програми виявлення прихованої інформації у зображеннях інтернет-ресурсів забезпечує не тільки потужність і ефективність, але й зручність у роботі з іншими технологіями Microsoft і інтеграцію з різними платформами.

Крім перелічених переваг, використання C# для розробки програми виявлення прихованої інформації в інтернет-ресурсах також дозволяє використовувати велику кількість інструментів і бібліотек для роботи з зображеннями і обробки даних. Мова програмування має багато готових рішень для обробки і аналізу графічного матеріалу, що спрощує розробку алгоритмів виявлення прихованої інформації.

Також, екосистема .NET забезпечує можливість використання різноманітних сервісів і інтеграцію з іншими технологіями Microsoft, такими як Azure для обробки великих обсягів даних чи SQL Server для зберігання інформації. Це важливо для програм, які потребують підтримки складних операцій з базами даних або обчислювальних ресурсів у хмарі [30].

Однією з ключових переваг є також широке використання C# в промисловості та корпоративних середовищах. Багато підприємств використовують .NET для розробки своїх продуктів і служб, тому вибір C# дозволяє з легкістю інтегрувати нові рішення з існуючими інфраструктурами та сервісами.

Загалом, C# володіє потужним інструментарієм для розробки програм з високим рівнем функціональності та безпеки. Використання цієї мови програмування дозволить ефективно реалізувати програму для виявлення прихованої інформації в

інтернет-ресурсах, забезпечуючи високу швидкодію та масштабованість для різних потреб користувачів.

Visual studio f Обираючи Visual Studio для розробки програми виявлення прихованої інформації у зображеннях інтернет-ресурсів, можна виділити декілька ключових переваг:

− Інтегрована середа розробки (IDE): Visual Studio є потужною інтегрованою середою розробки, яка підтримує весь життєвий цикл розробки програмного забезпечення. Вона включає у себе різноманітні інструменти для написання коду, налагодження, тестування, відлагодження помилок, управління версіями і розгортання програм [31].

− Підтримка мов програмування: Visual Studio підтримує багато мов програмування, включаючи C#, що була обрана для розробки програми. Це дозволяє розробникам використовувати мови з високим рівнем абстракції для створення складних програмних рішень.

− Інтеграція зі сторонніми сервісами і інструментами: Visual Studio має широку підтримку сторонніх бібліотек, фреймворків і сервісів, що спрощує інтеграцію з різноманітними технологіями. Наприклад, вона має вбудовану підтримку Azure для хмарних рішень, що може бути корисним для розробки програм, які потребують зберігання та обробки великих обсягів даних.

− Засоби для роботи з графічним інтерфейсом користувача: Visual Studio має інтегровані засоби для розробки графічного інтерфейсу користувача (GUI) для десктопних і веб-додатків. Це включає у себе дизайнери форм, які дозволяють швидко створювати і налаштовувати інтерфейс користувача [32].

− Спільнота та підтримка: Visual Studio має активну спільноту розробників і велику кількість документації, уроків і ресурсів для навчання. Це дозволяє швидко знаходити відповіді на питання та розв'язувати проблеми під час розробки.

Отже, обираючи Visual Studio для розробки програми виявлення прихованої інформації, ми отримуємо потужний інструмент з усіма необхідними функціями для ефективної і продуктивної розробки, що сприяє якісному результату і зменшує час розробки [33].

3.2 Програмна реалізація програми для виявлення прихованої інформації у зображеннях інтернет-ресурсів.

Програмна реалізація програми для виявлення прихованої інформації в інтернет-ресурсах передбачає декілька ключових етапів, які включають в себе збір зображень, їх аналіз на наявність прихованої інформації, а також зручний і зрозумілий інтерфейс для користувачів.

Першим етапом є збір зображень з інтернет-ресурсів. Для цього програма використовує веб-скрапінг, тобто автоматичне збирання даних з веб-сторінок. Цей процес реалізується за допомогою бібліотек для веб-скрапінгу, таких як HtmlAgilityPack, яка дозволяє завантажувати та парсити HTML-код веб-сторінок для витягування зображень. Програма може бути налаштована на сканування певних веб-ресурсів або пошукових запитів для автоматичного збору зображень.

Після збору зображень програма переходить до етапу аналізу. Тут важливу роль відіграють алгоритми стеганографічного аналізу, такі як StegDetect. StegDetect є інструментом для виявлення прихованих даних в зображеннях, особливо тих, які використовують методи стеганографії, такі як JSteg, OutGuess, F5 та інші. Він аналізує структуру зображення і шукає підозрілі патерни, які можуть вказувати на наявність прихованої інформації.

Для інтеграції StegDetect у програму, необхідно забезпечити взаємодію між компонентами збору зображень та аналізу. Це може бути реалізовано через виклик зовнішніх інструментів командного рядка з C# коду або через використання відповідних бібліотек, якщо такі існують для обраного інструменту. Програма

повинна обробляти результати аналізу і зберігати їх у відповідному форматі для подальшої обробки або відображення користувачам.

Наступний важливий компонент програми - це графічний інтерфейс користувача (GUI). Він повинен бути інтуїтивно зрозумілим і зручним у використанні. Інтерфейс надає користувачам можливість переглядати результати аналізу і отримувати детальну інформацію про знайдені приховані дані.

Таким чином, програмна реалізація програми для виявлення прихованої інформації в інтернет-ресурсах включає в себе збір зображень, їх аналіз за допомогою стеганографічних інструментів, розробку зручного графічного інтерфейсу, забезпечення безпеки даних.

Тому розглянемо кожен аспект програми окремо, розпочнемо з основного поля в яке користувач має ввести посилання інтернет ресурсу для його подальшого аналізу та виявлення прихованої інформації [34].

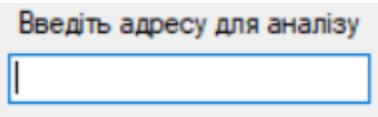


Рисунок 3.2 – Поле вводу адреси інтернет ресурсу

```
this.label1.AutoSize = true;
 this.label1.Location = new System.Drawing.Point(13, 25);
 this.label1.Name = "label1";
 this.label1.Size = new System.Drawing.Size(144, 13);
 this.label1.TabIndex = 1;
 this.label1.Text = "Введіть адресу для аналізу";
```

Наступна деталь графічного інтерфейсу виступає кнопка Аналізувати, функції якої заключається в тому що, потрібно на неї натиснути, щоб почався процес аналізу зображення [35].

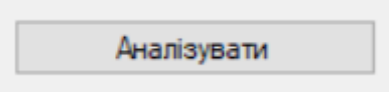


Рисунок 3.3 – Зовнішній вигляд кнопки для запуску аналізу

```
private async void button1_Click(object sender, EventArgs e)
        {
            string url = textBox1.Text;
            listBox1.Items.Clear();
            richTextBox1.Clear();
            label5.Text = "Початок аналізу...";
            List<string> imageUrls = await GetImageUrlsFromWebsite(url);

            foreach (var imageUrl in imageUrls)
            {
                listBox1.Items.Add(imageUrl);
                Bitmap bitmap = await DownloadImage(imageUrl);
                if (bitmap != null)
                {
                    pictureBox1.Image = bitmap;
                    bool hiddenData = AnalyzeImageForHiddenFiles(bitmap);
                    string hiddenText = ExtractHiddenText(bitmap);
                    richTextBox1.AppendText($"Image URL: {imageUrl}\nHidden data: {hiddenData}\nHidden text:
{hiddenText}\n");
                    label5.Text = $"Аналіз зображення: {imageUrl}";
                }
            }

            label5.Text = "Аналіз завершено!";
        }      private async void button1_Click(object sender, EventArgs e)
        {
            string url = textBox1.Text;
            listBox1.Items.Clear();
            richTextBox1.Clear();
            label5.Text = "Початок аналізу...";
            List<string> imageUrls = await GetImageUrlsFromWebsite(url);

            foreach (var imageUrl in imageUrls)
            {
                listBox1.Items.Add(imageUrl);
                Bitmap bitmap = await DownloadImage(imageUrl);
                if (bitmap != null)
                {
                    pictureBox1.Image = bitmap;
                    bool hiddenData = AnalyzeImageForHiddenFiles(bitmap);
                    string hiddenText = ExtractHiddenText(bitmap);
                    richTextBox1.AppendText($"Image URL: {imageUrl}\nHidden data: {hiddenData}\nHidden text:
{hiddenText}\n");
                    label5.Text = $"Аналіз зображення: {imageUrl}";
                }
            }

            label5.Text = "Аналіз завершено!";
        }
```

Далі буде показано вікно, в якому знаходяться усі картинки які пропускає через себе програма.

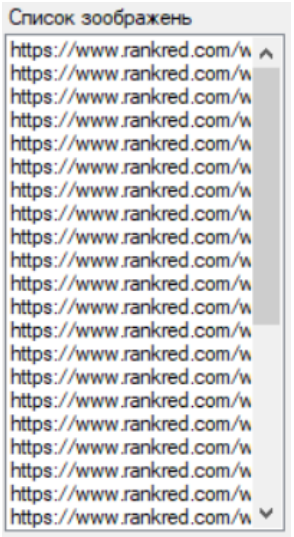


Рисунок 3.4 – Перелік зображень

```
private async Task<List<string>> GetImageUrlsFromWebsite(string url)
{
    List<string> imageUrls = new List<string>();
    using (HttpClient client = new HttpClient())
    {
        string html = await client.GetStringAsync(url);
        HtmlAgilityPack.HtmlDocument doc = new HtmlAgilityPack.HtmlDocument();
        doc.LoadHtml(html);
        foreach (var node in doc.DocumentNode.SelectNodes("//img[@src]"))
        {
            string src = node.GetAttributeValue("src", null);
            if (!string.IsNullOrEmpty(src))
            {
                imageUrls.Add(src);
            }
        }
    }
    return imageUrls;
}

private async Task<Bitmap> DownloadImage(string imageUrl)
{
    using (HttpClient client = new HttpClient())
    {
        try
        {
            byte[] imageBytes = await client.GetByteArrayAsync(imageUrl);
            using (MemoryStream ms = new MemoryStream(imageBytes))
            {
                return new Bitmap(ms);
            }
        }
        catch
        {
```

```
            return null;
        }
    }
}
```

Наступним компонентом є ділянка в якій показується зображення яке сканується в даний момент.

Рисунок 3.5 – Вигляд вікна у якому аналізується поточне зображення

```
private bool CheckForHiddenFiles(byte[] imageData)
{
    string[] fileSignatures = { "504B0304", "52617221", "25504446" }; // Zip, Rar, PDF
    string hexString = BitConverter.ToString(imageData).Replace("-", "");

    foreach (string signature in fileSignatures)
    {
        if (hexString.Contains(signature))
        {
            return true;
        }
    }

    return false;
}

private bool AnalyzeImageForHiddenFiles(Bitmap bitmap)
{
    byte[] imageData = ImageToByteArray(bitmap);
    return CheckForHiddenFiles(imageData);
```

```
}

private byte[] ImageToByteArray(Bitmap bitmap)
{
   using (MemoryStream ms = new MemoryStream())
   {
      // Ensure the bitmap is in a compatible format
      Bitmap clone = new Bitmap(bitmap);
      clone.Save(ms, System.Drawing.Imaging.ImageFormat.Png);
      return ms.ToArray();
   }
                                                                  }
```

Остання деталь даного графічного інтерфейсу є поле в якому показується результат аналізування прихованої інформації в зображеннях, та якщо в них знаходиться дана інформація вона виводиться текст, або посиланням в результаті.

Рисунок 3.6 – Вікно завершення аналізу

```
   private string ExtractHiddenText(Bitmap bitmap)
   {
      List<byte> extractedBits = new List<byte>();
      for (int y = 0; y < bitmap.Height; y++)
      {
         for (int x = 0; x < bitmap.Width; x++)
         {
            Color pixel = bitmap.GetPixel(x, y);
            extractedBits.Add((byte)(pixel.R & 1)); // Extract LSB from Red channel
            extractedBits.Add((byte)(pixel.G & 1)); // Extract LSB from Green channel
            extractedBits.Add((byte)(pixel.B & 1)); // Extract LSB from Blue channel
         }
```

```
        }

        List<byte> byteList = new List<byte>();
        for (int i = 0; i < extractedBits.Count; i += 8)
        {
            byte b = 0;
            for (int bitIndex = 0; bitIndex < 8; bitIndex++)
            {
                if (i + bitIndex < extractedBits.Count)
                {
                    b = (byte)(b | (extractedBits[i + bitIndex] << (7 - bitIndex)));
                }
            }
            byteList.Add(b);
        }

        string hiddenText = Encoding.UTF8.GetString(byteList.ToArray());

        int nullIndex = hiddenText.IndexOf('\0');
        if (nullIndex >= 0)
        {
            hiddenText = hiddenText.Substring(0, nullIndex);
        }

        return hiddenText;
    }
  }
}
```

3.3 Інструкція користувача для роботи з програмою

Розглянемо інструкцію для користування даною програмою. Першим пунктом буде шлях до .exe файлу для запуску додатку.

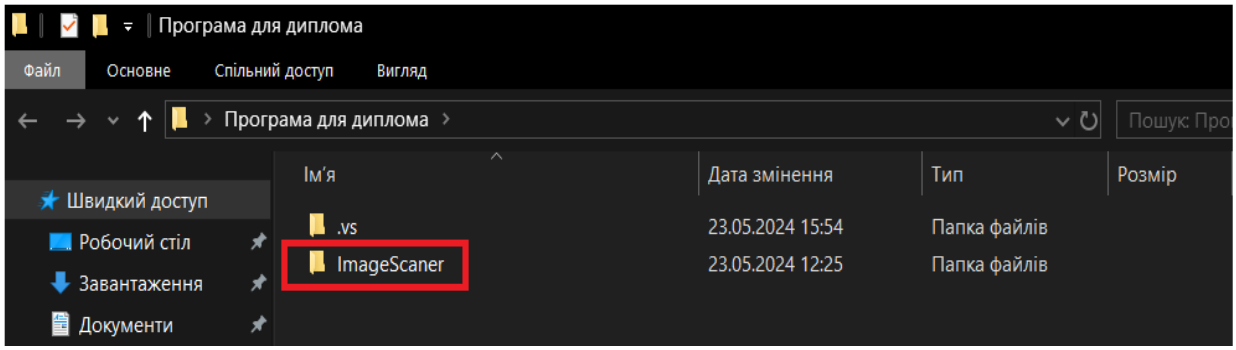


Рисунок 3.7 – Початкова папка програми

Далі рухаємось по наступни папкам.

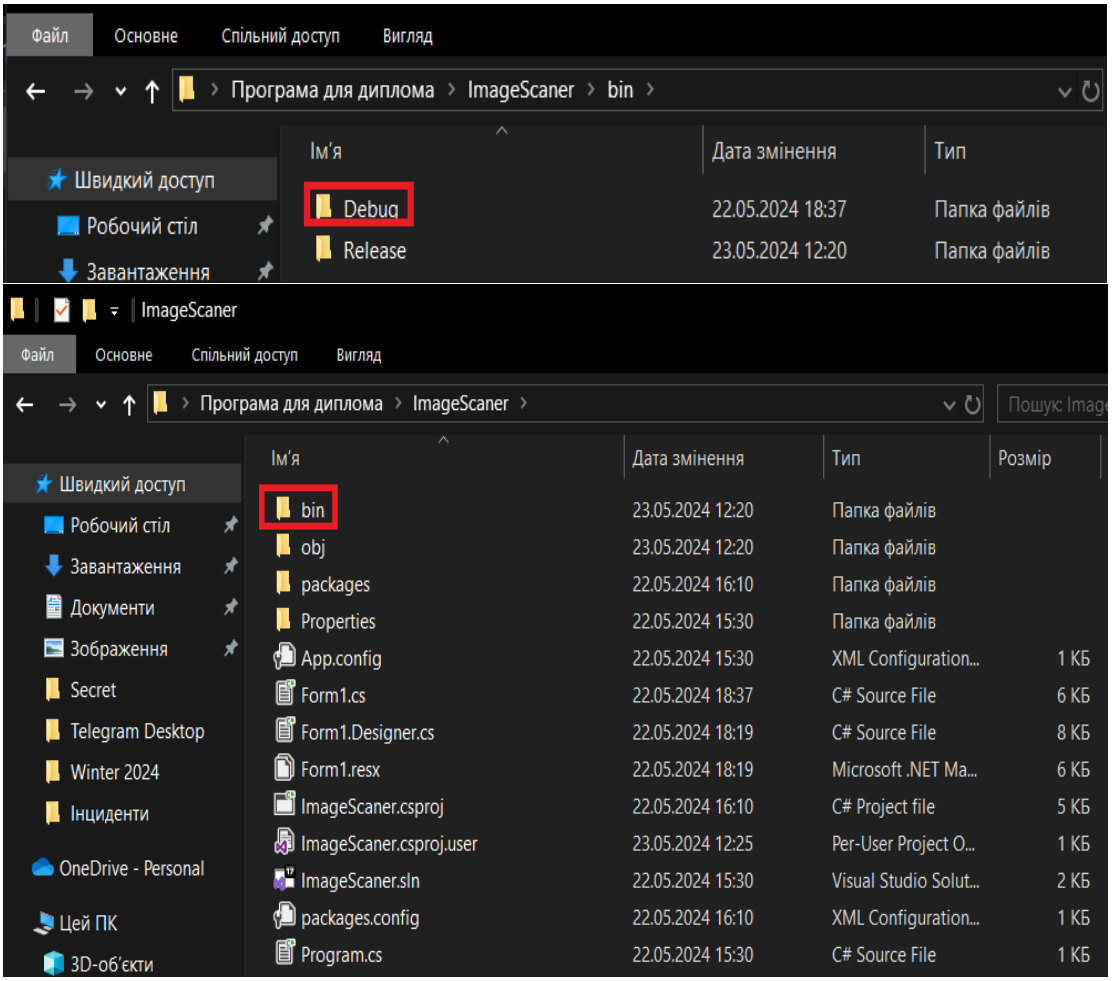


Рисунок 3.8 – Місцезнаходження програми

Останній крок це запуск .exe файлу

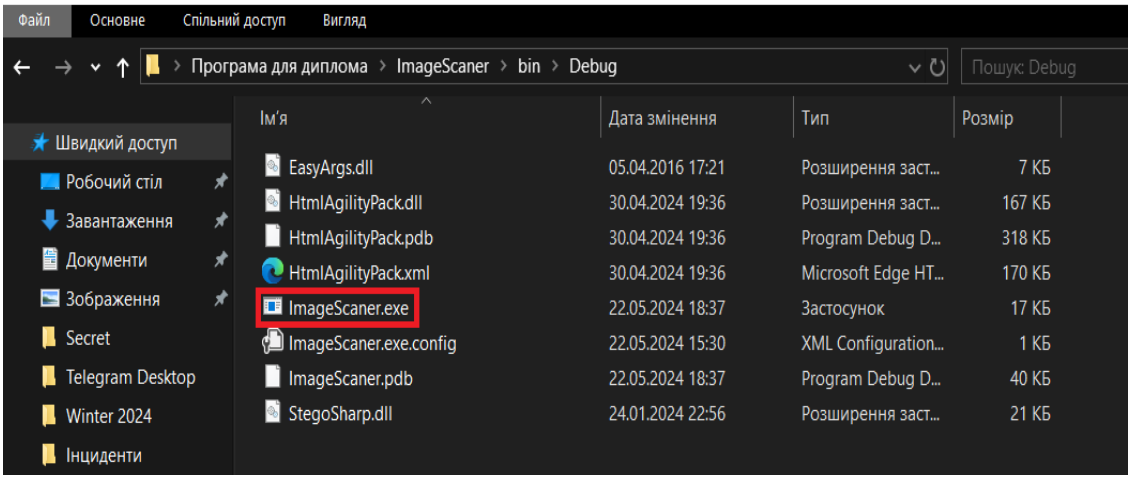


Рисунок 3.9 – Вигляд файлу запуску застосунку

Після цього побачимо основне вікно програми в якій можна проводити безпосередньо аналіз потрібного нам інтернет ресурсу.

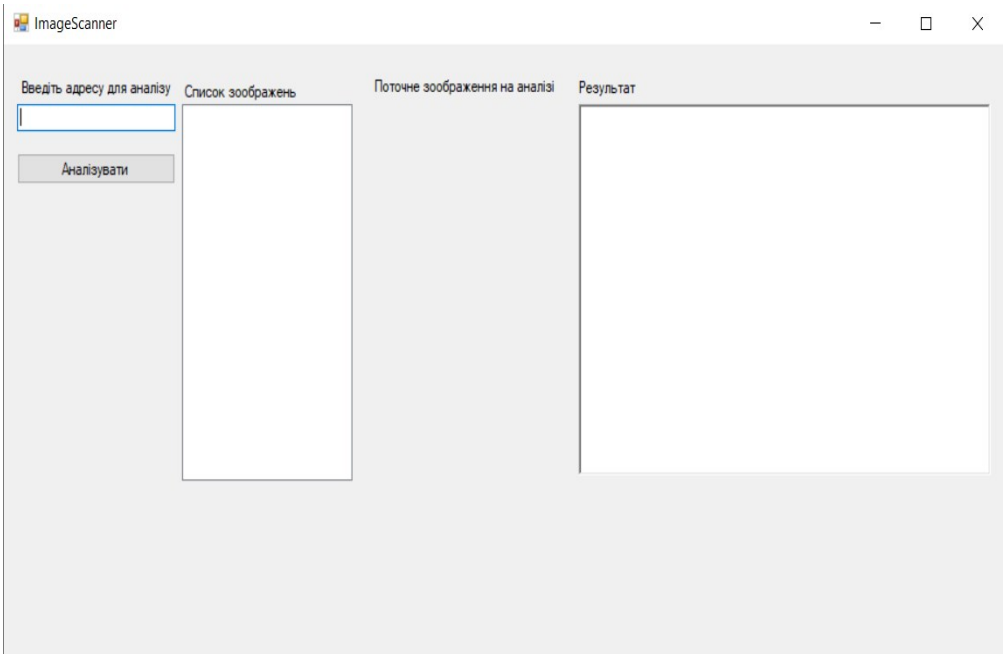


Рисунок 3.10 – Головне вікно програми

3.4 Тестування розробленої програми

Тестування програми є важливим етапом для забезпечення її надійності та правильного функціонування. Воно включає декілька видів тестування, таких як функціональне тестування, перегляду та експорту результатів, візуалізації та продуктивності. Нижче наведено кроки, які слід виконати для проведення комплексного тестування програми.

Функціональне тестування:

− Переконатися, що всі функції програми працюють належним чином, тобто збір зображень:

− Введемо різні URL-адреси веб-ресурсів і пошукові запити для сканування.

− Перевіримо, чи збирає програма зображення з вказаних ресурсів.

− Переконайтеся, що зібрані зображення відображаються у списку в головному вікні.

Візуалізація даних:

– Перевіримо, чи коректно відображаються зібрані зображення та результати аналізу.

Тестування продуктивності:

– Оцінимо швидкість та ефективність роботи програми під різними навантаженнями, тобто час збору зображень.

– Запустимо збір зображень з великих веб-ресурсів і заміряєм час виконання.

– Переконаємось, що програма не зависає і коректно завершує процес.

Таким чином, комплексне тестування забезпечить високу якість та надійність програми для виявлення прихованої інформації в інтернет-ресурсах, що, у свою чергу, сприятиме ефективності та зручності її використання.

Функціональне тестування

В даному тестуванні ми проведемо декілька етапів стресу для програми. Першим етапом буде перевірка чи коректно працює збір зображень, візьмемо декілька з аналізу нашої програми та перевіримо їх через консольні рядки коду в браузері.

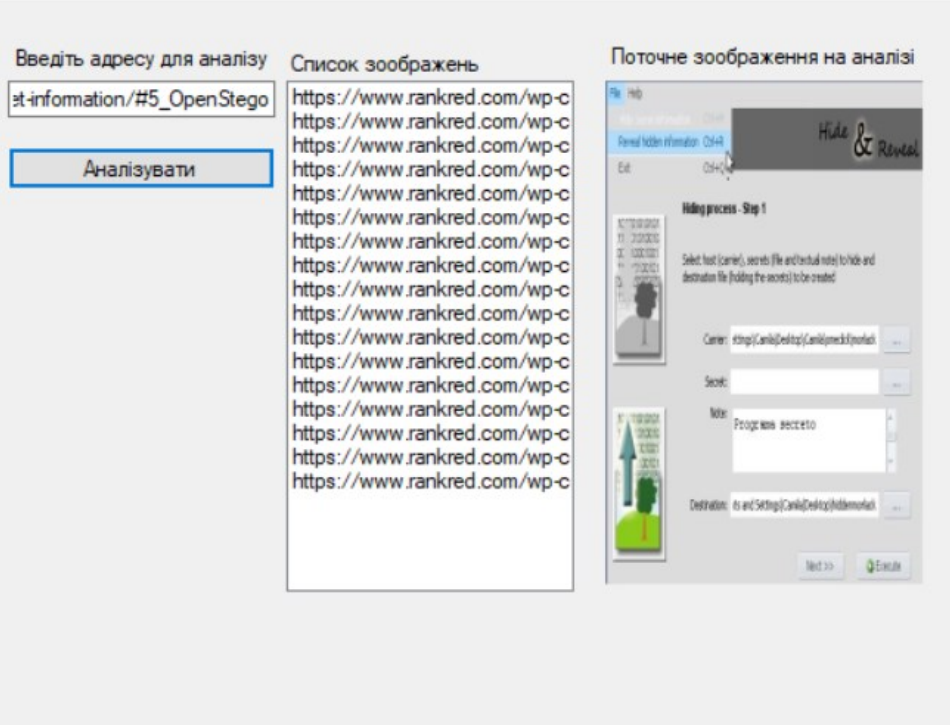


Рисунок 3.11 – Перший приклад зображення з програми

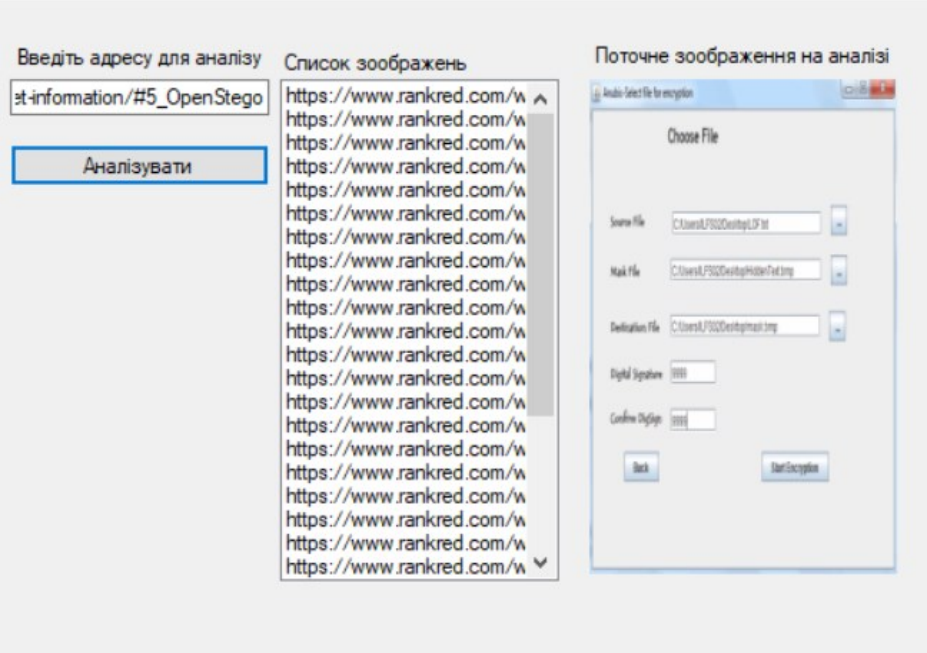


Рисунок 3.12 – Другий приклад зображення з додатку

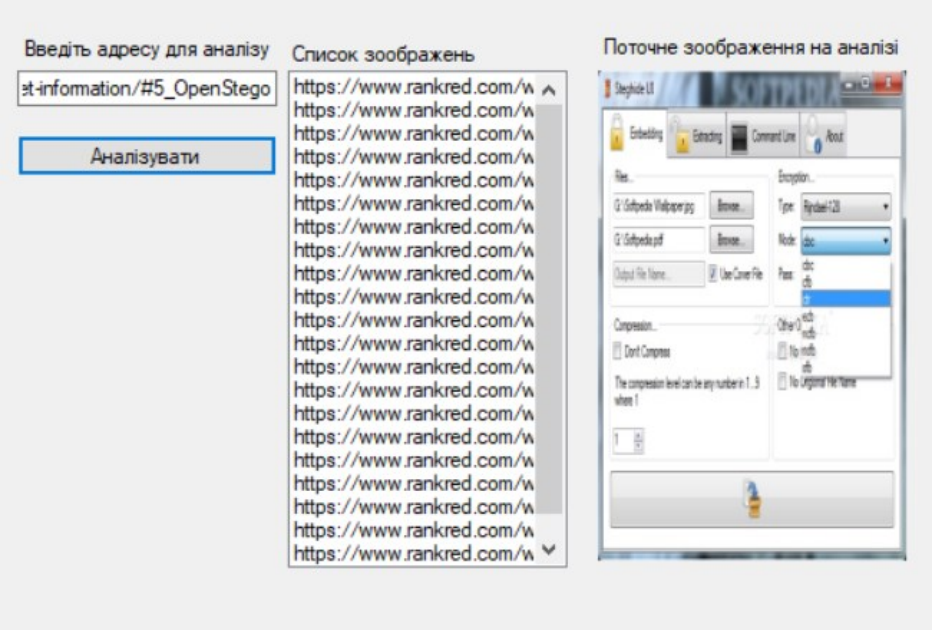


Рисунок 3.13 – Третій приклад з розробленого додатку

Далі будуть показані фото для підтвердження чи дійсно програма зчитує з сайту .img файли.

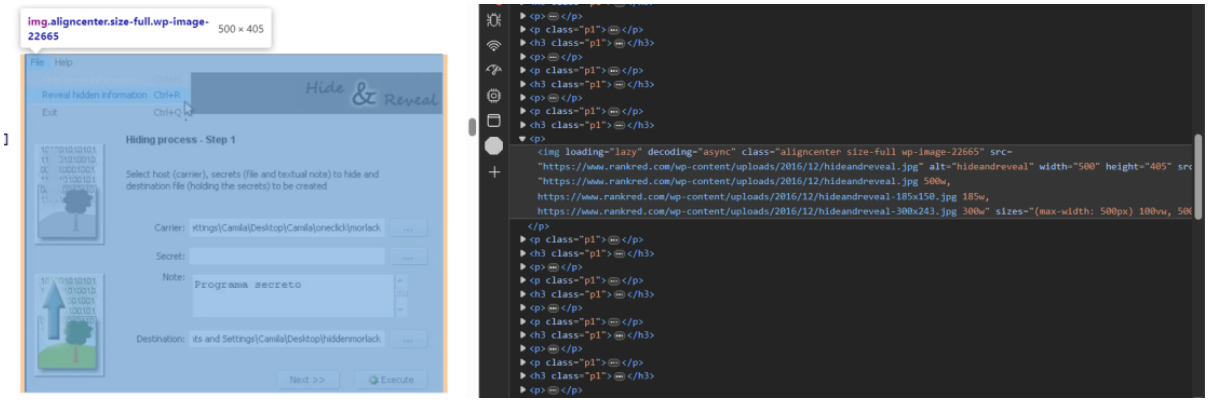


Рисунок 3.14 – Результат перевірки першого прикладу зображення (рис. 3.11)

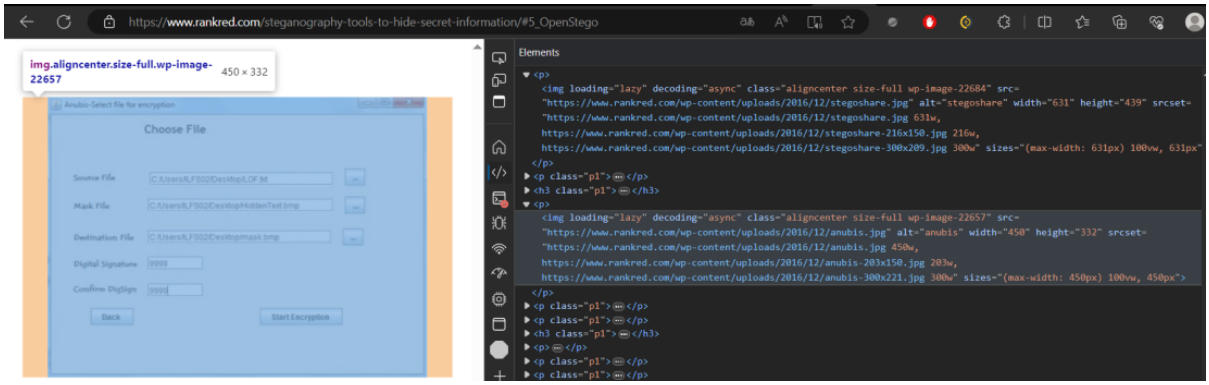


Рисунок 3.15 – Результат перевірки другого прикладу зображення (рис. 3.12)

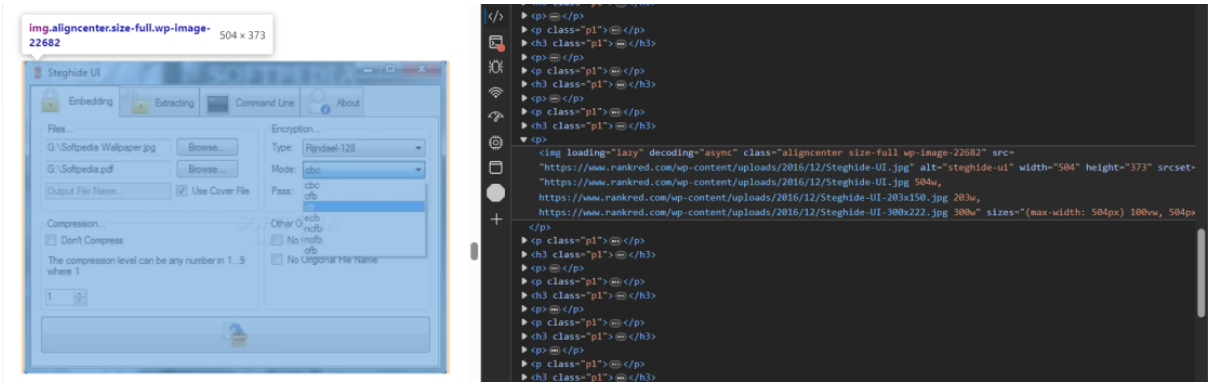


Рисунок 3.16 – Результат перевірки третього прикладу зображення (рис. 3.13)

Наступним кроком ми використаємо низку різних посилань та отримаємо результат обробки нашим додатком, ось декілька з них:

– 30 Useful Steganography Tools To Hide Secret Information - RankRed

– JetIQ (vntu.edu.ua)

– Reggae Music Mix 10 Hours (youtube.com)

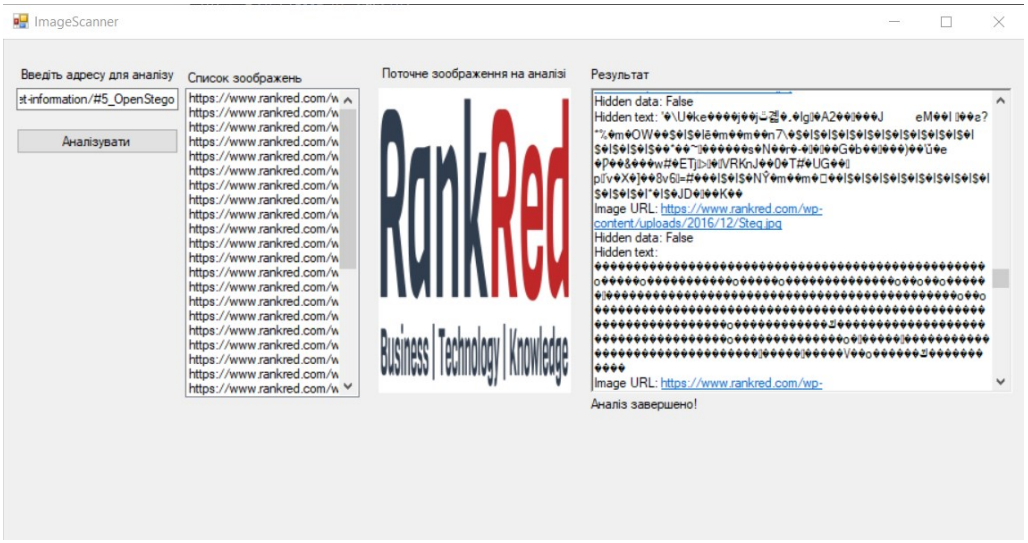


Рисунок 3.17 – Результат першого посилання

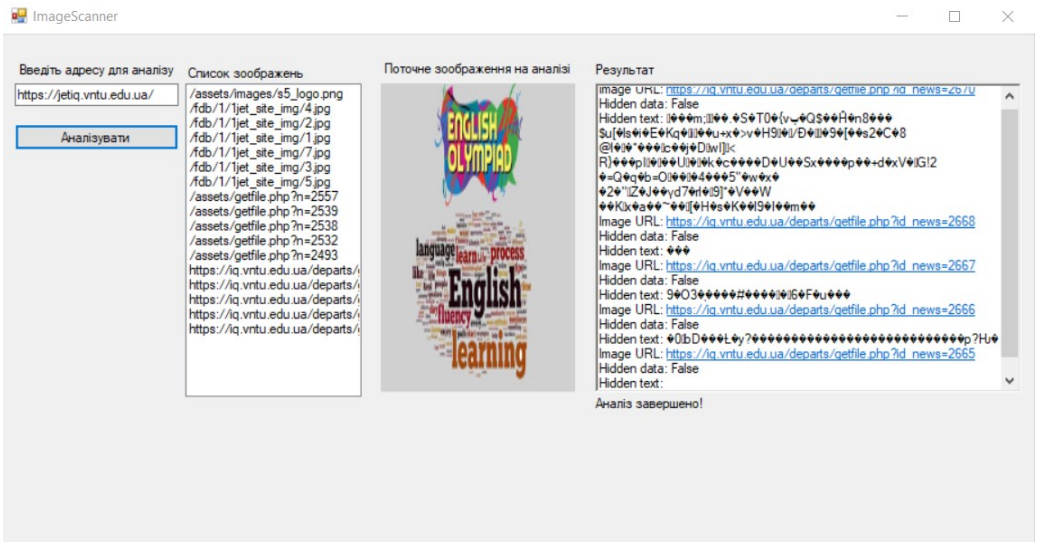


Рисунок 3.18 – Результат другого посилання

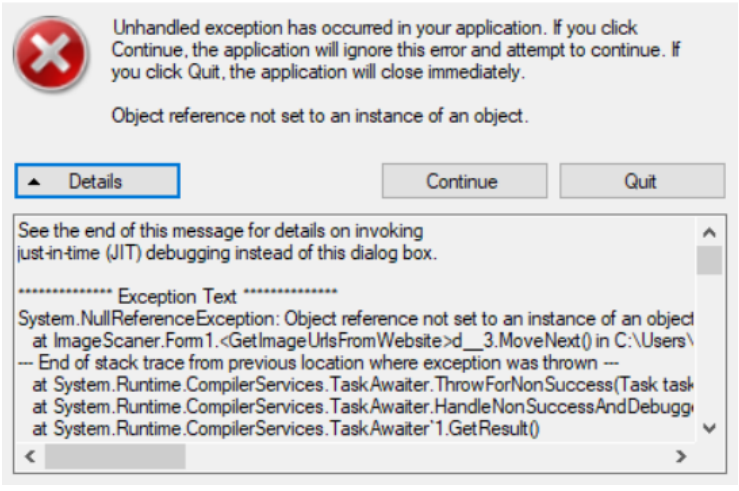


Рисунок 3.19 – Результат третього посилання

Візуальзація даних

В даному тестуванні розглянемо зовнішній вигляд зображень які знаходяться в центрі порограми під час проведення аналізу.

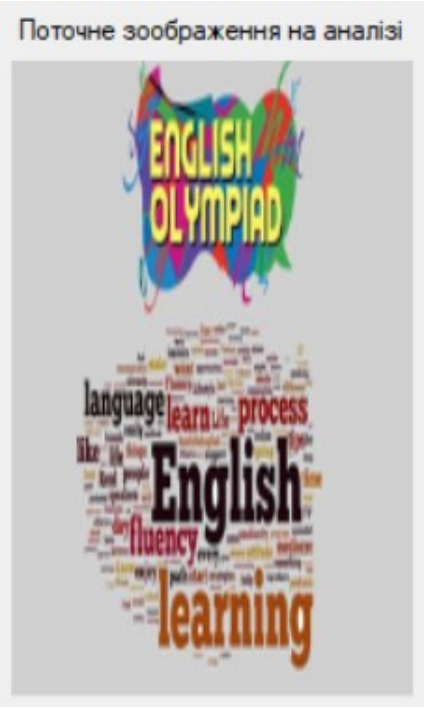


Рисунок 3.20 – Приклад зображення у вікні програми

Рисунок 3.21 – Приклад другого зображення у розробленому додатку

Як видно з попередніх скріншотів, зображення хоч і стисле але чітке та не деформоване.

Тестування продуктивності

Останнім тестуваннння являється тетсування продуктивності, в даному тесті ми будемо розглядати швидкість та якість аналізування зображень на різних сайтах з різним обсягом наявних на них картинках, також виміряємо швидкість виконання поставленої задачі.

Для розрахунку ми використаємо ті посилання які вже були використані в даній роботі, а саме:

- JetIQ (vntu.edu.ua)
- 30 Useful Steganography Tools To Hide Secret Information - RankRed

Вони зображені на рис. 3.17 та рис. 3.18. В процесі тестування маємо такі результати:

− Сайт ВНТУ, з моменту запуску програми пройшло 0.33 секунди, за які програма проаналізувала 17 зображень які були на інтернет ресурсі, з яких було виявлено що тільки в 5 була прихована інформація.

− Інтернет ресурс в якому була вказана інформація про використання корисних стеганографічних програм, зі старту відліку часу пройшло 28 секунд, за як додаток оброобив 36 зображень, з яких 35 зображень містили приховану інформацію.

3.5 Висновки до розділу

Отже, провівши функціональне тестування було виявлено, що всі зображення коректно виділяються і збираються файли тільки формату .img, на наявних скріншотах чітко виявлено відображаються в головному вікні програми та збирає інформацію конкретно з тих інтернет ресурсів які були введені. На останньому скріншоті даного тесту виявлено, що у великих інтернет ресурсах дана програма не може зчитувати і обробляти інформацію так як завелкий обсяг зображень та використовується інший тип шифрування який наявна програма не може обробити.

В тесті візуальзації даних ми розгялнули якість зображень які потрапляють і відображаються у програмі. Останнє тестування показало, що швидкість виконання аналізу даної розробленої програми для виявлення прихованої інформації залежить від кількості зображень які знаходяться на обраному інтернет ресурсі, а також кількості прихованої в них інформації.

ВИСНОВОК

В даній дипломній роботі було виконано розробку програми для виявлення прихованої інформації у зображеннях інтернет-ресурсів, що підтверджує її актуальність та важливість в сучасних умовах. Сучасний світ характеризується швидким розвитком інформаційних технологій та значним зростанням обсягів даних, що передаються через інтернет. Це створює нові виклики у сфері інформаційної безпеки, особливо у контексті захисту конфіденційної інформації та виявлення нелегальних дій, таких як стеганографія.

Також було створено та розглянуто алгоритм функціонування та роботи даної розробленої програми для виявлення прихованої інформації в інтернет-ресурсах. Був описаний кожен крок функціонування додатку та зв’язок між етапами аналізування зображення яке потряпляло в дану розробку з подальшим виведенням результату.

На основі аналізу наявних методів стеганографії та стеганографічного аналізу, було вирішено використати StegDetect як основний інструмент для виявлення прихованої інформації у JPEG-зображеннях. StegDetect, завдяки своїм можливостям з виявлення різних стеганографічних методів та використанню статистичних аналізів, довів свою ефективність і надійність.

Програма була розроблена з використанням мови програмування C# в середовищі розробки Visual Studio. Вибір C# обґрунтований його продуктивністю, підтримкою об'єктно-орієнтованого програмування та зручністю інтеграції з іншими технологіями. Visual Studio була обрана завдяки своїм потужним інструментам для розробки, відладки та тестування програмного забезпечення.

Актуальність розробки даної програми визначається кількома ключовими факторами:

− Зростання обсягів цифрових даних: Збільшення кількості зображень та інших медіа файлів в інтернеті вимагає ефективних методів для забезпечення їх безпеки.

− Загрози кібербезпеці: Використання стеганографії для передачі конфіденційної інформації або здійснення кіберзлочинів стає все більш поширеним.

− Необхідність автоматизації: Автоматизація процесу виявлення прихованої інформації дозволяє знизити навантаження на фахівців з безпеки та підвищити ефективність моніторингу великих обсягів даних.

Також була проведена низка тестів, які показали користність розробленої програми для виявлення прихованої інформації в інтернет-ресурсах. Було розглянуто як функціональність даної розробки так і візуалізація отриманих зображень, результатів які виводяться в окреме блок у вікні, а також тестування продуктивності, що показав плавну та чітку роботу програми без затримок та сильного сповільнення роботи.

Розроблена програма, заснована на StegDetect, дозволяє ефективно ідентифікувати приховану інформацію в JPEG-зображеннях, що робить її важливим інструментом у сфері інформаційної безпеки. Вона забезпечує надійність, точність і швидкість аналізу, що є критично важливим в умовах сучасних кіберзагроз. Подальше вдосконалення та інтеграція нових методів стеганографічного аналізу дозволить підвищити її функціональні можливості та ефективність у боротьбі з нелегальним використанням стеганографії.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1.Шишкін, О. В. (2013). Формування і виявлення цифрових водяних знаків у системі автоматичної ідентифікації ультракороткохвильових радіопередач. Вісник Національного авіаційного університету, (2), 57-61.

2. Прогонов, Д. О. (2017). Ефективність універсального стегодетектору фаріда при вбудовуванні даних у цифрові зображення згідно адаптивних методів.

3. Онікієнко, С. С. (2023). Метод автоматичного виявлення об'єктів з невеликою роздільною здатністю на зображеннях.

4. Федюшин, О. І., & Ковальчук, Д. Ю. (2022). Моделі і методи RANSOMWARE атак в кіберпросторі.

5. Наукова стаття URL: (PDF) A Survey of Image Information Hiding Algorithms Based on Deep Learning (researchgate.net) (дата звернення 23.05.2024).

6. Особливості приховування прихованої інформації URL: document.pdf (ldubgd.edu.ua) (дата звернення 23.05.2024)

7. StegDetect URL: Tag: steganography - Niels Provos (дата звернення 23.05.2024).

8. Olson, E., Carter, L., & Liu, Q. (2017). A Comparison Study using Stegexpose for Steganalysis.

9. Jeon, J., & Cho, Y. (2019). Construction and performance analysis of image steganography-based botnet in KakaoTalk openchat. Computers, 8(3), 61.

10. Прозур, В. (2023). Аналіз видів генеративних змагальних мереж. Наука і техніка сьогодні, (11 (25)).

11. Загальна інформація про Стеганографію URL: What is Steganography? - GeeksforGeeks (дата звернення 23.05.2024).

12. Наукова стаття URL: 90 questions with answers in STEGANOGRAPHY | Scientific method (researchgate.net) (дата звернення 23.05.2024)

13. Гороховатський, В. О., & Гадецька, С. В. (2020). Статистичне оброблення та аналіз даних у структурних методах класифікації зображень.

14. Денисюк, В. О. Реалізація стенографічного алгоритму захисту даних з використанням файлів зображень. Техніка, енергетика, транспорт АПК.-2018.-№ 1-С. 29-38.

15. Хома, Д. Ю. (2023). Стеганографічні алгоритми та стегоаналіз на основі класичних методів і нейронних мереж. Публікується згідно з рішенням Вченої ради ДВНЗ «Донецький національний технічний університет»(протокол № 2 від 29.02. 2024)., 42.

16. Яковенко, А. О. (2020). АЛГОРИТМИ МАШИННОГО НАВЧАННЯ. С 91 Матеріали III Всеукраїнської науково-практичної інтернет-конференції студентів, аспірантів та молодих вчених за тематикою «Сучасні комп'ютерні системи та мережі в управлінні»: збірка наукових праць/Під редакцією ГО Райко.–Херсон: Видавництво ФОП Вишемирський ВС, 2020.–312 с., 88.

17. Мороз, В. В. (2008). Часова інтерполяція на основі аналізу вейвлетних доменів. Радіоелектронні і комп'ютерні системи, (6), 264-268.

18. Nedashkivskyi, Y. (2019). Ефективність інформаційної технології для аналізу та прогнозування часових рядів з фрактальними властивостями на основі лінгвістичного моделювання. COMPUTER-INTEGRATED TECHNOLOGIES: EDUCATION, SCIENCE, PRODUCTION, (34), 79-90.

19. Майор, Є. В. (2024, January). ОГЛЯД РІЗНОМАНІТНИХ МЕТОДІВ ВИКОРИСТАННЯ НЕЙРОННИХ МЕРЕЖ В КОНТЕКСТІ КІБЕРБЕЗБЕКИ. In VII International scientific and practical conference «Scientific Research: Theoretical

Foundations and Practical Applications»(January 24-26, 2024) Vienna, Austria, International Scientific Unity. 2024. 596 p. (p. 162).

20. Наукова стаття URL: IEEE Xplore

21. Форум URL: Newest 'steganography' Questions - Stack Overflow (дата звернення 23.05.2024)

22. Грицюк, П., & Гаврилюк, М. (2023, November). Використання машинного навчання для управління процесами аграрної економіки. In 2023 2nd International Conference on Innovative Solutions in Software Engineering (ICISSE) (p. 150).

23. Форум URL: Steganography (reddit.com) (дата звернення 23.05.2024)

24. Digital image steganography: Survey and analysis of current methods URL: [PDF] Digital image steganography: Survey and analysis of current methods | Semantic Scholar (дата звернення 23.05.2024)

25. Березький, О. М., Піцун, О. Й., Лящинський, П. Б., Лящинський, П. Б., & Мельник, Г. М. (2017). Інтелектуальна система автоматизованої мікроскопії аналізу гістологічних та цитологічних зображень. Штучний інтелект.

26. Куделя, В. О. (2023). Розроблення вебзастосунку з редагування зображень.

27. Вовк, А. І., Гірник, А. В., & Фишко, Є. О. (2008). ГРАФІЧНИЙ ІНТЕРФЕЙС КОРИСТУВАЧА В АНАЛІТИЧНИХ ІНТЕРНЕТ-СИСТЕМАХ. Комп'ютерні технології в будівництві: Матеріали VI Міжнародної науково-технічної конференції «КОМТЕХБУД 2008»: Київ–Севастополь, 9-12 вересня 2008 р., 38.

28. Константінов, К. А., & Дібрівний, О. А. (2024). РОЗРОБКА ЗАСТОСУНКУ З ЕЛЕМЕНТАМИ ГЕЙМИФІКАЦІЇ ДЛЯ УПРАВЛІННЯ ПРОЦЕСАМИ В АВТОСАЛОНІ МОВОЮ C# З ВИКОРИСТАННЯМ ASP .NET. «Сучасна молодь в світі інформаційних технологій»: матеріали, 39.

29. Дідківська, С. О. (2016). Сучасні мови програмування та інструменти розробки програмного забезпечення. Актуальні питання сучасної інформатики, 3, 40-43.

30. Литвинова, С. Г. (2014). Поняття й основні характеристики хмаро орієнтованого навчального середовища середньої школи. Інформаційні технології і засоби навчання, (40, вип. 2), 26-41.

31. Голуб, І. О. (2023). Інформаційна технологія проєктування системи відео на замовлення (Master's thesis, Сумський державний університет).

32. Коноваленко, І. В., & Марущак, П. О. (2020). Платформа. NET та мова програмування C# 8.0.

33. Ярьоменко, М. Д., & Биков, А. О. АВТОМАТИЗОВАНА СИСТЕМА ДЛЯ ПОШУКУ РОБОТИ НА МОРСЬКИХ СУДАХ. Актуальні питання розвитку інформаційних технологій: тези доповідей Всеукраїнської конференції молодих учених (Маріуполь, 24 листопада 2020 р.)/ДВНЗ «ПДТУ».–Маріуполь: ПДТУ, 2020.–107 с., 38.

34. Стаття URL: Форми в HTML: як створити базові форми за допомогою HTML (freecodecamp.org) (дата звернення 23.05.2024)

35.Стаття URL: Active button (codepen.io) (дата звернення 23.05.2024)

ДОДАТКИ

Додаток А. Технічне завдання
Вінницький національний технічний університет
Факультет менеджменту та інформаційної безпеки
Кафедра менеджменту та безпеки інформаційних систем

ЗАТВЕРДЖУЮ
Голова секції "Управління інформаційною безпекою" кафедри МБІС
д.т.н., професор
_____ Юрій ЯРЕМЧУК
" 22 " березня 2024 р.

ТЕХНІЧНЕ ЗАВДАННЯ

до бакалаврської дипломної роботи на тему:

_____
_____Розробка програми виявлення прихованої інформації в _____
_____ інтернет-ресурсах _____

08-72.БДР.008.00.081.ТЗ

Керівник бакалаврської дипломної роботи
к.т.н., доцент
_____ Карпінець В.В.
"_____" _____

Вінниця – 2024 р.

1. Найменування та область застосування

Програма для виявлення прихованої інформації в інтернет-ресурсах. Область застосування: виявлення прихованого тексту або іншого типу інформації в зображеннях інтернет-ресурсів.

2.Підстава для розробки

Розробка виконується на основі наказу ректора ВНТУ № 80 від 11.03.2024 р.

3. Мета та призначення розробки

3.1 Мета розробки: розробка програми для забезпечення інформаційної безпеки.

3.2 Призначення: розроблена програма призначена для захисту конфіденційних даних, інтелектуальної власності.

4. Джерела розробки

4.1. Holoska, J., Oplatkova, Z., Zelinka, I., & Senkerik, R. (2010, September). Comparison between neural network steganalysis and linear classification method stegdetect. In 2010 Second International Conference on Computational Intelligence, Modelling and Simulation (pp. 15-20). IEEE.

4.2. Бурячок В.Л. Політика інформаційної безпеки: підручник. / В.Л.Бурячок, Р.В.Грищук, В.О.Хорошко / За заг. ред. докт. техн. наук, проф. В.О. Хорошка. – К.: ПВП «Задруга», 2014. – 222 с.

4.3. Єсін В.І. Безпека інформаційних систем і технологій / В.І.Єсін, О.О. Кузнецов, Л.С. Сорока. – Харків: ХНУ імені В.Н. Каразіна, 2013. – 632 с.

4.4. Зима, В. Л. (2021). Розробка програмного забезпечення для виявлення прихованої інформації в растрових зображеннях на основі досліджених властивостей стеганографічних контейнерів.

5. Вимоги до програми

5.1 Вимоги до функціональних характеристик:

5.1.1 Програмний засіб повинен мати зручний, легкий у використанні інтерфейс користувача;

5.1.2 Реалізація методу не повинна вимагати спеціальних ліцензійних програмних додатків;

5.1.3 Програмний засіб повинен виконувати процес обробки зображення котре знаходиться в інтренет-ресурсі.

5.2 Вимоги до надійності:

5.2.1 Програмний засіб повинен працювати без помилок, у випадку виникнення критичних ситуацій необхідно передбачити виведення відповідних повідомлень;

5.2.2 Програмний засіб повинен виконувати свої функції.

5.3 Вимоги до складу і параметрів технічних засобів:

– процесор – Pentium 1400 МГц і подібні до них;

– оперативна пам’ять – не менше 256 Mb;

– середовище функціонування – операційна система сімейство Windows;

– вимоги до техніки безпеки при роботі з програмою повинні відповідати існуючим вимогам та стандартам з техніки безпеки при користуванні комп’ютерною технікою.

6. Вимоги до програмної документації

6.1 Обов’язкова поетапна інструкція для майбутніх користувачів, наведена у пункті 3.4

7. Вимоги до технічного захисту інформації

7.1 Необхідно забезпечити захист розроблюваного програмного засобу від несанкціонованого використання.

7.2 Дану програму матимуть змогу придбати тільки зареєстровані користувачі, впишучи свої персональні дані.

8. Техніко-економічні показники

8.1 Цінність результатів використання даного проекту повинна перевищувати витрати на його реалізацію.

8.2 Має бути реалізований таким чином, щоб підходити для використання широкого загалу.

9. Стадії та етапи розробки

№ з/п	Назва етапів бакалаврської дипломної роботи	Початок	Закінчення
1.	Визначення напрямку бакалаврської роботи, формулювання теми	20.02.2024	04.03.2024
2.	Аналіз предметної області обраної теми	26.03.2024	11.04.2024
3.	Розробка алгоритму роботи	09.05.2024	16.05.2024
4.	Написання бакалаврської роботи на основі розробленої теми	16.05.2024	20.05.2024
5.	Передзахист бакалаврської дипломної роботи		
6.	Виправлення, уточнення, корегування бакалаврської дипломної роботи	22.05.2024	27.05.2024
7.	Захист бакалаврської дипломної роботи	13.06.2024	17.06.2024

10. Порядок контролю та прийому

10.1 До приймання бакалаврської дипломної роботи надається:

– ПЗ до бакалаврської дипломної роботи;

- демонстрація результату бакалаврської дипломної роботи;
- презентація;
- відзив керівника роботи;
- відзив рецензента

Технічне завдання до виконання прийняв ____Горохов А.В_____

Додаток Б. Лістинг виявлення прихованої інформації

```
using System;
using System.Collections.Generic;
using System.Drawing;
using System.IO;
using System.Net.Http;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using HtmlAgilityPack;

namespace ImageScaner
{
   public partial class Form1 : Form
   {
      public Form1()
      {
         InitializeComponent();
      }

      private void Form1_Load(object sender, EventArgs e)
      {
      }

      private async void button1_Click(object sender, EventArgs e)
      {
         string url = textBox1.Text;
         listBox1.Items.Clear();
         richTextBox1.Clear();
         label5.Text = "Початок аналізу...";
         List<string> imageUrls = await GetImageUrlsFromWebsite(url);

         foreach (var imageUrl in imageUrls)
         {
            listBox1.Items.Add(imageUrl);
            Bitmap bitmap = await DownloadImage(imageUrl);
            if (bitmap != null)
            {
               pictureBox1.Image = bitmap;
               bool hiddenData = AnalyzeImageForHiddenFiles(bitmap);
               string hiddenText = ExtractHiddenText(bitmap);
               richTextBox1.AppendText($"Image URL: {imageUrl}\nHidden data: {hiddenData}\nHidden text: {hiddenText}\n");
               label5.Text = $"Аналіз зображення: {imageUrl}";
            }
         }

         label5.Text = "Аналіз завершено!";
      }

      private async Task<List<string>> GetImageUrlsFromWebsite(string url)
      {
         List<string> imageUrls = new List<string>();
```

```
    using (HttpClient client = new HttpClient())
    {
        string html = await client.GetStringAsync(url);
        HtmlAgilityPack.HtmlDocument doc = new HtmlAgilityPack.HtmlDocument();
        doc.LoadHtml(html);
        foreach (var node in doc.DocumentNode.SelectNodes("//img[@src]"))
        {
            string src = node.GetAttributeValue("src", null);
            if (!string.IsNullOrEmpty(src))
            {
                imageUrls.Add(src);
            }
        }
    }
    return imageUrls;
}

private async Task<Bitmap> DownloadImage(string imageUrl)
{
    using (HttpClient client = new HttpClient())
    {
        try
        {
            byte[] imageBytes = await client.GetByteArrayAsync(imageUrl);
            using (MemoryStream ms = new MemoryStream(imageBytes))
            {
                return new Bitmap(ms);
            }
        }
        catch
        {
            return null;
        }
    }
}

private bool CheckForHiddenFiles(byte[] imageData)
{
    string[] fileSignatures = { "504B0304", "52617221", "25504446" }; // Zip, Rar, PDF
    string hexString = BitConverter.ToString(imageData).Replace("-", "");

    foreach (string signature in fileSignatures)
    {
        if (hexString.Contains(signature))
        {
            return true;
        }
    }

    return false;
}

private bool AnalyzeImageForHiddenFiles(Bitmap bitmap)
{
    byte[] imageData = ImageToByteArray(bitmap);
```

```
        return CheckForHiddenFiles(imageData);
    }

    private byte[] ImageToByteArray(Bitmap bitmap)
    {
        using (MemoryStream ms = new MemoryStream())
        {
            // Ensure the bitmap is in a compatible format
            Bitmap clone = new Bitmap(bitmap);
            clone.Save(ms, System.Drawing.Imaging.ImageFormat.Png);
            return ms.ToArray();
        }
    }

    private string ExtractHiddenText(Bitmap bitmap)
    {
        List<byte> extractedBits = new List<byte>();
        for (int y = 0; y < bitmap.Height; y++)
        {
            for (int x = 0; x < bitmap.Width; x++)
            {
                Color pixel = bitmap.GetPixel(x, y);
                extractedBits.Add((byte)(pixel.R & 1)); // Extract LSB from Red channel
                extractedBits.Add((byte)(pixel.G & 1)); // Extract LSB from Green channel
                extractedBits.Add((byte)(pixel.B & 1)); // Extract LSB from Blue channel
            }
        }

        List<byte> byteList = new List<byte>();
        for (int i = 0; i < extractedBits.Count; i += 8)
        {
            byte b = 0;
            for (int bitIndex = 0; bitIndex < 8; bitIndex++)
            {
                if (i + bitIndex < extractedBits.Count)
                {
                    b = (byte)(b | (extractedBits[i + bitIndex] << (7 - bitIndex)));
                }
            }
            byteList.Add(b);
        }

        string hiddenText = Encoding.UTF8.GetString(byteList.ToArray());

        int nullIndex = hiddenText.IndexOf('\0');
        if (nullIndex >= 0)
        {
            hiddenText = hiddenText.Substring(0, nullIndex);
        }

        return hiddenText;
    }
  }
}
```

Додаток В. Ілюстраційний матеріал

Розробка програми для виявлення прихованої інформації у зображеннях інтернет-ресурсів

Виконав студент групи 1КІТС-20Б Горохов А. В.
Науковий керівник: к.т.н., доц., зав. каф. МБІС Карпінець В. В

Актуальність теми

Актуальність програми для виявлення прихованої інформації в інтернет-ресурсах зумовлена зростанням кількості кіберзагроз та злочинних дій, що використовують стеганографію для приховування шкідливих даних. Програма сприяє забезпеченню інформаційної безпеки, захисту конфіденційних даних і інтелектуальної власності, а також підтримує правоохоронні органи в розслідуванні злочинів. У сучасних умовах, коли цифрові комунікації відіграють критично важливу роль, така програма є необхідним інструментом для забезпечення безпеки і захисту інформації.

Мета роботи

Метою даної роботи є розробка програми для виявлення прихованої інформації в зображеннях на інтернет-ресурсах, що дозволить своєчасно ідентифікувати та запобігти використанню стеганографії для приховування шкідливих даних, забезпечуючи таким чином підвищений рівень інформаційної безпеки.

Новизна

Новизна даної роботи полягає в інтеграції сучасних методів стеганографічного аналізу, таких як StegDetect, з ефективним використанням програмного забезпечення на основі мови C#. Розробка передбачає створення зручного графічного інтерфейсу для користувача та оптимізацію алгоритмів виявлення прихованої інформації, що забезпечує високу точність і швидкість аналізу зображень на інтернет-ресурсах.

Об’єкт та предмет дослідження

Об'єктом дослідження даної роботи є зображення, розміщені на інтернет-ресурсах, які можуть містити приховану інформацію.

Предметом дослідження є методи та алгоритми виявлення прихованої інформації в зображеннях, зокрема, використання інструментів стеганографічного аналізу, таких як StegDetect, для забезпечення інформаційної безпеки.

Порівняння стеганографічних методів

Метод	Принцип дії	Переваги	Недоліки	Рівень непомітності	Стійкість до атак
LSB (Least Significant Bit)	Заміна найменш значущих бітів пікселів зображення на біти прихованої інформації	Простота реалізації, велика прихована ємність	Легкість виявлення, вразливість до змін зображення	Низький	Низька
DCT (Discrete Cosine Transform)	Зміна коефіцієнтів дискретного косинусного перетворення в JPEG-зображеннях	Стійкість до стиснення JPEG, менше помітність	Складніша реалізація, менша прихована ємність	Середній	Висока
Spread Spectrum	Розподіл секретної інформації на декілька частин і вставка у різні частини зображення	Висока стійкість до атак, стійкість до шуму	Складність реалізації, низька прихована ємність	Високий	Висока
QIM (Quantization Index Modulation)	Модуляція секретної інформації шляхом зміни індексів квантованих коефіцієнтів перетворення	Висока стійкість до компресії, добра прихована ємність	Складність реалізації, вразливість до певних атак	Середній	Середгня

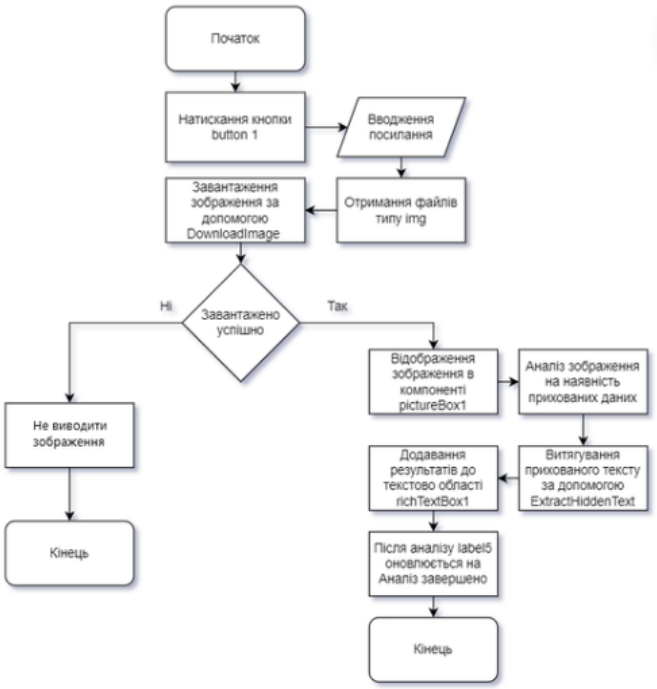
Прихована інформація

Прихована інформація в зображеннях – це дані, що зберігаються в графічних файлах так, щоб вони не були видимими для звичайного перегляду. Цей процес відомий як стеганографія, яка використовується для секретного обміну інформацією. Прихована інформація може бути текстовим повідомленням, іншим зображенням або будь-яким іншим типом даних, що зберігається в пікселях зображення, не змінюючи його зовнішнього вигляду значною мірою.

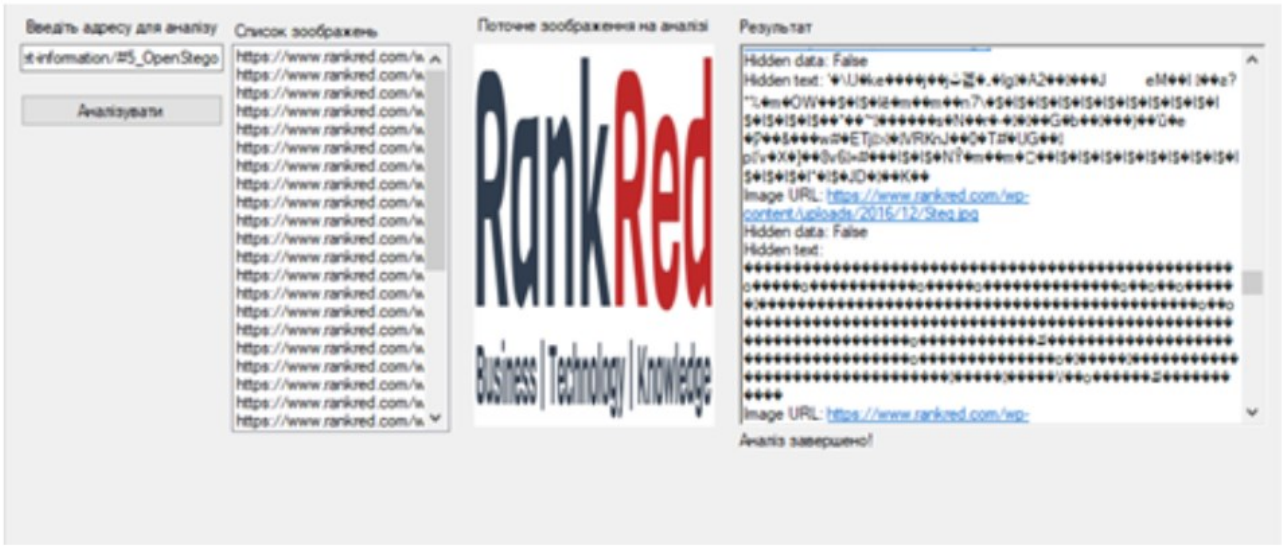
Структура роботи програмного додатку

В даній структурі розроблена модель роботи програми, Чітко описані кроки, етапи застосунку. Сформованиц чіткий цикл по якому рухається обробка зображення поки не пройде повне коло, щоб користувач отримав результат

Алгоритм роботи програми



Приклад виявлення прихованої інформації



Порівняння з програмами аналогів

Назва програми	Основні особливості	Види захисту	Оцінка функціоналу
Розроблена програма	Виявлення прихованої інформації у форматах зображень, таких як JPEG, BMP, PNG тощо	Аналіз на основі статистики існування прихованої інформації в різних форматах зображень.	Добре підходить для виявлення основних методів стеганографії
OpenStego	Підтримка різних методів стеганографії включаючи шифрування та паролізацію,	Простота реалізації, велика прихована ємність	Легкість виявлення, вразливість до змін зображення
OutGuess	Пошук прихованої інформації методом LSB і аналіз статистики	Використовує LSB для приховування даних, важко виявляється за допомогою стандартних методів аналізу зображень.	Підходить для простих методів, може бути обхідним для складних
StegExpose	Аналіз зображень на наявність прихованої інформації	Перевірка LSB, аналіз стеганографічних артефактів	Ефективний проти простих методів, може пропускати складні

Висновки

Актуальність розробки даної програми визначається кількома ключовими факторами:

- Зростання обсягів цифрових даних: Збільшення кількості зображень та інших медіа файлів в інтернеті вимагає ефективних методів для забезпечення їх безпеки.
- Загрози кібербезпеці: Використання стеганографії для передачі конфіденційної інформації або здійснення кіберзлочинів стає все більш поширеним.

Розроблена програма, заснована на StegDetect, дозволяє ефективно ідентифікувати приховану інформацію в JPEG-зображеннях, що робить її важливим інструментом у сфері інформаційної безпеки. Вона забезпечує надійність, точність і швидкість аналізу, що є критично важливим в умовах сучасних кіберзагроз. Подальше вдосконалення та інтеграція нових методів стеганографічного аналізу дозволить підвищити її функціональні можливості та ефективність у боротьбі з нелегальним використанням стеганографії.

Додаток Г. Протокол перевірки на антиплагіат

ПРОТОКОЛ
ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ НА
НАЯВНІСТЬ ТЕКСТОВИХ ЗАПОЗИЧЕНЬ

Назва роботи: Розробка програми виявлення прихованої інформації у зображеннях інтернет-ресурсів

Тип роботи: бакалаврська дипломна робота
(БДР, МКР)

Підрозділ: Кафедра менеджменту та безпеки інформаційних систем
Факультет менеджменту та інформаційної безпеки
(кафедра, факультет)

Показники звіту подібності Unicheck

Оригінальність 98 % Схожість 2 %

Аналіз звіту подібності (відмітити потрібне):

1. Запозичення, виявлені у роботі, оформлені коректно і не містять ознак плагіату.

2. Виявлені у роботі запозичення не мають ознак плагіату, але їх надмірна кількість викликає сумніви щодо цінності роботи і відсутності самостійності її виконання автором. Роботу направити на розгляд експертної комісії кафедри.

3. Виявлені у роботі запозичення є недобросовісними і мають ознаки плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень.

Особа, відповідальна за перевірку _____ Коваль Н.П.
(підпис) (прізвище, ініціали)

Ознайомлені з повним звітом подібності, який був згенерований системою Unicheck щодо роботи.

Автор роботи _____ Горохов А.В.
(підпис) (прізвище, ініціали)

Керівник роботи _____ Карпінець В.В.
(підпис) (прізвище, ініціали)