Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)
Кафедра комп'ютерних наук
(повна назва кафедри)

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

бакалавр
(назва освітнього ступеня)

на тему: Розробка програмних засобів передачі та аналізу прихованих даних засобами нейронних мереж

Виконав: студент IV курсу, групи СНзс-41
спеціальності 122 Комп’ютерні науки
(шифр і назва спеціальності)

_____ Лабунський Ю.В
(підпис) (прізвище та ініціали)

Керівник _____ Гащин Н.Б.
(підпис) (прізвище та ініціали)

Нормоконтроль _____ Шимчук Г.В.
(підпис) (прізвище та ініціали)

Завідувач кафедри _____ Боднарчук І.О.
(підпис) (прізвище та ініціали)

Рецензент _____ _____
(підпис) (прізвище та ініціали)

Тернопіль - 2024

Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)

Кафедра комп'ютерних наук
(повна назва кафедри)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Боднарчук І.О.
(підпис) (прізвище та ініціали)
«__» _____ 2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня Бакалавр
(назва освітнього ступеня)

за спеціальністю 122 Комп'ютерні науки
(шифр і назва спеціальності)

Студенту Лабунському Юрію Вікторовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Розробка програмних засобів передачі та аналізу прихованих даних засобами нейронних мереж

Керівник роботи Гащин Надія Богданівна, к.т.н., доц.
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом ректора від « 29 » 04 2024 року № 4/7-473

2. Термін подання студентом завершеної роботи 10.06.2024 р.

3. Вихідні дані до роботи наукові літературні джерела

4. Зміст роботи (перелік питань, які потрібно розробити)
Вступ
1. Аналітична частина
2. Реалізація стеганографічних методів як прихованих каналів
3. Розробка фреймворку для аналізу прихованих каналів
4. Безпека життєдіяльності, основи охорони праці
Висновки

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, слайдів)
1. Титулка. 2. Актуальність. 3. Мета, задачі дослідження.
4. Схема методу Deep Steganography. 5. Результати методу Deep Steganography.
6. Приклад результатів перетворень методу Deep Steganography
7. Схема розширення методу Deep Steganography. 8. Приклад роботи запропонованого розширення методу Deep Steganography. 9. Програмні засоби. 10. Архітектура фреймворку
11. Швидкість роботи методів фреймворку. 12. Висновки

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Безпека життєдіяльності, основи хорони праці			

7. Дата видачі завдання 2024 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1.	Ознайомлення з завданням до кваліфікаційної роботи	29.04 – 30.04.24	*Виконано*
2.	Підбір джерел про приховані канали	01.05 – 05.05.24	*Виконано*
3.	Опрацювання джерел про стеганографічні методи		
	як приховані канали	06.05 – 10.05.24	*Виконано*
4.	Проведення дослідження щодо побудови моделей та		
	Створення людино-машинного інтерфейсу	11.05 – 15.05.24	*Виконано*
5	Розроблення програмного коду	16.05 – 19.05.24	*Виконано*
6.	Оформлення розділу «Аналітична частина»	20.05 – 23.05.24	*Виконано*
7.	Оформлення розділу «Реалізація стеганографічних		
	методів як прихованих каналів»	24.05 – 27.05.24	*Виконано*
8.	Оформлення розділу «Розробка фреймворку для аналізу	28.05 – 30.05.24	*Виконано*
	прихованих каналів»		
9.	Виконання завдання до підрозділу «Безпека		
	життєдіяльності, основи хорони праці»	31.05 – 03.06.24	*Виконано*
10.	Оформлення кваліфікаційної роботи	20.05 – 03.06.24	*Виконано*
11.	Нормоконтроль	01.06 – 05.06.24	*Виконано*
12.	Перевірка на плагіат	04.06 – 07.06.24	*Виконано*
13.	Попередній захист кваліфікаційної роботи	07.06 – 09.06.24	*Виконано*
14.	Захист кваліфікаційної роботи	11.06.24	

Студент _____ Лабунський Ю.В.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Гащин Н.Б.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Розробка програмних засобів передачі та аналізу прихованих даних засобами нейронних мереж // Лабунський Юрій Вікторович // Тернопільський національний технічний університет імені Івана Пулюя, факультет комп'ютерно-інформаційних систем та програмної інженерії, кафедра комп'ютерних наук, група СНзс–41 // Тернопіль, 2024 // С. – 54, рис. – 13, табл. – 4, слайдів – 12, бібліогр. – 13.

Ключові слова: приховані канали, штучна нейронна мережа, машинне навчання, фреймворк, стеганографія

Кваліфікаційна робота присвячена побудові фреймворку, який дозволяє агрегатувати різні методи та архітектури нейронних мереж незалежно один від одного та мови програмування, обраної для реалізації, для того, щоб забезпечити єдину систему управління та розширення методів.

У першому розділі даної роботи докладно розглядається проблема можливості організації прихованих каналів шляхом використання нейронних мереж та можливі шляхи її вирішення.

У другому розділі розглядається необхідна теоретична база та відомості про приховані канали та стеганографічні методи, що використовуються в сучасній комп'ютерній безпеці. Також досліджено існуючі методи та підходи до реалізації прихованих каналів та стеганографічних контейнерів. Розглядаються наведені реалізації методів Deep Steganography, HiDDeN. Також запропоновано розширення методу Deep Steganography, яке дозволяє збільшити можливий обсяг переданої секретної інформації за один сеанс передачі даних.

Третій розділ розглядає ключові особливості розробленого фреймворку — використовувані компоненти, підходи, розроблену архітектуру та інші деталі. Також описано сценарій використання фреймворку можливими користувачами системи на практиці. Наводяться результати продуктивності та швидкості роботи з різними методами у рамках взаємодії з фреймворком.

ANNOTATION

Development of software for transmission and analysis of hidden data using neural networks // Labunskyi Yurii // Ternopil Ivan Pul'uj National Technical University, Faculty of Computer Information Systems and Software Engineering, Department of Computer Science // Ternopil, 2024 // P. - 54, Fig. - 13, Table - 4, Slide - 12, References - 13.

Keywords: covert channels, artificial neural network, machine learning, framework, steganography

Thesis deals with the construction of a framework that allows aggregating different methods and architectures of neural networks independently of each other and the programming language chosen for implementation, in order to provide a unified system of management and extension of methods.

In the first section of this work, the problem of the possibility of organizing hidden channels through the use of neural networks and possible ways of solving it is considered in detail.

The second chapter deals with the necessary theoretical background and information about covert channels and steganographic methods used in modern computer security. Existing methods and approaches to the implementation of hidden channels and steganographic containers are also investigated. The following implementations of the Deep Steganography, HiDDeN methods are considered. An extension of the Deep Steganography method is also proposed, which allows to increase the possible amount of transmitted secret information during one data transfer session.

The third section examines the key features of the developed framework—the components used, the approaches, the developed architecture, and other details. The scenario of using the framework by possible users of the system in practice is also described. The results of performance and speed of work with various methods in the framework of interaction with the framework are given.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ СКОРОЧЕНЬ І ТЕРМІНІВ

CUDA (Compute Unified Device Architecture) — архітектура ефективних паралельних обчислень.

HiDDeN – Hiding Data With Deep Networks.

GPU (Graphics Processing Unit) — спеціалізований високопродуктивний графічний процесор.

UUID (Universally Unique Identifier) — стандарт ідентифікації, який використовується при створенні програмного забезпечення,

БД – база даних.

Декодування (decoding) - процес отримання прихованого повідомлення із контейнера (в контексті даної роботи).

Кодування (encoding) - процес приховування повідомлення в контейнері (в контексті цієї роботи).

Контейнер (container / package) — сигнал, потік або файл, що містить приховане повідомлення. У контексті даної роботи під контейнером ми матимемо на увазі медіа-файл, створений з обкладинки та секретного повідомлення.

МН - машинне навчання (machine learning).

Прихований канал (covert channel) — комунікаційний канал, який надсилає інформацію методом, який спочатку був для цього не призначений.

Приховане повідомлення або секрет (covert message / secret) — корисне навантаженням, яке передається приховано (послідовність біт, текстове повідомлення, аудіо- або відео-інформація).

Обкладинка (cover) — вихідне зображення, яке використовується для створення переданого медіа-контейнера.

Стеганографія (steganography) – спосіб передачі або зберігання інформації з урахуванням збереження в таємниці самого факту такої передачі.

ШНМ штучна нейронна мережа (artificial neural network)

ЗМІСТ

ВСТУП

Актуальність теми. ШНМ та МН на даний момент є одними з найактуальніших та найперспективніших областей комп'ютерних наук. Область застосування нейронних мереж досить велика: визначення об'єктів на зображеннях і відео, розпізнавання та синтезу мови, машинний переклад різних текстів, генерація та обробка зображень і відео та багатьох інших завдань.

Можливості, що надаються ШНМ активно входять в область завдань сучасної комп'ютерної безпеки: нейронні мережі, як і класичні алгоритми МН, активно застосовуються для детектування автоматизованих дій у веб-застосунках (визначення ботів і вилучення даних зі сторінок веб-ресурсів), для аналізу трафіку та детектування різних атак, таких як атаки відмови доступу, атаки на веб-додатки (SQL -ін'єкції, перебір значень методу «грубої сили» та інші), нелегітимне сканування ресурсів програми, а також для аналізу інформації — класичні алгоритми кластеризації допомагають розділяти дані різної природи (художні тексти, HTTP -запити, HTML -вміст сторінок) без попереднього навчання, а ШНМ та алгоритми обробки природної мови дозволяють виділити найбільш значущі та критичні сутності серед отриманих об'єктів. Однак, завдання застосування ШНМ і механізмів МН в різні нові галузі комп'ютерних наук є вкрай складним навіть для фахівців. Процеси збору даних, навчання нейромережі, підготовки моделі, підбору параметрів для досягнення максимальних результатів є одними з найбільш трудомістких , що, тим не менш, не зменшує популярність застосування цих напрямків.

Можливості сучасних ШНМ, пов'язані з генерацією та обробкою зображень, можуть бути використані в контексті стеганографії та прихованих каналів, що дозволяє організувати прихований канал за допомогою передачі прихованого повідомлення (секрету) усередині будь-якого медіа-контейнера, наприклад, зображення. На даний момент не існує інструментів, утиліт або фреймворків, що дозволяють практично використовувати та досліджувати дані методи в контексті прихованих каналів та приховування інформації. Виходячи з цього, застосування цих методів неможливо без їх самостійної реалізації за

допомогою однієї з сучасних мов програмування, модифікації архітектури для використання в контексті прихованих каналів, підготовки тренувальних даних та повного опису умов застосування.

Мета роботи – розробка дослідницького фреймворку для можливості реалізації та аналізу прихованих каналів із застосуванням нейронних мереж.

Для досягнення мети виділено ряд завдань:

– реалізувати існуючі методи прихованих каналів за допомогою можливостей ШНМ;

– розширити існуючі методи прихованих каналів із використанням ШНМ;

– розробити фреймворк для реалізації прихованих каналів при допомозі ШНМ.

РОЗДІЛ 1. АНАЛІТИЧНА ЧАСТИНА

1.1 Постановка задачі

На даний момент існують наукові праці, пов'язані із застосуванням нейронних мереж у галузі стеганографії. Основна ідея даних методів полягає у можливості навчання ШНМ таким чином, щоб при створенні контейнера, що містить у собі секретне повідомлення, він був максимально схожий на обкладинку, а повідомлення при відновленні мала повну або максимально можливу схожість з вихідним секретним повідомленням (зображенням, текстом) або іншим типом представлення даних).

На жаль, дані роботи розглядають методи застосування нейронних мереж тільки в контексті стеганографії і є експериментальними, тобто не адаптованими і не готовими для використання в реальних практичних завданнях. Крім того, у цих роботах не розглядаються можливі сценарії використання наведених методів у рамках можливості організації прихованих каналів загалом. Таким чином, на даний момент не існує програмних засобів, інструментів чи застосунків, котрі дають змогу практично застосовувати та досліджувати такі методи для прихованих каналів та власне приховування даних. Більше того, в рамках існуючих робіт часто не розглядається конкретна програмна архітектура, структура або можлива реалізація таких інструментів. Виходячи з цього, використання даних методів неможливо без їх самостійної реалізації за допомогою однієї з сучасних мов програмування, модифікації архітектури для використання в контексті прихованих каналів, підготовки тренувальних даних та повного опису умов використання.

Ця робота присвячена розробці фреймворку, що дозволяє реалізовувати та аналізувати приховані канали за допомогою нейронних мереж та існуючих методів стеганографії в нейронних мережах. До основних вимог розробки фреймворку відносяться кросплатформність, відмовостійкість, масштабованість, децентралізованість, незалежність від мов програмування методів та швидкість обробки даних .

На даний момент, завдання розробки такого фреймворку для реалізації та аналізу прихованих каналів на основі нейронних мереж є актуальною та затребуваною. Крім того, виходячи з аналізу існуючих наукових праць, даний фреймворк є унікальним у своєму роді.

У роботі буде описана архітектура фреймворку, використовувані компоненти, особливості реалізації та загальний сценарій використання фреймворку. Крім того, в рамках роботи буде описано розроблене розширення одного з методів, що дозволяє значно збільшити обсяг інформації, що передається за один сеанс передачі. Також будуть розглянуті особливості реалізації методів приховування інформації мовою програмування Python з використанням Keras API та бібліотеки TensorFlow.

1.2 Загальний опис стеганографічних методів

Розглянемо стеганографічні методи докладніше. Стеганографія є способом приховування власне самого факту передавання повідомлення. Термін «стеганографія» вперше виник у 15-му столітті, коли повідомлення фізично приховували будь-яким чином (спеціальні глиняні дощечки, «невидиме чорнило» за допомогою лимонного соку, та інші методи). У сучасних термінах метою стеганографії є приховування самого факту зберігання або передачі інформації деяким чином або в деякому контейнері. Процес передачі має на увазі під собою приховування повідомлення в якому-небудь контейнері, який потім може бути розкритий тільки одержувачем або групою одержувачів, яким відомо стін алгоритм кодування або приховування інформації. Більше того, сам контейнер може бути публічно доступний, але незважаючи на це, метод приховування інформації і факт наявності інформації повинні бути відомі тільки сторонам відправника і одержувача, щоб ніякий інший учасник спілкування або зловмисник не зміг розкрити повідомлення, що передається, знаючи факт його передачі . Для підвищення рівня безпеки, секретне повідомлення може бути зашифроване, що збільшує складність виявлення такого спілкування та його розкриття навіть у тому випадку, якщо факт передачі секретного повідомлення

став відомий третій стороні.

Основною проблемою при використанні стеганографічних методів є потенційна зміна властивостей контейнера, як зовнішніх (медіа-властивості: зміна якості зображення, спотворення звуку, втрата даних) так і статистичних (зміна розміру контейнера, довжини повідомлення). Зазвичай, кількість і якість змін у контейнері залежить від двох факторів:

– кількість інформації, що передається (розмір секретного повідомлення);

– кількість можливих змін у контейнері в цілому (наприклад , більша кількість змін на зашумленому і багатобарвному зображенні буде менш помітна, ніж та ж кількість змін на чорно-білому або більш простому в контексті кольоровості зображенні).

Кількість інформації, що передається в контейнерах-зображеннях, прийнято вимірювати в «bits- per- pixels» (далі — «bpp») або кількість бітів інформації, що передаються, на один піксель зображення. Найчастіше, для сучасних методів стеганографії зображення це значення становить 0.4 bpp або менше. Чим довше повідомлення — тим більше значення bpp, отже, тим більше спотворюється оригінальне зображення під впливом об'єму інформації, що передається, і тим більше ці зміни помітні.

1.3 Загальний опис прихованих каналів

Першою згадкою терміна «приховані канали» в технічних статтях та літературі прийнято вважати статтю «A Note on the Confinement Problem» [1] , в якій приховані канали розглядаються як механізм приховування шкідливих або нелегітимних даних усередині деяких дозволених об'єктів, які передаються по визначеному комунікаційному каналу. В царині комп'ютерної безпеки прихованим каналом називається тип повідомлень, що створюють можливість передачі будь-якої інформації між будь-якими процесами, які не повинні між собою взаємодіяти та комутувати за умовами заданої політики безпеки певної системи [2].

Термін «прихований канал» походить від основних необхідних властивостей каналу — його скритності та невиявленності, оскільки ці канали приховані від механізмів управління контролем доступу, і з цієї причини вбудовані в операційну систему або веб-додатки механізми захисту не можуть їх виявити [3]. Незважаючи на це, в реальних умовах приховані канали вимагають серйозних трудовитрат для реалізації на існуючих системах, і часто можуть бути виявлені механізмами моніторингу та продуктивності, такими як:

– моніторинг мережної активності (зв'язок із нелегітимними ресурсами, доступ до невідомих мереж тощо);

– моніторинг навантаження складових системи (завантаження процесора, обсяг записуваних даних на жорсткі диски і т.д.);

– аналіз фізичних характеристик (швидкість обертання корпусних вентиляторів, ступінь нагріву компонентів, потужність електромагнітного випромінювання тощо).

Тим не менш, на практиці це показує, що потрібно аналізувати велику кількість інформаційних потоків і не меншу кількість ПЗ, котре використовується, що може бути неефективно і недоцільно [4] .

Доповненням до можливих недоліків прихованих каналів може бути малий обсяг інформації, що передається через прихований канал (не використовуючи додаткових інструментів для розширення обсягу переданих даних і методів стеганографії). За наявності широкої обізнаності в предметній галузі, що вивчається, і достатньої технічної бази, фахівець у галузі комп'ютерної безпеки може використовувати широкий спектр технічних пристроїв і тактик для вилучення інформації з необхідного джерела шляхом використання прихованих каналів (наприклад, за допомогою електромагнітного випромінювання, використання загальнодоступних мереж Wi-Fi, мобільних мереж тощо).

В даний час, на відміну від описів і методів роботи прихованих каналів у своєму первісному значенні, приховані канали також є такою, що швидко розвивається і широко технологічно розвиненою областю загроз, в яку можуть входити такі можливості:

– створення глобальних бот-мереж для масованої атаки на мережеві

ресурси будь-якого підприємства чи компанії;

– передати порушником команди для виконання захопленим агентом;

– прихованої передачі ключів шифрування;

– передачі шкідливих програм на захоплені зловмисником пристрої та інформаційні системи.

Приховані канали часто використовуються спільно з різними методами стеганографії, що дозволяють передати додаткову інформацію в будь-якому контейнері (зображенні, документі, аудіо- або відео-запису).

Незважаючи на те, що система може бути наскільки можливо акуратно спроектована і глибоко проаналізована на кожному етапі проектування, уникнути можливості реалізації в ній прихованого каналу неможливо. Приховані канали можуть бути реалізовані в будь-якій області функціонування програми — найчастіше це ті області , на які найменше звертає увагу кінцевий користувач. Отже, прихований канал може залишатися невиявленим досить тривалий час.

Спираючись на базові документи в галузі комп'ютерної безпеки [4], приховані канали прийнято розділяти на дві основні категорії:

– канали пам'яті;

– канали часу.

У першому випадку передача інформації відбувається за допомогою модифікації об'єкта в деякій гаданій області пам'яті — наприклад, на жорсткому диску, карті пам'яті, флеш-накопичувачі з метою подальшого вилучення інформації шляхом зчитування та обробки цих змін. У другому випадку інформація передається шляхом модуляції деякого змінного в часі процесу, який згодом приймаючий суб'єкт буде здатний демодулювати. Справедливо вважатиме, що за порівняно однакової складності реалізації прихований канал за часом виявити складніше, ніж прихований канал по пам'яті, оскільки складно визначити, чи є процес у часі випадковим чи модулюється на боці деякого суб'єкта, на відміну від пам'яті, яку можна якимось або зберегти або скопіювати для подальшого , більш глибокого аналізу.

1.3.1 Приклади прихованих каналів за пам’яттю та часом

Наведемо найпростіші приклади прихованих каналів для кожної категорії: розглянемо деяку теоретичну систему, в якій представлені два різні рівні секретності: «Високий» і «Низький». Можлива передача інформації з низького рівня на високий, але передача у зворотному напрямку заборонена. Метою програмного агента у певній системі є передача інформації з високого рівня на низький.

У разі прихованого каналу пам'яті суб'єкт, що функціонує в середовищі з високим рівнем секретності, може виконувати налаштування та змінювати атрибути параметрів безпеки елементів файлової системи в середовищі з низьким рівнем секретності, таким чином, суб'єкт може закодувати деяке повідомлення у значеннях параметрів без небезпеки елементів файлової системи. Прикладом прихованого каналу за часом може послужити деяка синхронна БД, яка здатна одночасно працювати тільки з одним клієнтом — таким чином, ми можемо припустити для нашої системи, що будь-який суб'єкт з високого або низького рівня таємності може працювати з БД тому чи іншому рівні. Тоді суб'єкт, що знаходиться в області з високим рівнем секретності, може модулювати зайнятість БД певними діями через задані інтервали, а суб'єкт, що знаходиться в області з низьким рівнем секретності, здатний визначати зайнятість і доступність БД у певні проміжки часу, тим самим, здатний отримати закодоване повідомлення.

На закінчення, приховані канали можуть бути як односторонніми, так і двосторонніми. У першому випадку заздалегідь визначається, яка сторона виступатиме в ролі суб'єкта-відправника, а яка — у ролі суб'єкта-отримувача, і в процесі експлуатації каналу ці ролі не можуть бути змінені технічно ніяким чином: передача інформації можлива тільки в одну сторону . У другому випадку обидва суб'єкти виступають як у ролі відправника, так і в ролі одержувача — таким чином, передача інформації можлива взаємно в обидві сторони.

1.3.2 Приклади застосування прихованих каналів

Аналітичні звіти великих антивірусних компаній показують [5], що

приховані канали також часто використовуються при проведенні атак різної складності на всілякі цивільні та державні ресурси різного ступеня важливості в сукупності з різними типами command-and-control серверів для передачі команд на віддалені пристрої, які були схильні до атаки. Враховуючи все більше покриття та доступність різних класів пристроїв, таких як мережеве обладнання, IoT-пристрої, мікрокомп'ютери та мобільні пристрої, слід передбачати збереження та зміцнення позицій описаних методів та аналогічних технік у найближчому майбутньому [6] .

Як приклади використання прихованих каналів на практиці можуть послужити документи, пов'язані з великими загрозами — Advanced Persistent Threat (APT). За найчастіше першим етапом атаки для багатьох APT стає розсилання шкідливих Microsoft Office документів, що містять у собі шкідливий виконуваний код у вигляді макросів і powershell -скриптів, що дозволяють організувати подальший процес взаємодії між command-and-control сервером зловмисника і машиною, що атакується [7; 8]. Для приховування факту такої взаємодії також використовуються всілякі приховані канали передачі даних, що маскують як трафік, так і самі команди під легітимні дії користувача.

У рамках цього розділу була представлена актуальна інформація про стеганографічні методи і приховані канали, їх типи і сценарії використання на момент написання роботи. Приклади застосування прихованих каналів, наведені в цьому розділі, будуть використовуватися надалі як можливі приклади сценаріїв та моделей використання для реалізованих методів та об'єкта аналізу отриманого фреймворку. Також дані приклади, звіти та досліджені роботи показують, що приховані канали в цілому можуть бути організовані за допомогою різного набору технологій та механізмів, спільно з різними методами, що допомагають розширити семантику повідомлень, що передаються, або їх обсяг.

Крім того, аналіз досліджень та звітів показав, що на момент написання роботи не було відомо реальних прикладів використання нейронних мереж як прихованих каналів, command-and-control серверів та інших допоміжних інструментів.

РОЗДІЛ 2. РЕАЛІЗАЦІЯ СТЕГАНОГРАФІЧНИХ МЕТОДІВ ЯК ПРИХОВАНИХ КАНАЛІВ

2.1 Загальний підхід до реалізації методів прихованих каналів

У всіх зазначених нижче методах для реалізації була обрана мова програмування python , оскільки вона є на даний момент стандартом у галузі МН та ШНМ. Більше того, python є однією з найпопулярніших і відомих мов програмування у світі — маючи простий синтаксис і великий набір універсальних і доступних бібліотек, є легко підтримуваним і швидким як для розробки прототипів, так і для кінцевих продуктів. Проекти, написані мовою python, легко можуть підтримуватися вільним співтовариством робітників на одній з доступних платформ, таких як GitHub, GitLab та інших, що дозволяє залучити інтерес інших розробників до реалізованих методів та фреймворку, їх поліпшення та модифікації.

Кожна з наведених нижче реалізацій методів є модулем кінцевого розробленого фреймворку - реалізовані методи ізольовані за допомогою окремих docker -контейнерів і взаємодіє з фреймворком за допомогою черги завдань. Додатково , кожна з реалізацій підтримує два необхідні внутрішні API- методи: «encode» і «decode», дозволяють кодувати і декодувати секретне повідомлення, відповідно. Кожен із цих методів може приймати на вхід секретне повідомлення, зображення (обкладинку або секрет), файл та будь-який інший необхідний набір параметрів, який потрібний для конкретного методу (існуючого або доданого в майбутньому в рамках взаємодії з фреймворком). Більш детально схема взаємодії з фрейм ворком, його структура та архітектурні особливості описані в наступному розділі даної роботи.

Навчання моделей у рамках реалізацій проводилося на хмарному сервері за допомогою високопродуктивних GPU та технології CUDA для ефективних обчислень у рамках навчання. Середовищем для розробки реалізацій методів та допоміжних скриптів тренування моделей було обрано Conda та Jupyter Notebook, які також є стандартним набором для сучасних розробок у галузі МН

та нейронних мереж. Conda дозволяє уникнути конфліктів бібліотек і модулів при великому наборі допоміжних інструментів, а Jupyter Notebook надає зручний інтерактивний веб-інтерфейс розробки та візуалізації мовою python .

Для всіх реалізованих методів справедливі такі характеристики:

— процес навчання моделей для використання з фреймворком виконується один раз при ініціалізації спілкування між сторонами. Модель, що взаємодіє з будь-яким з методів, здатна працювати з новими, невідомими до цього даними того ж типу, на яких була навчена: у випадку методу Deep Steganography, можливо передавати секретні зображення та використовувати зображення-обкладинки, які не брали участь у наборі для навчання. Створення нової моделі (запуск нового процесу навчання) необхідно тільки в тому випадку, якщо модель була розкрита та отримана зловмисником, який має обчислювальні ресурси та знання для того, щоб розкрити вміст контейнера, або у випадку, коли потрібно змінити якісь критичні параметри системи (алгоритма, архітектури), пов'язані з нейронними мережами та їх властивостями та параметрами;

— навчена модель є найбільш критичною частиною системи - вона містить у собі всі необхідні ваги (значення і коефіцієнти) для того, щоб приховати або відновити секрет. За наявності ідентичної моделі та знання архітектури методу зловмисник зможе приховувати та розкривати дані аналогічно легітимним учасникам системи, оскільки модель у даному випадку виконує роль деякого «секретного ключа», або секретного алгоритму, за яким ховаються та розкриваються дані. Однак, отримати будь-яку іншу критичну інформацію, таку як попередні закодовані зображення або вихідні зображення, використані для тренування, зло навмисник не зможе;

— модель є файлом або набором файлів, які можуть бути замінені або оновлені за допомогою механізму звичайної заміни файлів на носії (у разі, якщо параметри мереж, обробки даних та архітектура не були змінені). Не потрібно внесення змін до вихідного коду реалізації або будь-якого іншого доступного методу;

— для виконання кодування або декодування інформації не потрібна велика обчислювальна потужність або наявність високопродуктивного GPU —

після завершення процесу навчання та перевірки коректності та достовірності даних, навчена модель може бути використана на будь-якому пристрої зі звичайним типом процесорів та середніми обчислювальними потужностями;

— отримане зображення-контейнер не містить класичних стеганографічних методів, таких як вкладення файлів у медіа-архів, додавання прихованих колірних шарів або каналів, додавання нових кадрів або інших додаткових методів приховування інформації - контейнер є звичайним графічним зображення формату .png (у разі генерації через взаємодію з фреймворком);

— розмір отриманого контейнера не розкриває факт наявності в ньому будь-якої прихованої інформації — розмір зображення-контейнера, що генерується в процесі використання одного з наведених нижче методів, відповідає розміру будь-якого іншого зображення такої ж кольоровості, формату та таких самих розмірів (висоти та ширини у пікселях).

Процес та етапи розробки реалізацій методів у загальному випадку виглядають так.

1. Підготовка даних для навчання - даний етап залежить від конкретної реалізації або результату, який потрібно отримати в процесі тренування. Так, наприклад, вхідні дані для навчання попередньо необхідно обробити - кожне зображення необхідно привести до певного розміру, що відповідає вхідним шарам мережі, у разі нестачі даних потрібно вдатися до процесу аугментації даних для збільшення вибірки шляхом додавання додаткових шумів, трансформацій зображення (поворотів, відображень по горизонталі та вертикалі, випадкової обрізки областей зображення, колірної корекції та інших), та додавання своїх розмічених даних за потреби.

2. Реалізація архітектури нейронних мереж, що використовуються, — даний етап ґрунтується на значеннях та інформації, наведеній у конкретній науковій роботі, пов'язаній з одним з методів (Deep Steganography, HiDDeN) або на власних гіпотезах і передбачуваних поліпшеннях (розширення методу Deep Steganography). Також на даному етапі враховуються особливості підготовлених даних для навчання — розмір згортки, кількість шарів та інші параметри .

3. Підбір параметрів мереж - даний етап ґрунтується як на конкретних значеннях, що використовуються в наведених роботах, так і на специфічних значеннях, що залежать від обчислювальних ресурсів сервера і достатнього результату (розмір партії даних (batch) в рамках навчання, максимальний розмір і форма зображень по всіх вимірах (shape), коефіцієнт швидкості навчання (learning rate), кількість епох тренування (epochs), число ітерацій (iterations) та інші)

4. Навчання або тренування - даний етап дозволяє навчити модель даних, отриманих на попередніх трьох кроках: набір даних, отриманий на кроці 1; підготовлена архітектура нейронних мереж, отримана на кроці 2; необхідний набір параметрів тренування, отриманий або обчислений на кроці 3. Результатом цього кроку є навчена модель, яку надалі можна використовувати для приховування та відновлення секретних повідомлень у деяких контейнерах без повторного навчання

Обчислення метрик і аналіз результатів — на даному етапі, маючи готову навчену модель, необхідно переконатися в коректності одержуваних даних: потрібно перевірити, залежно від методу, що дані, що розкриваються, відповідають прихованим даним (у разі зображень — коефіцієнт схожості секретного і розкривного зображення, у разі бітової послідовності — кількість помилок на кожну бітову послідовність, та інші відповідні типу даних метрики).

2.2 Метод Deep Steganography

2.2.1 Опис методу

Вперше був вперше описаний у роботі [9]. Даний метод дозволяє приховати секретне зображення в контейнері, використовуючи для цього саме два об'єкти: секретне зображення, яке має бути розкрите одержувачем контейнера при отриманні, і обкладинку, яке використовується як база для генерації зображення-контейнера. Обкладинка є видимою частиною контейнера і не є частиною інформації, що приховується.

Схема приховування та розкриття секретного зображення складається з

трьох нейронних мереж, які навчаються одночасно як одна єдина мережа, але працюють незалежно один від одного і можуть бути розділені між відправником та одержувачем. Даний метод дозволяє приховувати секретне зображення всередині зображення-контейнера за допомогою підготовчої (prep network) і приховувальної (hiding network) мереж, які знаходяться на стороні відправника, і розкривати секретне зображення на стороні одержувача за допомогою розкривної (reveal network) мережі. Загалом, для двостороннього прихованого каналу всі учасники спілкування повинні мати доступ до всіх мереж методу - підготовчої, приховує та розкриває.

Графічна схема методу зображена на рис. 2.1.

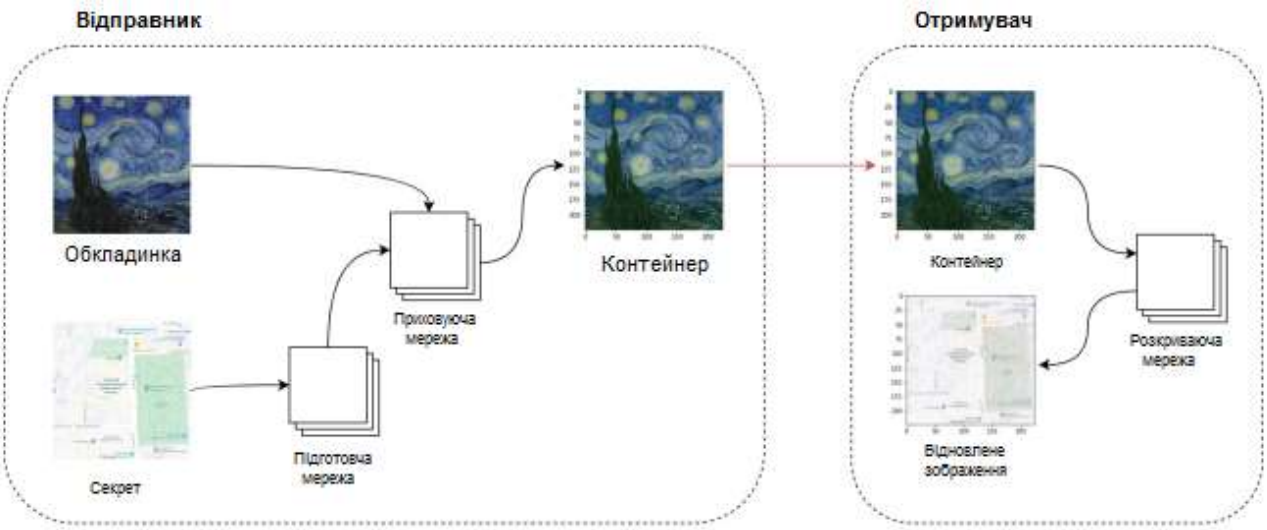

Рисунок 2.1 – Схема методу Deep Steganography

Помилка першого роду (різниця між видимим зображенням-обкладинкою та зображенням-контейнером) поширюється назад тільки через перші дві мережі (які підготовлюють та приховують), що забезпечує максимальну схожість між обкладинкою і контейнером, що передається. Помилка другого роду (різниця між секретом і кінцевим відновленим зображенням) поширюється через всі три мережі, для забезпечення максимальної подібності та якості розкриття секретного зображення на стороні одержувача.

Кожна з мереж, що використовуються у методі Deep Steganography, виконує свою задачу:

— підготовча мережа (prep network) готує секретне повідомлення у вигляді графічного кольорового або монохромного зображення для того, щоб помістити його в контейнер. Дана мережа виконує два важливі завдання: в першу чергу, якщо секретне зображення менше, ніж зображення-обкладинка, підготовча мережа збільшує розміри секретного зображення до розмірів зображення -обкладинки, маючи все зображення під обкладинкою відповідно; потім, підготовча мережа перетворює всі пікселі вихідного секретного зображення в більш придатні для кодування ознаки (features) - такі, як, наприклад, межі і краї між об'єктами і різними кольорами зображення, які не залежать від кольору та інших якостей;

— приховувальна мережа (hiding network) приймає на вхід два зображення: секретне зображення, отримане в результаті перетворень підготовчої мережі та зображення-обкладинку. На виході мережа, що приховує, створює зображення-контейнер, що несе в собі секретне зображення. На даному етапі отримане зображення-контейнер готове до передачі одержувачу;

— розкриваюча мережа (reveal network*)* приймає на вхід тільки зображення-контейнер (без знання зображення-секрету або зображення-обкладинки, відповідно), видаляє зображення-обкладинку і відтворює вихідне секретне зображення, яке було приховано за допомогою мережі, що приховує

2.2.2 Особливості реалізації

Як базова бібліотека для розробки була використана бібліотека TensorFlow. Для підготовки та навчання моделі в якості тренувального набору даних випадковим чином було обрано 10 різних категорій зображень із набору зображень ImageNet. Кожне зображення було приведено до розміру 224×224 пікселів для навчання.

2.2.3 Результати

Як основна метрика для визначення коректності отриманих результатів і якості навчання моделі був обраний індекс структурної подібності (structure similarity — SSIM). Цей індекс дозволяє визначити структурну схожість між

двома зображеннями методом повного зіставлення — або, інакше, даний метод проводить вимір якості на основі вихідного (секретного) зображення. Значення, близьке до 1.0 (або 100% в інтерпретації) досягається тільки при повній автентичності двох зразків. Приватні результати порівняння отриманих у результаті навчання прикладів та середні значення представлені у табл. 2.1.

Таблиця 2.1 – Результати методу Deep Steganography

Вихідні розміри зображень	**Модифіковані розміри зображень**	**Індекс схожості обкладинки та контейнерів (*SSIM*)**	**Індекс схожості секретного зображення та відновленого (*SSIM*)**
934x934 (обкладинка, картина ван Гога) 1338x1338 (секрет, зображення карти)	224x224 (кольорові)	≈ **92%** **(0.9176)**	≈ 75% (0.7479)
1370x1370 (обкладинка, фотографія людини) 988x988 (секрет, фотографія іншої людини)	224x224 (кольорові)	≈ **88%** **(0.8841)**	≈ 84% (0.8373)
934x934 (обкладинка, картина ван Гога) 1370x1370 (секрет, фотографія людини)	224x224 (кольорові)	≈ 91% (0.9147)	≈ **87%** **(0.8714)**
Середнє (10 різних пар)	224x224 (кольорові)	≈ 90% (0.8975)	≈ 81% (0.8054)

Виходячи з отриманих результатів, середнє значення схожості секретного зображення і відновленого досягає 81%, індекс схожості обкладинки і контейнера досягає 90% при розмірі вихідних зображень 224x224, що є хорошим показником для зображень таких розмірів, з урахуванням використання тільки 10 класів набору ImageNet (з 1000 можливих). Дані значення показують, що метод може бути застосований у практичному сенсі навіть за мінімальних

обсягів тренувальної вибірки. Варто припускати, що при використанні більшої кількості класів, зміні розмірності, використанні більш потужних обчислювальних кластерів та інших модифікаціях вхідних даних та середовища значення схожості за індексом SSIM можуть досягати великих значень.

Приклад перетворень, відповідних першому рядку таблиці, наведено на рис. 2.2.

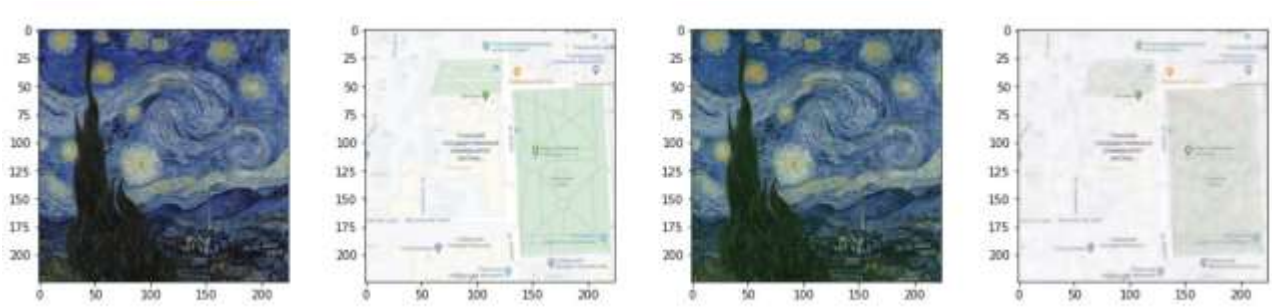

Рисунок 2.2 – Приклад результатів перетворень методу Deep Steganography

На рис. 2.2 перше зображення є обкладинкою, друге зображення є секретом, який потрібно приховати. Третє зображення є контейнером - обкладинкою, що містить секретне зображення. Четверте зображення — відновлене на стороні отримувача секретне зображення із контейнера.

Для порівняння та наочної демонстрації результатів обчислення індексу, на рис. 2.3 наведено приклад трьох зображень — оригінального зображення та двох зображень з накладенням додаткових фільтрів. Незважаючи на те, що для людського ока зображення є майже ідентичними, індекс схожості другого зображення (з накладенням шуму) дорівнює 0.15; для третього, де була змінена лише яскравість зображення, індекс схожості дорівнює 0.85, що приблизно дорівнює середньому результату порівняння оригінального секретного зображення і розкритого одержувачем через розкривну мережу наведеного вище методу Deep Steganography.

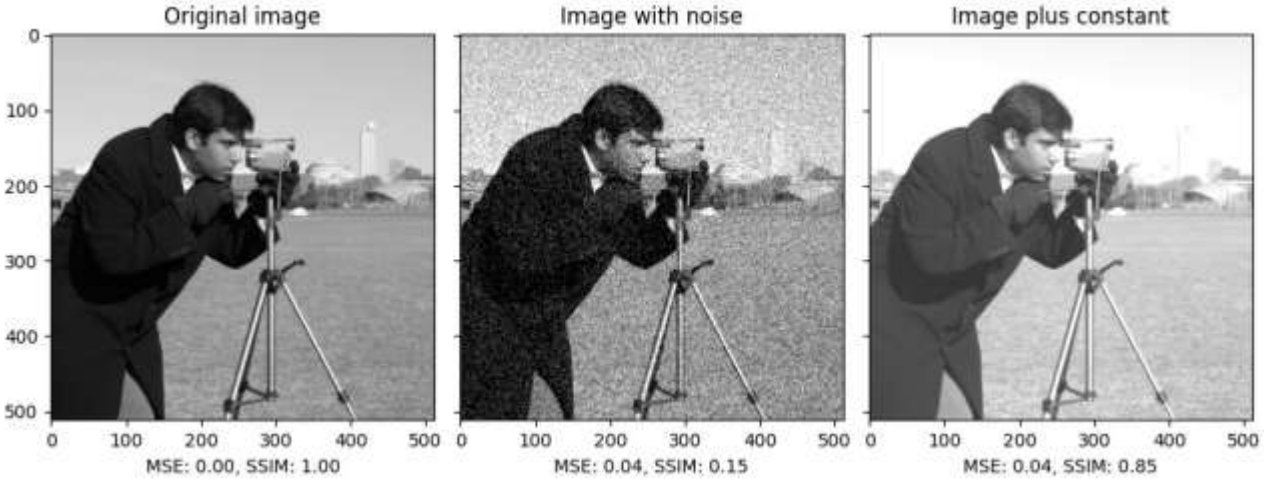

Рисунок 2.3 – Приклад SSIM- індексу

Приклад демонструє порівняння оригінального зображення із самим собою (Original image), з результатом накладання шуму (Image with noise), з результатом освітлення (Image plus constant).

2.3 Розширення методу Deep Steganography

2.3.1 Опис запропонованого розширення методу

В результаті реалізації методу Deep Steganography було запропоновано та розроблено розширення методу у вигляді додавання додаткових прихованих шарів у закодоване зображення.

Основний принцип роботи методу і роль мереж залишаються без змін, за винятком того факту, що замість однієї підготовчої та однієї розкривної мережі використовуються n -підготовчих і n -розкривних мереж відповідно - модифікується архітектура самої мережі. Кожна з мереж дозволяє закодувати в контейнері та розкрити n -зображень. Число n визначається допустимим рівнем спотворень у контейнері та потребою у необхідній кількості секретних зображень, яке потрібно закодувати та передати.

Приховувальна мережа приймає на вхід одне зображення-обкладинку і n - число секретних зображень. Розкривні мережі здатні розкодувати набір секретних зображень (кожна мережа розкодує свій шар, що містить секрет).

Це розширення дозволяє передати більший обсяг прихованої інформації за

один сеанс передачі (при використанні одного контейнера) порівняно з класичним методом. Графічна схема методу наведена на рис. 2.4.

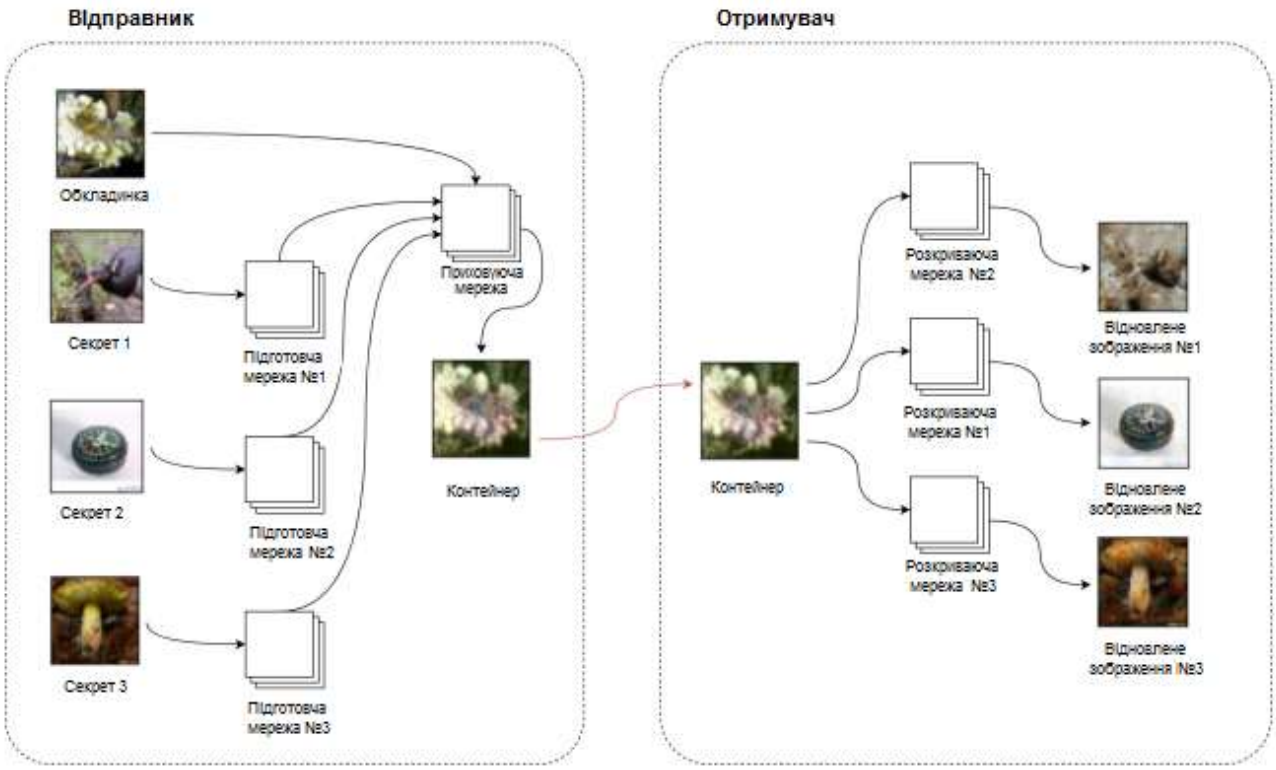

Рисунок 2.4 – Схема розширення методу Deep Steganography

2.3.2 Особливості реалізації

Як базова бібліотека для розробки була використана бібліотека TensorFlow. Для підготовки та тренування моделі як тренувальний набор даних був обраний датасет Tiny ImageNet, який є зменшеною версією основного датасету ImageNet. Датасет Tiny ImageNet містить 200 категорій зображень, по 500 зображень розміру 64x64 пікселів у кожному. Сумарно Tiny ImageNet містить 100 000 зображень.

2.3.3 Результати

Виходячи з того, що основний принцип роботи і підхід залишилися незмінними щодо основного методу Deep Steganography, як основна метрика також був обраний індекс структурної подібності SSIM. Набір у цій таблиці являє собою чотири зображення: три секретні зображення та одне зображення-

обкладинку. Результати представлені у табл. 2.2.

Таблиця 2.2 – Результати запропонованого розширення методу Deep Steganography

Розміри тестових зображень	**Індекс схожості обкладинки та контейнера *(SSIM)***	**Індекс схожості секрету (1) і відновленого зображення (1) *(SSIM)***	**Індекс схожості секрету (2) і відновленого зображення (2) *(SSIM)***	**Індекс схожості секрету (3) і відновленого зображення (3) *(SSIM)***
64x64 (набір №1)	≈ 70%(0.7008)	≈ 56% (0.5554)	≈ 81% (0.8133)	≈ 81% (0.8143)
64x64 (набір №2)	≈ 70%(0.7031)	≈ **71% (0.7133)**	≈ 71%(0.7054)	≈ 76% (0.7632)
64x64 (набір №3)	≈ 59% (0.5929)	≈ 55% (0.5482)	≈ **85% (0.8546)**	≈ **88% (0.8798)**
64x64 (набір №4)	≈ 58% (0.5834)	≈ 49% (0.4900)	≈ 80% (0.7974)	≈85% (0.8549)
64x64 (набір №5)	≈ 64%(0.6364)	≈ 45% (0.4505)	≈ 68% (0.6845)	≈ 80% (0.8039)
64x64 (набір №6)	≈ **78% (0.7815)**	≈ 49% (0.4851)	≈ 72%(0.7164)	≈ 86% (0.8570)
Середнє (10 різних наборів)	≈ 66% (0.6574)	≈ 53% (0.5312)	≈ 75% (0.7503)	≈ 84% (0.8378)

Виходячи з отриманих результатів, можна зробити висновок, що останній застосуваний (прихований) шар у зображенні-контейнері вдається відновити з найменшими втратами — індекс структурної подібності ~ 83%.

Зворотньо, перший прихований шар (найглибший щодо розкриття контейнера) — відновлюється найгірше (індекс структурної подібності ~ 54%).

Приклади перетворень кількох наборів представлені на рис. 2.5.

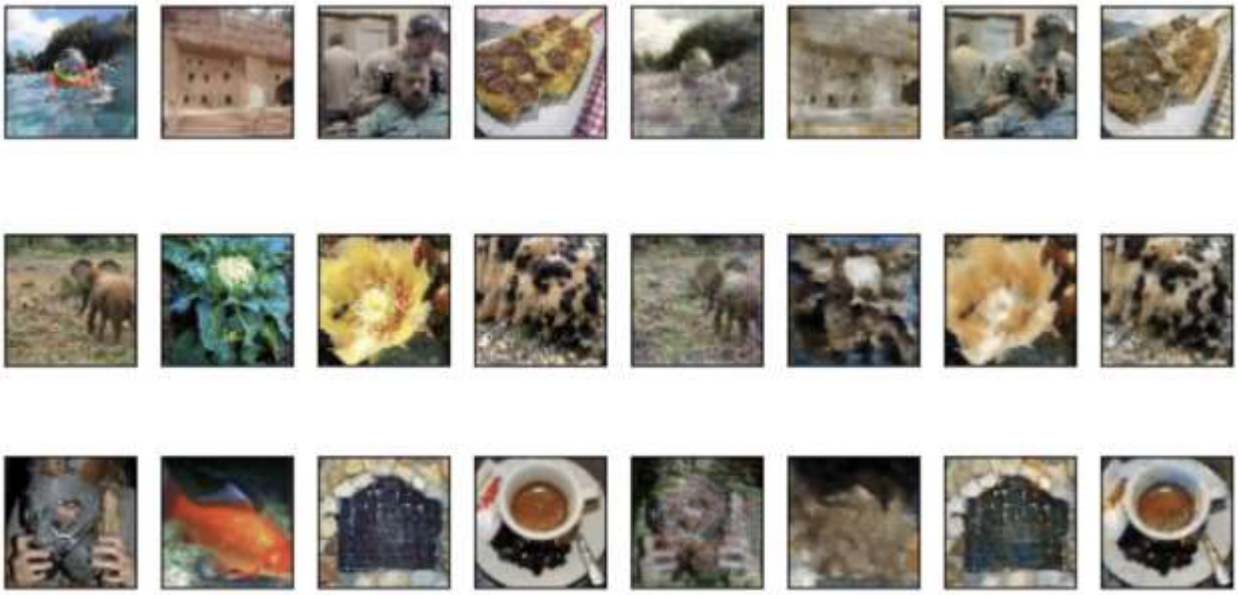

Рисунок 2.5 – Приклад роботи запропонованого розширення методу Deep Steganography

На рис. 2.5 кожен ряд відповідає одному повному циклу роботи: перше зображення є обкладинкою, наступні три секретні зображення. П'яте зображення - отриманий в результаті роботи приховує мережі контейнер, що містить в собі три секретні зображення та обкладинку. Останні три зображення - відновлені з контейнера секрети.

2.4 Метод HiDDen

2.4.1 Опис методу

Метод HiDDeN був вперше описаний у роботі [10]. Загальна схема і структура схожа з методом Deep Steganography, за винятком того, що даний метод дозволяє кодувати бінарні послідовності в зображеннях. Приховуюча мережа отримує на вхід секретне повідомлення і зображення-обкладинку, і генерує з цих даних зображення, що містить секретне повідомлення. Зображення-контейнер, що містить у собі секретне повідомлення, максимально схоже на оригінальне зображення-обкладинку. Мережі, що приховують і розкривають, тренуються спільно, але працюють незалежно одна від одної.

2.4.2 Особливості реалізації

Як бібліотека для реалізації була використана бібліотека TensorFlow та Keras API. Для підготовки та навчання моделі як тренувального набору даних також було обрано датасет Tiny ImageNet.

2.4.3 Результати

Виходячи з того, що метод HiDDeN дозволяє передавати бінарні послідовності (такі як частини тексту, секретні ключі або паролі, частини аудіо або зображень), як основний метод тестування коректності результатів застосовується середня кількість помилок (середня кількість повідомлень, що містять помилку до числа всіх повідомлень і середнє число помилкових біт на повідомлення до всіх переданих біт). Результати представлені в табл. 2.3. Виходячи з отриманих результатів, верхня можлива межа довжини повідомлення, що передається, становить не більше 32 біт (з прийнятним рівнем помилки нижче 1%). З урахуванням можливого використання алгоритмів кодування та стиснення інформації, дана межа потенційно може бути розширена. Приклади перетворень з допомогою методу HiDDeN представлені на рис. 2.6.

Таблиця 2.3 – Результати методу HiDDeN

Реалізація	**Довжина повідомлення**	**Розмір тестових зображень**	**Число тестових зображень**	**Середнє число помилок**	**Середня кількість помилок на повідомлення**	**bpp (bits-per-pixel)**
Оригінальна робота HiDDeN	52 біта	16x16 (чорно білі)	Не вказано	Не вказано	0.00001%	0.203
Дана робота	32 біта	64x64 (кольорові)	10.000	0.54%	0.0275%	0.007
Дана робота	64 біта	64x64 (кольорові)	10.000	90.7%	3.7125%	0.015625

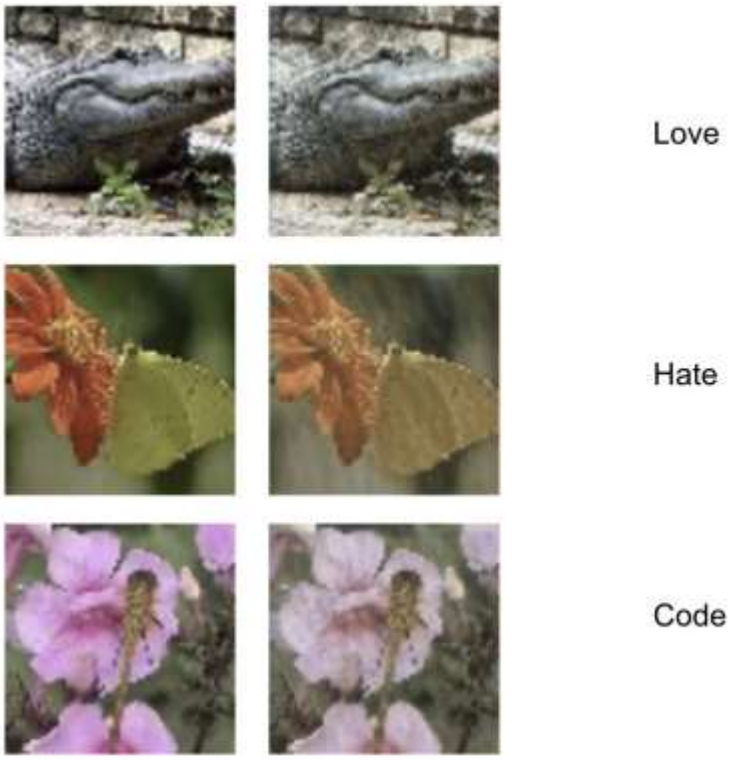

Рисунок 2.6 – Приклад повного циклу роботи та результатів перетворень методу HiDDeN

На рис. 2.6 перший стовпець - оригінальне зображення-обкладинка, яке використовується для приховування в ньому певного слова у вигляді бітової послідовності (Love, Hate) або "Code" з правого стовпця). Другий стовпець — це зображення, яке містить секретне повідомлення. Одержувач розкодує цей контейнер і отримує послідовність бітів, яку він може інтерпретувати залежно від завдання (текст чи інший тип інформації).

У рамках розробки дослідницького фреймворку для реалізації та аналізу прихованих каналів було розглянуто та реалізовано два основні існуючі на даний момент методи приховування інформації за допомогою нейронних мереж: Deep Steganography [9] та HiDDeN [10]. Основною відмінністю даних методів є можливий тип даних, що передаються: у випадку Deep Steganography секретним повідомленням є зображення; у випадку HiDDeN ним може бути будь-яка бітова послідовність.

Дана відмінність обумовлена відповідною архітектурою ШНМ і алгоритмом додавання секретного повідомлення до контейнера: у випадку

зображення (Deep Steganography), точність не є критичною властивістю — зображення може бути відновлене з деяким відсотком артефактів, мало помітних для людського ока. Тим не менш, в загальному випадку секретне зображення відновлюється максимально схожим на вихідне секретне зображення, що передається; у випадку бітової послідовності (HiDDeN) точність відновлення інформації відіграє критичну роль (наприклад, в рамках передачі секретних ключів і паролів), тому, обсяг переданої інформації в загальному випадку менший, але секретне повідомлення відновлюється ідентично відправленому .

Також у рамках реалізації методів було запропоновано та розроблено розширення методу Deep Steganography у вигляді додавання додаткових шарів, що містять секретні зображення — таким чином, в одному зображенні-контейнері можливо передавати більше, ніж одне зображення-секрет (три секретні зображення для конкретної реалізації в рамках фреймворку).

РОЗДІЛ 3. РОЗРОБКА ФРЕЙМВОРКУ ДЛЯ АНАЛІЗУ ПРИХОВАНИХ КАНАЛІВ

3.1 Дизайн фреймворку

Відсутність існуючих рішень для застосування методів стеганографії та прихованих каналів за допомогою ШНМ та актуальність наведених методів стали основними причинами розробки універсального дослідницького фреймворку, що дозволяє експлуатувати наведені раніше методи прихованих каналів та стеганографії для реальних практичних завдань та аналізу результатів. Розроблений програмний засіб дозволяє використовувати будь-який із зазначених методів у зручному для користувача форматі взаємодії з веб-додатком.

Практична застосовність і доступність інтерфейсів реалізованих механізмів дозволяють швидко і ефективно аналізувати якість передачі повідомлень, максимальний обсяг інформації, що передається, і швидкість роботи повного циклу приховування і розкриття даних в рамках реальних завдань.

При розробці фреймворку враховувалися різні аспекти та можливі сценарії використання методів користувачами, а також необхідні умови та залежності системного середовища для запуску — у результаті фреймворк має повністю модульну архітектуру, пов'язану між собою різними API та мережевими інтерфейсами. Модулі фреймворку мають стійкість до відмов у разі недоступності сервісів, можуть бути розділені між комп'ютерами або серверами, а також можуть легко масштабуватися і збільшувати можливий обсяг оброблюваних даних за рахунок запуску нових екземплярів сервісів у «гарячому» режимі, без перезапуску основних компонентів системи.

Більш докладно, до ключових особливостей архітектури та реалізації розробленого фреймворку належить таке.

Кросплатформенність. Завдяки системі docker -контейнерів і описаному сценарію складання та запуску сервісів за допомогою інструмента Docker

Compose, фреймворк може бути зібраний і запущений на будь-якій системі, яка підтримує Docker. На даний момент, Docker підтримується у всіх відомих сучасних операційних системах (таких як Linux, Windows, MacOS). У разі недоступності складання та запуску через Docker, доступний класичний спосіб складання та запуску фреймворку самостійно, шляхом встановлення необхідних дистрибутивів PostgreSQL, RabbitMQ, Python та інших на персональний комп'ютер або сервер — у цьому випадку достатньо експортувати змінні середовища оточення в середовище запуску веб-сервера та допоміжних модулів.

Відмовостійкість. Сервіси та реалізовані модулі влаштовані таким чином, щоб забезпечувати максимальну відмовостійкість за будь-якого сценарію використання — недоступність одного з сервісів або файлових сховищ, відсутність можливості з'єднання з базою даних або брокером повідомлень, критична помилка системи:

— у випадку мережевої недоступності одного та більше сервісів, всі пов'язані з ним сервіси перейдуть у режим очікування доступності зв'язку з недоступними модулями шляхом опитування сервісу кожні 10 секунд;

— у випадку, якщо помилка при запуску є критичною або в процесі роботи виникає необроблений виняток, буде здійснено автоматичний перезапуск сервісу та автоматичне під’єднання його до всіх пов'язаних ресурсів.

Також у режимі реального часу проводиться моніторинг стану модулів та їх доступності завдяки Healthcheck механізму сервісів Docker - кожні 30 секунд всі сервіси перевіряються на предмет мережевої доступності, коректної роботи кінцевих ендпоінтів та доступності ресурсів для читання та запису в профілактичних цілях.

Масштабованість. Додавання додаткових ресурсів (наприклад, запуск кількох примірників сервісу Deep Steganography для прискорення обробки даних) можливо в режимі реального часу без перезапуску веб-сервера, БД або черги повідомлень. Запущений новий екземпляр буде автоматично під’єднаний до спільної черги завдань. Завдання виконуватимуться даним модулем у порядку загальної черги виконання.

Децентралізованість. Фреймворк підтримує розподілений запуск та

функціонування. Підтримується авто-розподіл навантаження між сервісами і вузлами, більше того, сервіси можуть бути запущені на різних комп'ютерах або серверах - таким чином, є можливість запустити за витратні за ресурсами модулі на високопродуктивному сервері і зв'язати їх з іншими сервісами, що знаходяться на іншому сервері або комп'ютер (у будь-якому сегменті мережі).

Незалежність від мов програмування та бібліотек для модулів, що реалізуються. Завдяки архітектурному поділу компонентів на контейнери та мікросервіси, немає залежності від конкретної мови програмування для розробки нових модулів приховування або розкриття інформації — достатньо підтримки черги повідомлень усередині модуля, реалізувати яку можна через будь-яку доступну бібліотеку конкретної мови програмування. Так, наприклад, нові методи, що реалізуються, можуть працювати на мовах Lua , R , Python або будь-яких інших, що підтримують можливості МН та ШНМ.

Стійкість до можливих архітектурних помилок або людського фактора. Завдяки мікросервісній архітектурі, на відміну від монолітної, з появою помилок як у логіці додатка, так і в кодовій частині, вони можуть бути легко локалізовані та виправлені без істотного впливу на решту функціоналу програми: так, наприклад, при виявленні помилок в одному з модулів, робота будь-якого іншого модуля не буде порушена — кодова частина різних модулів є окремим апаратом, що дозволяє виконати одну або кілька конкретних і заздалегідь визначених завдань. Кожна нова особливість або деталь фреймворку може бути імплементована в якомусь конкретному існуючому модулі, або новий модуль відповідного завдання може бути доданий до загального ланцюжка виконання.

Швидкість обробки даних. Навчені моделі з вагами та необхідні бібліотеки завантажуються в пам'ять автоматично при старті сервісів (при запуску фреймворку). При отриманні повідомлення від користувача створюється завдання, яке починає виконуватися миттєво, без тимчасових витрат на підготовку ресурсів, завантаження моделі та необхідних значень в оперативну пам'ять. Даний аспект фреймворку дозволяє прискорити процес приховування секрету та відновлення даних у кілька разів (щодо класичного програмного підходу у вигляді консольної утиліти або нативного додатку, що запускаються та

ініціалізуються тільки при старті обробки).

Простий механізм запуску та повного видалення. Завдяки опису сценарію в docker-compose.yml файлі та фіксованим інструкціям складання кожного сервісу через Dockerfile, запуск і складання фреймворку відбуваються автоматично. Знання певних версій використовуваних бібліотек або програмних рішень (таких як версія БД, брокера повідомлень, інтерпретатора конкретних реалізацій) не потрібне - дані про необхідні залежності зафіксовані в описаних сценаріях та інструкціях складання. Також, завдяки цьому, доступне просте видалення всіх використовуваних ресурсів після використання - таблиці БД, повідомлення черг, збережені файли зберігаються як окремі розділи (volume) в рамках системи Docker, і можуть бути легко видалені як незалежно від самого контейнера, так і разом з ним при повному очищенні.

3.2 Компоненти та архітектура фреймворку

Схема компонентів та архітектура фреймворку представлена на рис. 3.1.

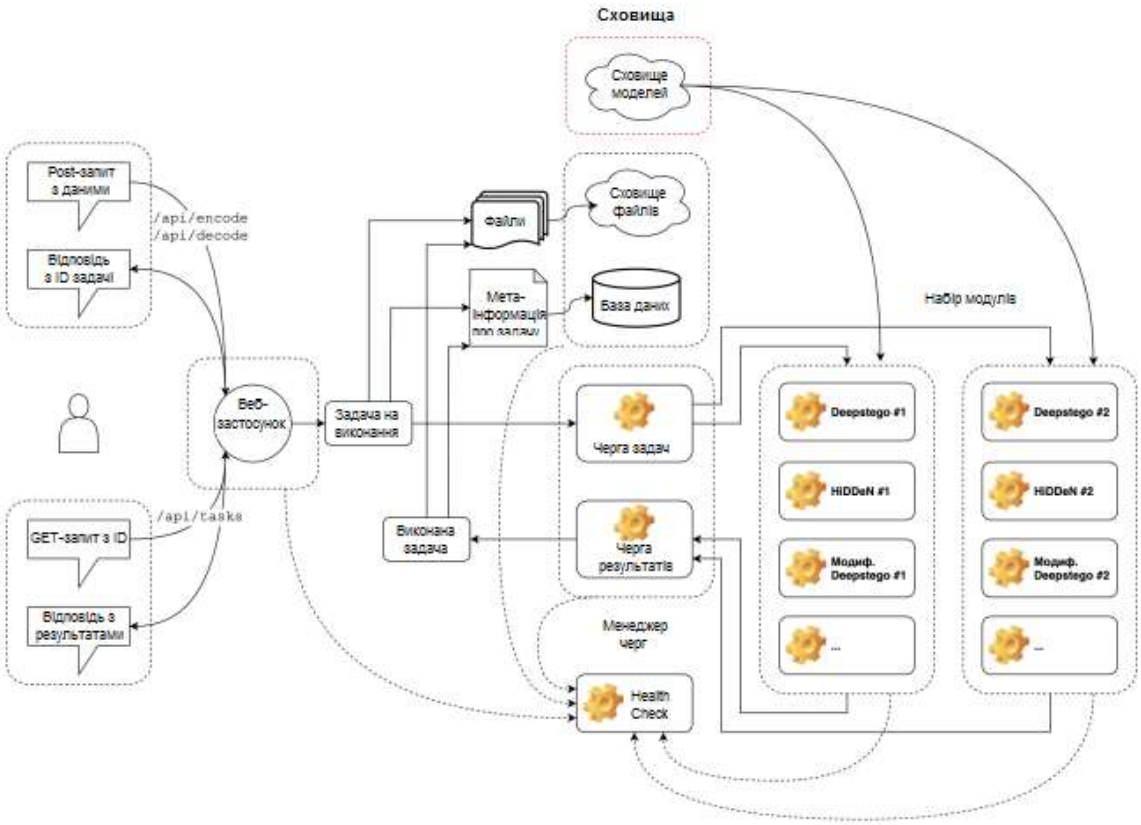

Рисунок 3.1 – Архітектура фреймворку

Схема взаємодії йде зліва направо від найзагальніших абстракцій (доступні для користувача методи REST API) до конкретних мікросервісів та модулів, що виконують поставлене завдання (кодування або декодування інформації).

Фреймворк був розроблений з урахуванням того, що в майбутньому, за подальших розробок і поліпшень, деякі нові експериментальні методи можуть займати значний час на обробку (особливо з урахуванням малопродуктивних систем). Таким чином, процес виконання конкретної задачі потенційно може займати від 30 секунд до 1 хвилини, можливо, і більше. Користувачеві не має сенсу чекати HTTP -відповідь від веб-додатку на свій запит весь цей час - натомість, архітектура фреймворку представлена у вигляді моделі виконуваних асинхронних завдань: користувач моментально отримує відповідь на свій HTTP-запит з описом, який містить всю коротку інформацію про завдання, яке було запущено, — час, статус виконання, відповідні дії та інша інформація.

Наступний набір компонентів та сервісів дозволяє забезпечити таку модель взаємодії.

1. Асинхронний веб-сервер на базі мови python та веб-фреймворку Tornado Web Server*:* даний компонент є основним у контексті взаємодії користувачів з фреймворком, доступними методами та іншими сервісами. Веб сервер обробляє запити від користувачів, створює завдання для виконання кодування та декодування за допомогою певних методів, а також забезпечує доступ до файлового сховища та БД. Завдяки асинхронності та системі фонового створення завдань, велика кількість користувачів може працювати з сервісом одночасно, без впливу на продуктивність системи та швидкість обробки даних.

2. БД PostgreSQL: забезпечує зберігання результатів виконання завдань — час запуску та завершення завдання, її статус («pending» — у процесі виконання, «success» — успішне виконання, «error» — помилка під час виконання завдання), та додаткові поля для отримання розгорнутої інформації про задачі (або помилці, у разі її виникнення).

3. Брокер повідомлень та менеджер черг RabbitMQ*:* у фреймворку реалізовано систему черг за допомогою брокера повідомлень. Таким чином, при

отриманні запиту від користувача на кодування або декодування інформації, користувачу немає необхідності чекати відповіді на запит доти, поки завдання не виконається - користувач відразу отримує ідентифікатор створеного завдання, черга повідомлень (черг завдань) в цей час займається обробкою завдання та її відправкою на один із доступних сервісів з методами.

4. Набір реалізованих модулів - Deep Steganography, розширення Deep Steganography, HiDDeN*:* кожен з реалізованих методів представляється у фреймворку у вигляді незалежних docker -контейнерів, пов'язаних з фреймворком системою черг. При отриманні завдання менеджер черг відправляє завдання на кодування або декодування на один з доступних у системі модулів. Потім, після завершення процесу кодування або декодування, модуль повертає результат на чергу результатів, яка потім обробляється менеджером черг, і необхідний результат зберігається в БД (мета-інформація) і файлове сховище.

5. Сховище моделей*:* критичний компонент системи, що містить у собі всі необхідні навчені моделі для кодування та декодування інформації через доступні модулі. Тільки реалізовані модулі мають доступ до читання до сховища моделей, жоден інший компонент системи не може модифікувати сховище моделей або файли в ньому для забезпечення безпеки критичних даних. Перед використанням та запуском фреймворку, всі необхідні моделі додаються у файлове сховище користувачами системи вручну.

6. Сховище файлів*:* компонент системи, що відповідає за зберігання файлів (секретних зображень, обкладинок, контейнерів та відновлених повідомлень). Також є критичним компонентом системи. При взаємодії користувачів з веб-сервером, всі необхідні та допоміжні файли записуються у файлове сховище під випадковими унікальними UUID ідентифікаторами відповідного завдання (кодування або декодування).

Кодова структура фреймворку прагне розумного перевикористання існуючих компонентів і укладання більшої частини логіки в загальні та зрозумілі абстракції та об'єкти. Так, наприклад, для зручності роботи із завданнями у тій структурі, в якій ми її описуємо, у рамках фреймворку реалізований окремий

модуль, який забезпечує управління завданнями, їх статусом, повідомленнями та іншими атрибутами.

На рис. 3.2 наведено найпростіший приклад взаємодії з модулем TaskItem в рамках інтерпретатора IPython — ініціалізація та створення завдання, та виставлення статусу завершення завдання як «успіх».

```
Python 3.9.1 (default, Apr 10 2024, 11:11:14)
Type 'copyright', 'credits' or 'license' for more information
IPython 7.19.0 -- An enhanced Interactive Python. Type '?'
   for help.

In [1]: from task import TaskItem

In [2]: task = TaskItem()

In [3]: print(task)
Task item, instance of the class 'TaskItem'. Values:
   'status'='pending', 'message'='',
   'task_id'='c9da5d6b-39a8-4323-8058-881c405b0423',
   'datetime_start'='2021-01-13 21:41:22.058229',
   'datetime_finish'='None'

In [4]: task.set_success("Additional message")

In [5]: print(task)
Task item, instance of the class 'TaskItem'. Values:
   'status'='success', 'message'='Additional message',
   'task_id'='c9da5d6b-39a8-4323-8058-881c405b0423',
   'datetime_start'='2021-01-13 21:41:22.058229',
   'datetime_finish'='2021-01-13 21:42:28.482675'
```

Рисунок 3.2 – Фрагмент коду функції взаємодії

Компоненти та методи, що відповідають за обробку запитів від користувача (handlers) мають наступну структуру у фреймворку (результат виконання команди «tree -charset=ascii -P '*.py' handlers» в Linux -середовищі) наведені на рис. 3.3.

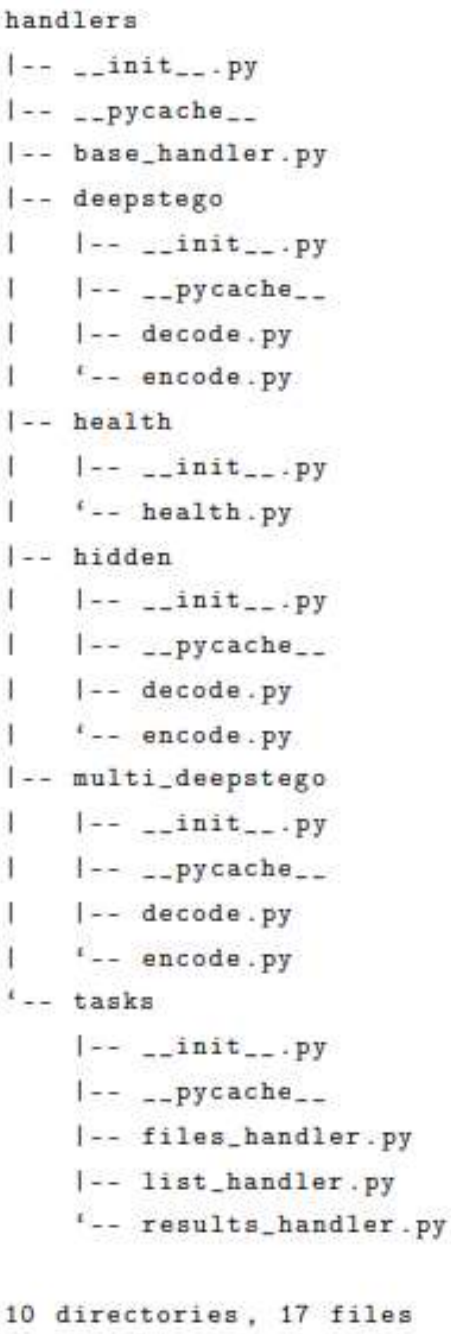

Рисунок 3.3 – Програмний код handlers

Модуль base_handler.py містить у собі всі абстрактні та перевикористовувані методи , від яких успадковуються всі дочірні обробники. Методи, представлені в модуль base_handler.py :

— set_ default_ headers, success, error — модифікація HTTP -відповіді залежно від результатів виконання завдання та даних, що надаються;

— upload_files, validate_params, validate_files — обробка та валідація вхідних параметрів та файлів;

— get_file_links, serve_file - надання доступу до файлів користувачам;

— post та method_ handler — базовий абстрактний обробник для всіх запитів, що несуть у собі інформацію для модулів кодування та декодування інформації.

Модуль base_handler.py дозволяє описувати (рис. 3.4) обробники будь-якого методу наступним чином (обробник методу «Deep Steganography», файл ../handlers/deepstego/encode.py):

```
from abc import ABC
from src.server.handlers.base_handler import BaseHandler
class EncodeDeepStegoHandler(BaseHandler, ABC):
    REQUIRED_FILES = ["secret", "cover"]
    REQUIRED_PARAMS = []
    METHOD = "deepstego"
    ACTION = "encode"
```

Рисунок 3.4 – Програмний код обробника будь-якого методу

Таким чином, будь-який користувач системи може максимально просто і доступно описати всі необхідні параметри власної імплементації будь-якого нового методу через змінні класу в обробнику:

— REQUIRED_FILES — визначає набір необхідних файлів, які будуть зібрані із запиту у процесі обробки (зображення, аудіо, відео, інші файли);

— REQUIRED_PARAMS — визначає набір необхідних параметрів, які будуть зібрані із запиту у процесі обробки (текст);

— METHOD - назва методу, що використовується для внутрішніх структур (черг , БД);

— ACTION - дія, що виконується даним обробником (кодування, декодування , або визначено користувачем).

Описаний вище підхід та його приклади використовуються також і в інших частинах фреймворку та його модулів — у системі HTTP -відповідей фреймворку, загальної черги завдань, взаємодії з БД та інших, що забезпечує низький поріг входу та зрозумілу структуру для більшості потенційних розробників на таких відкритих платформах, як GitHub та GitLab.

3.3 Сценарій використання фреймворку

Розглянемо повний процес приховування та відновлення секретного зображення на прикладі методу Deep Steganography передбачуваним користувачем. Наступні кроки є прикладом процесу приховування повідомлення за допомогою фреймворку — зворотний процес (розкриття повідомлення) відбувається аналогічно, за винятком відповідної зміни параметрів запитів (файлів, що передаються, і ідентифікаторів завдань).

1. Створення завдання: користувач надсилає на веб-сервер POST запит типу Multipart/form-data, що містить два файли-зображення у випадку кодування за допомогою методу Deep Steganography : зображення-секрет довільного розміру (secret-1.jpg у прикладі на рис. 3.5), яке необхідно приховати, та зображення-обкладинку довільного розміру (cover-1.jpg у прикладі на рис. 3.5), яке виступатиме у ролі видимої частини кінцевого контейнера. У разі декодування необхідно надіслати тільки зображення-контейнер. Конкретний запит кодування секретного зображення наведено нижче:

```
POST /api/encode/deepstego HTTP/1.1
Host: localhost:8888
Content-Length: 320
Content-Type: multipart/form-data;
   boundary=----WebKitFormBoundary7MA4YWxkTrZu0gW
----WebKitFormBoundary7MA4YWxkTrZu0gW
Content-Disposition: form-data; name="cover";
   filename="cover-1.jpg"
Content-Type: image/jpeg
(data)
----WebKitFormBoundary7MA4YWxkTrZu0gW
Content-Disposition: form-data; name="secret";
   filename="secret-1.jpg"
Content-Type: image/jpeg
(data)
----WebKitFormBoundary7MA4YWxkTrZu0gW
```

Рисунок 3.5 – Фрагмент коду запиту кодування

Веб-сервер отримує необхідний для роботи методу набір файлів, зберігає їх у файловому сховищі, створює завдання і записує всю необхідну інформацію про створену задачу в БД. Завдання ставиться в загальну чергу виконання, і вся необхідна інформація про завдання відправляється в доступний екземпляр сервісу Deep Steganography через менеджер завдань. Веб-сервер відповідає HTTP -статусом «200 OK», в JSON -тіло відповіді (рис. 3.6) веб-сервер включає повну інформацію про створену задачу - статус виконання завдання (pending на момент створення), повідомлення з результатами виконання (якщо завдання виконано і якщо метод може надати додаткову інформацію про виконання), унікальний ідентифікатор задачі, час старту завдання та час завершення (якщо задача виконана - у прикладі нижче завдання тільки ініційована і знаходиться в процесі виконання):


```
{
    "status": "pending",
    "message": "",
    "task_id": "8d7c529f-a43c-4056-ab81-690371c95a11",
    "datetime_start": "2020-12-25 12:44:47.890615",
    "datetime_finish": null
}
```


Рисунок 3.6 – Код відповіді

2. Перевірка статусу виконання завдання*:* користувач або потенційна front end частина може відслідковувати статус завдань через список усіх завдань, запущених у рамках фреймворку. Також можна отримати статус виконання конкретної задачі, передавши її унікальний ідентифікатор «task_id» як додатковий параметр. Для отримання результатів статус завдання повинен бути переведений у стан «success» (UUID -ідентифікатор задачі нижче скорочено в рамках відображення):

GET / api / tasks / list?task_id=8d7c529f-... HTTP/1.1

Host: localhost: 8888

3. Отримання результатів виконання завдання*:* користувач,

використовуючи значення поля «task_id» (ідентифікатор завдання) відповіді звертається до веб-сервера, щоб отримати результати виконання завдання (якщо вони доступні та завдання виконано до моменту обробки запиту):

GET / api / tasks / results?task_id=8d7c529f-... HTTP/1.1

Host: localhost:8888

Веб-сервер відповідає HTTP -статусом «200 OK» і набором доступних для даного завдання результатів: в даному випадку, користувач отримує посилання для доступу до зображення-контейнеру (рис. 3.7), яке було згенеровано в результаті процесу роботи методу Deep Steganography (поле «encoded_files» в даному випадку):

```
{
    "encoded_messages": [],
    "encoded_files": [

   "http://localhost:8888/api/tasks/files/8d7c529f-..."
    ]
}
```

Рисунок 3.7 – Код посилання

4. Збереження отриманих результатів (завершення роботи): Користувач звертається до посилань, представлених у полі «encoded_files» і зберігає отримані файли — згенеровані або відновлені. Процес завершено.

3.4 Результати і продуктивність

Результати продуктивності та швидкості роботи методів у рамках взаємодії з фреймворком представлені в табл. 3.1. Зазначений у таблиці час є повним циклом кодування та декодування, включаючи створення завдання, повне виконання та запис усіх допоміжних даних.

Таблиця 3.1 – Швидкість роботи методів фреймворку

Метод	**Час виконання кодування (сек.)**	**Час виконання декодування (сек.)**
Deep Steganography	4.3866	2.4494
Модифікований Deep Steganography	0.7495	0.3860
HiDDeN	0.6359	0.1106

Цей розділ розглядає ключові особливості розробленого фреймворку. В рамках опису функцій фреймворку розглядаються компоненти та архітектура, що використовуються. Також розглядається сценарій використання фреймворку користувачем у практичному середовищі застосування для виконання реальних завдань. Крім того, наведено результати продуктивності та швидкості роботи фреймворку.

РОЗДІЛ 4. БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ, ОСНОВИ ОХОРОНИ ПРАЦІ

4.1 Класифікація шкідливих та небезпечних виробничих факторів

Шкідливий виробничий фактор – небажане явище, яке супроводжує виробничий процес і вплив якого на працюючого може призвести до погіршення самопочуття, зниження працездатності, захворювання, виробничо зумовленого чи професійного, і навіть смерті, як результату захворювання. Небезпечний виробничий фактор – небажане явище, яке супроводжує виробничий процес і дія якого за певних умов може призвести до травми або іншого раптового погіршення здоров'я працівника (гострого отруєння, гострого захворювання) і навіть до раптової смерті [11].

Поділ несприятливих чинників виробничого середовища на шкідливі та небезпечні зумовлене різним характером їх дії на людський організм, тим, що вони потребують різних заходів та засобів для боротьби з ними та профілактики викликаних ними ушкоджень, а також рядом причин організаційного характеру. В той же час між шкідливими та небезпечними виробничими факторами інколи важко провести чітку межу. Один і той же чинник може викликати травму і профзахворювання (наприклад, високий рівень іонізуючого або теплового випромінювання може викликати опік або навіть призвести до миттєвої смерті, а довготривала дія порівняно невисокого рівня цих же факторів – до хвороби; пилинка, що потрапила в око, спричиняє травму, а пил, що осідає в легенях, – захворювання, що зветься пневмоконіоз). Через це всі несприятливі виробничі чинники часто розглядаються як єдине поняття – небезпечний та шкідливий виробничий фактор (НШВФ) [12]. За своїм походженням та природою дії всі НШВФ можна поділити на 5 груп: фізичні, хімічні, біологічні, психофізіологічні та соціальні До фізичних НШВФ відносяться машини та механізми або їх елементи, а також вироби, матеріали, заготовки тощо, які рухаються або обертаються; конструкції, які руйнуються; системи, устаткування або елементи обладнання, які знаходяться під підвищеним тиском; підвищена запиленість та загазованість повітря; підвищена або понижена температура повітря, поверхонь

приміщення, обладнання, матеріалів; підвищені рівні шуму, вібрації, ультразвуку, інфразвуку; підвищений або понижений барометричний тиск та його різкі коливання; підвищена та понижена вологість; підвищена швидкість руху та підвищена іонізація повітря; підвищений рівень іонізуючих випромінювань; підвищене значення напруги в електричній мережі; підвищені рівні статичної електрики, електромагнітних випромінювань; підвищена напруженість електричного, магнітного полів; відсутність або нестача світла; недостатня освітленість робочої зони; підвищена яскравість світла; понижена контрастність; прямий та віддзеркалений блиск; підвищена пульсація світлового потоку; підвищені рівні ультрафіолетової та інфрачервоної радіації; гострі крайки, зачипки, шершавість на поверхні заготовок, інструментів та обладнання; розташування робочого місця на значній висоті відносно землі (підлоги); слизька підлога; невагомість.

Хімічні НШВФ:

- за характером дії на організм людини поділяються на токсичні, задушливі, наркотичні, подразнюючі, сенсибілізуючі, канцерогенні, мутагенні та такі, що впливають на репродуктивну функцію;

- за шляхами проникнення в організм людини поділяються на такі, що потрапляють через: 1) органи дихання; 2) шлунково-кишковий тракт; 3) шкіряні покриви та слизова оболонка;

- які перебувають у різному агрегатному стані: 1)твердому 2)газоподібному 3)рідкому.

Біологічні НШВФ – це: - патогенні мікроорганізми (бактерії, віруси, рикетсії, спірохети, грибки, найпростіші) та продукти їхньої життєдіяльності; - макроорганізми (тварини та рослини) та продукти їхньої життєдіяльності. До психофізіологічних НШВФ відносяться фізичні (статичні та динамічні) перевантаження і нервово-психічні перевантаження (розумове перенапруження, перенапруження аналізаторів, монотонність праці, емоційні перевантаження). 6 Соціальні НШВФ – це неякісна організація роботи, понаднормова робота, змушеність праці в колективі з поганими відносинами між його членами, соціальна ізольованість з відривом від сім_ї, зміна біоритмів, незадоволеність

роботою, фізична та/або словесна образа та її ризик, насильство та його ризик. Один і той же НШВФ за природою своєї дії може належати водночас до різних груп.

4.2 Вплив вібрації на людину

Вібрація - це механічні коливання пружних тіл або коливальні рухи механічних систем. Для людини вібрація є видом механічного впливу, який має негативні наслідки для організму [12].

Причиною появи вібрації є неврівноважені сили та ударні процеси в діючих механізмах. Створення високопродуктивних потужних машин і швидкісних транспортних засобів при одночасному зниженні їх матеріалоємності неминуче призводить до збільшення інтенсивності і розширення спектру вібраційних та віброакустичних полів. Цьому сприяє також широке використання в промисловості і будівництві високоефективних механізмів вібраційної та віброударної дії.

Дія вібрації може приводити до трансформування внутрішньої структури і поверхневих шарів матеріалів, зміни умов тертя і зносу на контактних поверхнях деталей машин, нагрівання конструкцій. Через вібрацію збільшуються динамічні навантаження в елементах конструкцій, стиках і сполученнях, знижується несуча здатність деталей, ініціюються тріщини, виникає руйнування обладнання. Усе це приводить до зниження строку служби устаткування, зростання імовірності аварійних ситуацій і зростання економічних витрат. Вважають, що 80% аварій в машинах і механізмах здійснюється внаслідок вібрації. Крім того, коливання конструкцій часто є джерелом небажаного шуму. Захист від вібрації є складною і багатоплановою в науково-технічному та важливою у соціальноекономічному відношеннях проблемою нашого суспільства [13].

Вплив вібрації на людину залежить від її спектрального складу, напрямку дії, прикладення, тривалості впливу, а також від індивідуальних особливостей людини. При оцінці вібраційного впливу потрібно враховувати, що коливальні процеси притаманні живому організму. В основі серцевої діяльності і кровообігу

та біострумів мозку лежать ритмічні коливання. Внутрішні органи людини можна розглядати як коливальні системи з пружними зв'язками. Частоти їх власних коливань лежать у діапазоні 3..6 Гц. Частоти власних коливань плечового пояса, стегон і голови щодо опорної поверхні (положення стоячи) складають 4...6 Гц, голови щодо плечей (положення сидячи) 25...30 Гц.

При впливі на людину зовнішніх коливань (хитавиці, струсів, вібрації) відбувається їхня взаємодія з внутрішніми хвильовими процесами, виникнення резонансних явищ. Так, зовнішні коливання частотою менш 0,7 Гц утворюють хитавицю і порушують у людини нормальну діяльність вестибулярного апарата. Інфразвукові коливання (менш 16 Гц), впливаючи на людину, пригнічують центральну нервову систему, викликаючи почуття тривоги, страху. При певній інтенсивності на частоті 6..7 Гц інфразвукові коливання, втягуючи у резонанс внутрішні органи і систему кровообігу, здатні викликати травми, розриви артерій, тощо [11].

Вібрація, що діє на людину, має широкий діапазон – від десятих часток одного до декількох тисяч Гц. Характерними ознаками шкідливого впливу вібрації на людину є можливі зміни у функціональному стані: підвищена втома, збільшення часу моторної реакції, порушення вестибулярної реакції. Медичними дослідженнями встановлено, що вібрація є подразником периферичних нервових закінчень, розташованих на ділянках тіла людини, що сприймають зовнішні коливання. Адекватним фізичним критерієм оцінки її впливу на організм людини є коливальна енергія, що виникає на поверхні контакту, а також енергія, поглинена тканинами і передана опорно-руховому апарату та іншим органам. У результаті впливу вібрації виникають нервовосудинні розлади, ураження кістково-суглобної та інших систем організму. Відзначаються, наприклад, зміни функції щитовидної залози, сечостатевої системи, шлунково-кишкового тракту. Так, медичні дослідження показали , що у працюючих в умовах вібрації відбуваються значні зміни кістковосуглобної системи, які виражаються у функціональній перебудові кісткової тканини, регіональному остеопорозі, кістовидних утвореннях у кістках, асептичному некрозі кісток, хронічних

переломах. Відзначається, що терміни виникнення змін у кістках у працівників вібраційних професій коливається в межах від 6-8 місяців до 2-5 років.

Шкідливість вібрації збільшується при одночасному впливі на людину таких факторів, як знижена температура, підвищений шум, запиленість повітря, тривала статична напруга тощо. Сучасна медицина розглядає виробничу вібрацію як могутній стрес-фактор, що має негативний вплив на психомоторну працездатність, емоційну сферу і розумову діяльність, підвищує ймовірність виникнення різних захворювань і нещасних випадків. Особливо небезпечний тривалий вплив вібрації для жіночого організму. Широкий комплекс патологічних відхилень, викликаний впливом вібрації на організм людини, кваліфікується як віброзахворювання [13].

Вібрація як фізичний чинник виробничого середовища спостерігається в металообробній, гірничодобувній, металобудівній, машинобудівній, авіаційній та інших галузях народного господарства. Джерелом вібрації можуть бути різні механізми, вібраційне устаткування, віброінструменти, акустичні системи, транспортні та сільськогосподарські машини.

Загальна вібрація поділяється на транспортну вібрацію, яка діє на людину на робочих місцях в транспортних засобах (трактори сільськогосподарські та промислові, самохідні сільськогосподарські машини (комбайни), тягачі, грейдери ті інші); транспортно-технологічну вібрацію, яка діє на людину на робочих місцях машин з обмеженою рухливістю (екскаватори, крани промислові та будівельні, гірничі комбайни, транспорт виробничих приміщень та інші) та технологічну вібрацію, яка діє на людину на робочих місцях стаціонарних машин чи передається на робочі, де немає джерел вібрації (верстати та метало-деревообробне, пресувально-ковальське обладнання, ливарні машини, електричні машини, насосні агрегати та вентилятори, обладнання для буріння свердловин, бурові верстати, машини для тваринництва, очищення та сортування зерна (у тому числі сушарні), обладнання промисловості будматеріалів (крім бетоноукладачів), установки хімічної та нафтохімічної промисловості та інші.

Оператори машин, які зазнають у процесі трудової діяльності впливу вібрації, підлягають попереднім та періодичним медичним оглядам відповідно

до Порядку проведення медичних оглядів працівників певних категорій, затвердженого Наказом МОЗ України від 21.05.2007 р. №246. Обов'язкові попередні (під час прийняття на роботу) та періодичні (протягом трудової діяльності) медичні огляди дозволять визначити стан здоров'я працівника та можливість виконання без погіршення стану здоров'я професійних обов'язків, своєчасно виявити ранні ознаки хронічного професійного захворювання, забезпечує динамічне спостереження за станом здоров`я в умовах дії шкідливих та небезпечних факторів і трудового процесу, вирішує питання щодо можливості продовжувати роботу в умовах дії шкідливих та небезпечних факторів і трудового процесу [13].

За результатами періодичних медичних оглядів роботодавець забезпечує проведення відповідних оздоровчих заходів Заключного акта у повному обсязі та усуває причини, що призводять до професійних захворювань. Організовує проведення лабораторних досліджень умов праці на робочих місцях та вживає заходів до усунення небезпечних і шкідливих для здоров'я виробничих факторів.

До роботи операторами машин допускаються особи не молодші 18 років, які пройшли попередній медичний огляд, мають відповідну кваліфікацію та ознайомлені з характером впливу вібрації на організм.

ВИСНОВКИ

У цій роботі описуються сучасні підходи до реалізації прихованих каналів та методів стеганографії за допомогою ШНМ, а також описується процес розробки дослідницького фреймворку для реалізації та аналізу прихованих каналів на основі ШНМ.

Розглянуто існуючі методи та підходи до реалізації прихованих каналів та створення стеганографічних контейнерів для передачі даних. Також розглянуто та реалізовано існуючі методи приховування інформації на основі нейронних мереж, такі як Deep Steganography, HiDDeN.

У рамках роботи запропоновано та розроблено розширення методу Deep Steganography, що дозволяє збільшити обсяг переданої секретної інформації за один сеанс передачі шляхом збільшення кількості секретних зображень, що передаються.

В результаті, спроектований та розроблений фреймворк мовою програмування python для реалізації та аналізу прихованих каналів на основі нейронних мереж. До основних особливостей фреймворку належать кросплатформеність, відмовостійкість, масштабованість, децентралізованість, незалежність від мов програмування реалізацій та швидкість обробки даних. Цей фреймворк забезпечує єдину систему управління та додавання методів, а також REST API методи взаємодії з кінцевим користувачем.

ПЕРЕЛІК ДЖЕРЕЛ

1. Lampson BW Note on the Confinement Problem // Communications of the ACM. - New York, NY, USA, 1973. - с. 613-615.

2. ДСТУ ISO/IEC 13335-1:2004 Інформаційні технології. Методи захисту. Керування інформацією й безпекою технології комунікацій. Частина 1. Поняття й моделі для інформації й керування безпекою технології комунікацій

3. ДСТУ 3396.0-96 ДЕРЖАВНИЙ СТАНДАРТ УКРАЇНИ Захист інформації Технічний захист інформації

4. CENTER NCS Guide to Understanding Covert Channel Analysis of Trusted Systems. - 1993.

5. Kaspersky Security Bulletin 2023. Top security stories. [Електронний ресурс]. – Режим доступу: https://www.kaspersky.com/about/press-releases/2023_rising-threats-cybercriminals-unleash-411000-malicious-files-daily-in-2023 (дата звертання: 28.03.2024).

6. Kaspersky Security Bulletin 2023. [Електронний ресурс]. – Режим доступу: https://media.kasperskycontenthub.com/wp-content/uploads/sites/43/2023/11/28102415/KSB_statistics_2023_en.pdf (дата звертання: 28.03.2024).

7. APT Trends Report Q2 2018. [Електронний ресурс]. – Режим доступу: https: // securelist. com/apt-trends- report-q2-2018/86487/ .

8. APT trends report Q1 2019. [Електронний ресурс]. – Режим доступу: https: // securelist. com/apt - trends- report-q1-2019/90643/ дата звертання: 10.04.2024).

9. Baluja S. Hiding Images в Plain Sight: Deep Steganography. [Електронний ресурс]. – Режим доступу: http: //www.esprockets.com/papers/nips2017.pdf (дата звертання: 11.04.2024).

10. HiDDeN: Hiding Data With Deep Networks / J. Zhu [та ін]. - 2018. - arXiv: 1807.09937 [cs.CV] .

11. Зеркалов Д.В. Безпека життєдіяльності та основи охорони праці. Навчальний посібник. К.: «Основа». 2016. – 267 с.

12. Яремко З. М. Безпека життєдіяльності: Навч. посіб. — Львів., 2005. –

301 с.

13. Желібо Є. П. Заверуха Н.М., Зацарний В.В. Безпека життєдіяльності. Навчальний посібник. – К.: Каравела, 2004. -328 с.