НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

ДОПУСКАЄТЬСЯ ДО ЗАХИСТУ

Завідувач кафедри

комп'ютерних систем, мереж та кібербезпеки

Касаткін Д.Ю., к. пед.н., доц.

Підпис ПІБ, вчене звання і ступінь

«___»_____2025 р.

КВАЛІФІКАЦІЙНА БАКАЛАВРСЬКА РОБОТА

На тему: Розробка комп'ютерної системи для захисту інформації на агропідприємстві

Спеціальність 123 «Комп'ютерна інженерія»

Гарант освітньої програми

к.фіз.-мат.н., доцент	_____	Євгеній НІКІТЕНКО
(науковий ступінь та вчене звання)	(підпис)	(ПІБ)

Керівник випускної бакалаврської роботи

к.пед.н., доцент	_____	Дмитро КАСАТКІН
(науковий ступінь та вчене звання)	(підпис)	(ПІБ)

Виконав

(підпис) (ПІБ студента)

Київ – 2025

Дмитро КАТРЕВИЧ

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ
І ПРИРОДОКОРИСТУВАНН УКРАЇНИ
ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

«ЗАТВЕРДЖУЮ»
завідувач кафедри
комп'ютерних систем, мереж та кібербезпеки

Касаткін Д.Ю., к.пед.н., доц. /
підпис ПІБ, вчене звання і ступінь
р.

З А В Д А Н Н Я
ДО ВИКОНАННЯ КВАЛІФІКАЦІЙНОЇ БАКАЛАВРСЬКОЇ СТУДЕНТА

Катревич Дмитро Олександрович
(прізвище, ім'я, по батькові)

Спеціальність (напрям підготовки) Комп'ютерна інженерія
Тема випускної бакалаврської роботи Розробка комп'ютерної системи для захисту інформації на агропідприємстві
керівник проекту (роботи) Касаткін Д.Ю., к.пед.н., доц.
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Затверджено наказом ректора НУБіП України від «16» грудня 2024 р. № 2250 «С»
Термін подання завершеної роботи на кафедру 28.05.2025
(рік, місяць, число)

Вихідні дані до випускної бакалаврської роботи _____
_____
Перелік питань, які потрібно розробити: Аналіз вимог до комп'ютерної системи, аналіз предметної області, проєктування, реалізація, тестування системи
_____

Перелік графічних документів (за потреби)_____

Дата видачі завдання "17" грудня 2024 р
Керівник випускної бакалаврської роботи _____ / Касаткін Д.Ю.
(підпис) (прізвище та ініціали)
Завдання прийняв до виконання _____ / Катревич Д.О.
(підпис) (прізвище та ініціали студента)

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів бакалаврської роботи	Строк виконання етапів роботи	Примітка
1	Аналіз вимог до системи	07.03.2025 р.	Виконано
2	Проєктування системи	21.03.2025 р.	Виконано
3	Реалізація системи	12.04.2025 р.	Виконано
4	Тестування розробленої системи	27.04.2025 р.	Виконано
5	Оформлення пояснювальної записки	07.05.2025 р.	Виконано
6	Оформлення графічного матеріалу	15.05.2025 р.	Виконано

Студент_____ / Катревич Д.О. /

(підпис) (ініціали та прізвище)

Керівник проекту (роботи)_____ / Касаткін Д.Ю./

(підпис) (ініціали та прізвище)

АНОТАЦІЯ

Дипломна робота стосується теми “ Розробка комп’ютерної системи для захисту інформації на агропідприємстві ”.

Мета роботи – вивчення проблематики секретних комунікацій і їх проведення методами правдоподібного заперечення, а також розробка програмного прототипу на базі стеганографії світлин і повного протоколу зі зберігання і приховування інформації в комп’ютерних системах, розрахована на користувача з мінімальним до середнього знаннями інформаційних технологій і інформацію рівня екстремальної чутливості. Також вивчення можливостей аналогічного методу в поєднанні з IoT, розробка повної системи для сценарію з датчиками.

Ключовим елементом роботи є розробка методів покращення і перспектив майбутнього, що були виявлені в результаті досліджень.

ABSTRACT

The diploma thesis addresses the topic "Development of a protocol to Enhance the Degree of Information Security in Computer Systems."

The goal of the work is to study the issues of covert communications and their implementation using plausible deniability methods, as well as to develop a software prototype based on image steganography and a complete protocol for storing and hiding information in computer systems. This system is designed for users with minimal to intermediate knowledge of information technology and aims to protect information of extreme sensitivity.

Additionally, the thesis explores the possibilities of combining this method with IoT, developing a comprehensive system for a scenario involving sensors.

A key element of the work is the consideration of improvement methods and prospects identified as a result of the research.

ЗМІСТ

ВСТУП 8

РОЗДІЛ ПЕРШИЙ: ТЕОРІЯ ЗА МЕТОДАМИ..... 10

1.1 Правдоподібне заперечення..... 10

1.2 Стеганографічний метод 14

РОЗДІЛ ДРУГИЙ: СТЕГАНОГРАФІЯ КАРТИНКИ**Помилка! Закладку не визначено.**

2.1. Алгоритм LSB (Least Significant Bit)**Помилка! Закладку не визначено.**

2.2 Застосування генетичного алгоритму в алгоритмі LSB ..... **Помилка! Закладку не визначено.**

2.2.1. Початкова популяція ..... **Помилка! Закладку не визначено.**

2.2.2. Функція пристосованості ..... **Помилка! Закладку не визначено.**

2.2.3. Генетичні оператори..... **Помилка! Закладку не визначено.**

2.2.4. Еволюційний процес..... **Помилка! Закладку не визначено.**

2.2.5. Остаточне рішення..... **Помилка! Закладку не визначено.**

2.3. Застосування алгоритму кластеризації в алгоритмі LSB..... **Помилка! Закладку не визначено.**

РОЗДІЛ ТРЕТІЙ: ІМПЛЕМЕНТАЦІЇ І ІДЕЇ ПОКРАЩЕННЯ 33

3.1 Прототип, розроблений для роботи 33

3.2 Ідеї для покращення і перспективи..... 38

3.3 Стеганографія та IoT..... 42

ВИСНОВКИ..... 45

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ..... 47

ВСТУП

В часи розквіту електронних технологій і в умовах військового конфлікту як ніколи важливе усе, що пов’язане з кібербезпекою. Як на вищих інстанціях, так і серед звичайних людей, важливо розсудити про спосіб передачі і захисту екстремально чутливої інформації в умовах того, що нападник може бути більш авторитетною людиною, мати фізичний доступ до пристроїв суб’єкта та сам факт наявності чогось зашифрованого на пристрої може бути компрометуючим для суб’єкта.

В умовах сучасності ми частіше за все можемо спостерігати серед людей не-інформатичних спеціальностей дві тенденції: або вони надто легковажно ставляться до інформації, яка може виявитись для них компрометуючою, або навпаки, і намагаються завдати дій, які б їх захистили (наприклад, повністю видаляють переписки зі свого боку), але цим іноді можуть поставити себе в більш незручне положення. Хоча в другому випадку це може бути ефективним, але для того, щоб приховати екстремально чутливу інформацію, не привернувши до себе непотрібної уваги, його може бути недостатньо.

В подібних ситуаціях, коли інформація під питанням надто чутлива, щоб навіть сам факт її наявності був викритим, вступає в гру метод «правдоподібного заперечення», легальний концепт якого розглянутий в багатьох роботах з і полягає в тому, щоб відвертати наявність об’єкту під питанням до останнього, заздалегідь, звичайно, запевнивши її приховування і захист кількома рівнями, які спираються в тому числі і в великому ступені на людяний фактор.

Говорячи про людяний фактор, мається на увазі, що ми беремо за перевагу те, що «атакуючий» – теж людина і може недооцінити нас в багатьох інстанціях. В залежності від того, скільки рівнів оминання ми додамо, ми зможемо захиститись від більшої кількості потенційних атакуючих, якщо не усіх одразу. Треба також брати до уваги, що окрім того, щоб зробити свій

пристрій, який має чутливу інформацію, максимально «не-підозрілим», потрібно також самому фізично поводитись не-підозріло. Подібний метод має застосовуватись досвідченими людьми з достатньо хорошою акторською грою і лише в ситуаціях, де інформація під питанням дійсно настільки чутлива, щоб бути схованою таким чином. Оскільки чим більше рівнів захисту, тим більше також незручностей для користувача, а також чим більш розсповсюджений метод – тим більше причин його підозрювати.

До того ж, використання цього методу має бути вдумливим також тому, що може поставити користувача в більшу небезпеку, ніж він був до того, якщо застосування ним цього методу все ж викриють. Тому що після цього суб’єкт навіть якщо викриє те, що з самого початку сховав, не зможе довести, що він не сховав щось більше.

Мета роботи полягає в тому, щоб розміркувати і сформулювати спосіб передачі або зберігання чутливої інформації від початку і до кінця, за допомоги методи правдоподібного заперечення і стеганографії в фотографіях і інш., базуючись на тому ж методі, також надавши прототип базової імплементації. Також в роботі планується розкрити потенціали для покращення і запропонувати інші способи застосування цього методу.

РОЗДІЛ ПЕРШИЙ: ТЕОРІЯ ЗА МЕТОДАМИ

1.1 Правдоподібне заперечення

Існує безліч можливих методів посилення ступеню захисту інформації в комп'ютерних системах і мережах. В залежності від структури системи, потенційних загроз і рівня чутливості інформації, необхідно розглядати різні можливі варіанти.

Метод правдоподібного заперечення – це більше, ніж набір інструкцій або конкретний алгоритм, якому потрібно слідувати, щоб захистити інформацію. Цей метод використовується не лише в інформатиці і має відношення до захисту інформації і себе в цілому, випливаючи з назви, прагнучи дати собі можливість правдоподібно заперечити своє відношення до справи під питанням, або існування цієї справи. Правдоподібне заперечення в інформатиці також представляє собою підхід в цілому, підхід, який вимагає креативності і який достатньо гнучкий, щоб до невпізнавості викривити дані, які необхідно захистити. Все залежить від кількості рівнів, які користувач бажає додати, щоб захистити свої дані. Чим більше рівнів – тим більше ймовірність, що їх не розкриють, але також більше незручностей для того, щоб пізніше повернутись до початкової інформації. [1]

Правдоподібне заперечення включає в себе сукупність дій і поведінки, спрямовану на посилення ефективності стеганографічного методу, застосованого до інформації, а також інші можливі дії, які спрямовані на те, щоб дозволити об'єкту заперечувати своє відношення до передачі або обладання інформацією під питанням. Протягом роботи ми будемо концентруватись на імплементації з стеганографією, але важливо визначити і пояснити, що для ефективності методу необхідно розуміти його історію, цілі і його "батька".

Далеко не в усіх випадках використання цього методу його можна назвати дійсно “правдоподібним”. Як приклад використання методу в неінформатичному, а шпигунському контексті, ще з середньовіччя можна навести класичний трюк з письменністю лимонним соком, коли таємне послання на папері з не-підозрілим листом проявляється після того, як його підігрієш – і після прочитання лист одразу спалюють. В ті часи, коли цей метод був невідомим, той, хто міг перехопити листа, не здогадувався підігріти його, щоб побачити таємне послання – на це і покладалась ефективність такого способу. Так само легко можна було потім заперечувати факт обміну секретною інформацією. В наші ж часи це вже не можна назвати дійсно “правдоподібним”, оскільки набагато більше людей знають про цей трюк і викриття вкрай ймовірне. На протиставлення цьому пізніше стали використовувати невидимі чорнила, проявлення яких вимагало іншої формули, ніж просте підігрівання. Але і цей метод перестав бути актуальним, на протиставлення дешевшим і надійнішим.

Схожим принципом ми користуємося, застосовуючи “правдоподібне заперечення” в інформатиці. Перш за все, ми вивчаємо очікування нашого потенційного супротивника і шукаємо способи обманути його очікування, бажано кілька разів, настільки, щоб він не зміг довести, що ми з самого початку щось приховуємо і обманули його хоч раз. Це метод, який так чи інакше застосовували в усі часи розвитку інформатичних технологій і не лише, тому важко виокремити, коли конкретно він, разом з стеганографією (див. 1.2) набув більшої популярності. Очевидно одне: для захисту інформації його використовують, коли інформація під питанням є надзвичайно чутливою, тому цільова авдиторія цього методу – це шпигуни, журналісти, посадовці і злочинці.

Коли мова йде про обмін чутливою інформацією між двома людьми, для ефективності оного необхідно заздалегідь узгодити протокол між надсилачем і отримувачем (бажано – асиметричний) який забезпечить наступне:

- Автентифікацію надсилача і адресата [2]
- Перевірку цілісності даних
- Стійкість проти цілеспрямованих атак

Коли ми говоримо про автентифікацію, в ідеальному випадку пропонується робити її на трьох рівнях з боку отримувача:

- автентифікація пристрою – класичний метод, використованих в багатьох протоколах обміну інформації, де на пристрої буде наявний ключ ідентифікації, а також “white-list” знайомих пристроїв.
- автентифікація персони – ми розглядаємо випадок, коли ми не можемо гарантувати фізичну безпеку пристрою. В такому випадку зазвичай важко говорити про інформаційну безпеку, але тим не менш, в зазначених сценаріях це необхідний етап і має критичну важливість. Автентифікація особистості – це етапи протоколу обміну, відомі лише персоні, які вона має зберігати в голові і може відмовляти своє їх знання. Це не такі очевидні речі, як пароль від комп’ютера, а більш неочевидні дії, які можна списати на випадкові. Приклад буде наведено в розділі 3, при описі пропонованого pipeline.
- автентифікація намірів – необов’язковий етап, який не буде далі розглядатись протягом роботи. Він представляє собою останній шанс для надсилача відмовити в наданні доступу до секретних даних в випадку виявлення підозрілих дій, наприклад, повторного запиту до певної інформації через довгий час. Але цей етап буде вимагати доступу до мережі після передачі стеганоконтейнера, що може не бути оптимальним рішенням. До того ж, в певній мірі захист, заміщуючий це можна забезпечити зашифровуючи всередину контейнера також умовний TTL, який буде слугувати строком придатності для секретних даних і буде “спалювати” сам себе через певний час після прочитання. Цей етап необхідний для того, щоб забезпечити безпеку даних в випадку, якщо одна з персон в обміні інформації зрадить домовленності.

Потенційну ефективність демонструє наступна схема з прикладом (рис.1), де в базі даних умовної дослідницької станції зберігається інформація під питанням. Інформація має сама по собі представляти деяку цінність, щоб змусити супротивника повірити в те, що ми захищали саме її, коли насправді, всередині неї прихована інформація вищої цінності.

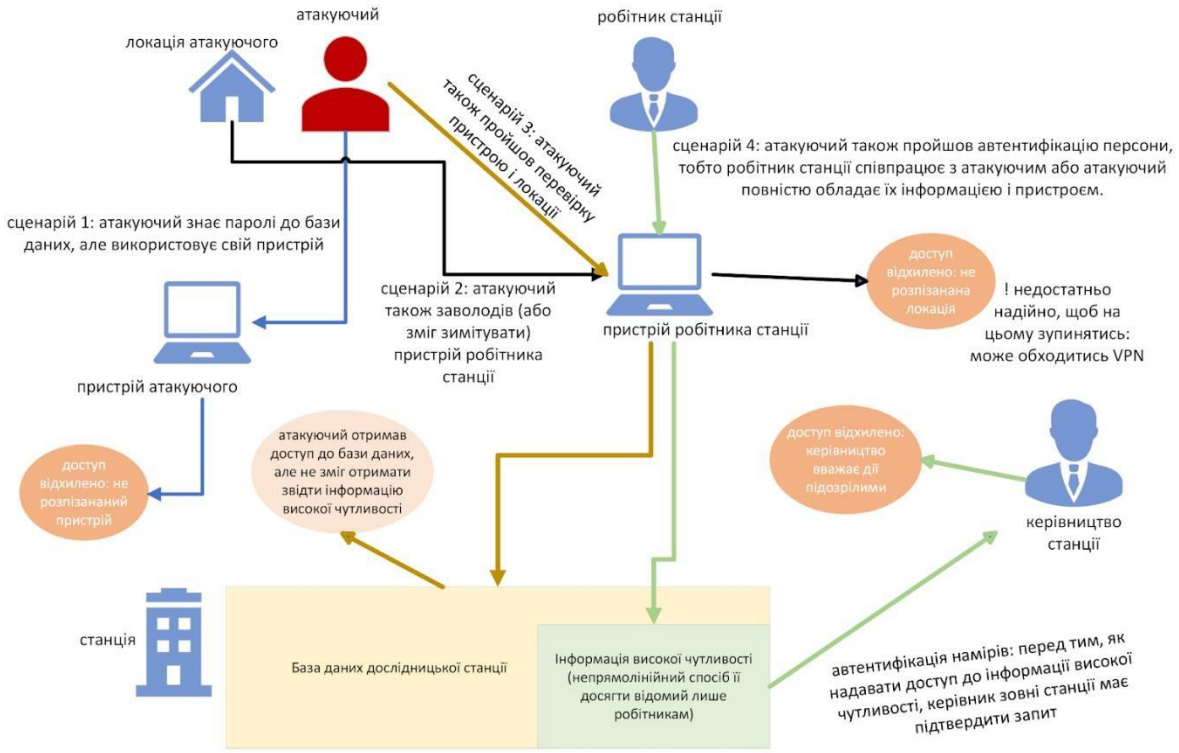


Рис. 1.1 – Захист новими шарами автентифікації в різних сценаріях

Підсумовуючи вищесказане, сутність методу полягає в тому, щоб зробити вигляд, що наші секретні дії – це випадковість, і поводити себе непідозріло, тим самим захищаючи себе і секретні дані навіть в випадку, коли атакуючий має фізичний доступ до пристрою суб'єкта або погрожує життю суб'єкта задля отримання інформації. Якщо з'являється підозра, то сенс методу знищений, незалежно від того, чи дістався атакуючий інформації.

Випадки, коли метод застосовуються, мають бути дійсно екстремальними і метод має бути використований з особливою обережністю,

натренованими персонами, інакше він може поставити суб'єкта під більшу загрозу, ніж той був до того. Звичайно, можна посперечатись, що метод можна використовувати для інтересу, з не надто секретними даними, одначе при виявленні того, що цей метод використовується один раз, ви не зможете довести, що не використали його знову і відмовляти те, що зробили щось ще, навіть якщо ви вже розкрили всю секретну інформацію, яка в вас була. Це умовна гра на довіру, і як тільки вона підірвана, ситуація стає лише гірше.

1.2 Стеганографічний метод

Стеганографія – ключовий інструмент для правдоподібного заперечення в інформатичній безпеці, коли мова йде про ситуації, де ми не можемо гарантувати приватність каналів передачі, анонімність і збереження власності фізичного пристрою з інформацією під питанням, або ж ми не можемо використати достатньо стійкий кріптографічний метод (в деяких країнах існують закони, що забороняють використання стійких кріптографічних методів без ліцензії або в принципі, або дозволяють державним органам, поліції або інш. вимагати розшифрування у суб'єкта [3]), або хочемо посилити його ефективність, приховавши сам факт його використання. Сам термін складається з двох грецьких слів – *steganós* (στεγανός), що означає «прихований», і *-graphia* (γραφή), що означає «письмовість», що повністю передає його суть.

В цілому же стеганографія має декілька застосувань: електронні відбитки, вотермарки, виявлення модифікацій файлу, обмін секретною інформацією, тощо. Оскільки робота центрується навколо захисту інформації за допомоги правдоподібного заперечення, то опис, наданий далі, посилаючись на стеганографічний метод, має на увазі саме його застосування в контексті передання секретної інформації. Деякі джерела часто використовують поняття стеганографії лише в контексті секретних

комунікацій і виділяють вотермарки в окрему категорію, хоч технічно їх принцип і схожий, [4] але вотермарки в будь-якому випадку не є ціллю дослідження.

Принцип стеганографічного методу полягає в тому, щоб заповнити стеганоконтейнер інформацією, не піддаючи початковий зміст контейнеру видимій людському оку зміні. Саме тому для найбільш класичного випадка як контейнери використовують медіа аналогового характеру, e.g. світлини, аудіо, відео. В цьому випадку незначні зміни контейнеру списуються на похибку . В сучасності, однак, також існують методи, які дозволяють програмно передати секретну інформацію, не генеруючи змін контейнеру, такі як, наприклад: генерування валідного тексту, де слова будуть індикаторами, або шифрування інформації за допомогою шахової дошки [5]. В роботі надалі буде використовуватись класичний метод, тому мова нижче буде йти про нього.

Принцип роботи класичного методу доволі уніфікований, і включає в себе дві сторони, між якими стається введення й виведення інформації, самі дані, які будуть приховані, та стеганоконтейнер, який буде в ході передання між цима двома інстанціями експозований потенційному атакуючому (рис. 1.2). Варто зазначити, що “надсилач” і “отримувач” – це умовні позначення і можуть фізично бути однією і тією самою людиною.

Рис. 1.2 – Модель стеганосистеми

Стеганографічним каналом виступає будь-який обраний метод передачі інформації, від соціальної мережі до фізичного конверту з листом. Проведучи паралелі зі старими стеганографічними методами, таким як невидимі чорнила, пустим контейнером буде виступати звичайний, не-підозрілий лист. “Кодером” буде написання певними невидимими чорнилами справжніх даних, а ключем декодування буде формула чи протокол проявки. Стеганографічним каналом в такому сценарії, відповідно, буде метод передання листа, пошта чи гонець, а декодування – це проявка.

Але навіть в такому, симпліфікованому сценарії, легко бачити, що до обрання стеганографічного каналу необхідно підходити з уважністю. Оскільки ті, хто контролюють канал, можуть, якщо не виявити початкове повідомлення, то принаймні пошкодити контейнер так, що воно не зможе бути декодоване в результаті.

Оскільки класичний метод продукує зміни початкового контейнеру, то критично важливим в протоколі передачі є те, щоб початковий контейнер у вигляді до додання туди інформації не потрапив в руки атакуючого.

В роботі ми розглядатимемо класичний варіант розвитку подій багатьом відомого методу з відносно очевидними пропозиціями адаптації алгоритму. Хоча основна мета цієї роботи полягає в тому, щоб розміркувати можливості використання методу, необхідно зазначити, що найчастіше на практиці він використовується з зловмисними намірами. Тому подальші розмірковування мають також слугувати застереженням.

Коротко також розглянемо, як виявити сліди застосування стеганографії?

Найлегше це, звичайно, зробити, якщо на руках мається початковий контейнер, щоб порівняти його з заповненим. Ще легше, якщо, погано розраховуваши пропорції контейнеру до прихованої інформації, користувач деформував контейнер настільки, що зміни стають видимими людському оку. Але обидва ці випадки на практиці виникають вкрай рідко.

Для того, щоб виявити сліди застосування стеганографії, можна спробувати виявити в ньому повторюючиїся патерни за допомогою статистичного аналізу, методи якого включають в себе, але не обмежені хістоаналізом (де актуально), трансформативним, аналізом JPEG-компресії і так далі. За іншим методом, шукаються патерни чи характерні маркери відомих стеганографічних систем. [6] Також пошук полегшується, якщо ми маємо уявлення про те, що саме необхідно шукати – як, наприклад, з вотермарками.

Також, в залежності від використаного каналу, існують більш доступні для людини індикатори, які можуть містити сліди застосування стеганографії. В випадку, наприклад, коли каналом передачі є картинка на вебсайті, її несправедливо величезний розмір може вже вказувати на наявність всередині прихованої інформації, оскільки зазвичай медіа на веб сторінках намагаються оптимізувати (хоча це і може, звичайно, бути й всього лише індикатором поганих практик розробки).

Загалом, існують застосунки, такі як Stego Suite, які здатні виконувати стеганографічний аналіз для виявлення слідів застосування стеганографії, але з часом з’являється більше нових методов і алгоритмів, і обидві сторони – приховування і виявлення – постійно розвиваються, тому неможливо цілковито покладатись на застосунки і існуючі програми, хоч їх приклад і, звичайно, стане в нагоді, а в випадку виявлення – додаткова перевірка ніколи не буває зайвою, щоб почувати себе спокійніше.

Якщо, наприклад, певний співробітник компанії поводиться себе підозріло, а компанія має на це ресурси, то не грішно перевірити всі файли на його пристрої на наявність слідів стеганографії відомими методами аналізу. Але не-виявлення слідів ще не буде стовідсотковою гарантією, що ця людина не користувалась методом.

Підсумовуючи, стосовно термінології варто зазначити, що стеганографія не обов’язково використовується для правдоподібного заперечення, а

правдоподібне заперечення не обов’язково використовує стеганографію. Але в контексті уявної ситуації, для якої буде використовуватись розроблений в ході роботи програмний прототип, ці два поняття нероздільні. (див. 3 розділ)

Також, цікаво розглянути один з історичних способів стеганографії “маска”. Відомим прикладом його використання є лист від Генрі Клінтона до Джона Бургойне 1777 року [7]. Його принцип складається в тому, що лист з самого початку складається таким чином, що виглядає як звичайний, не викликаючий підозру лист. Насправді же, якщо накласти на нього листаплікацію певної форми, яка буде показувати лише певні слова (або літери), то буде виявлене справжнє, секретне повідомлення. Цей метод раніше потребував багато зусиль задля справжньої ефективності.

ПРОГРАМНІ МЕХАНІЗМИ ОЦІНКИ РІВНЯ ЗАХИЩЕНОСТІ КОМП'ЮТЕРНИХ СИСТЕМ

2.1 Сучасні програмні аналоги оцінки захищеності

З розвитком комп'ютерних систем почався «великий бум» в індустрії технологій, який спричинив зацікавленість багатьох верств населення до даних пристроїв. В той час, деякі люди досконально вивчили комп'ютери та операційні системи, доступні для них, та почали несанкціоновано отримувати доступ за допомогою незахищених портів або операційних систем та незаконно привласнювати інформацію з даних пристроїв. Тоді розробники програмного забезпечення почали створювати певні програмні засоби, що отримали назву сканери уразливостей систем та мереж, для перевірки та перешкоджанню виникнення загроз для комп'ютерних системи.

Сьогодні вже існує величезні набори різних програм для оцінки захищеності комп'ютерних систем. До таких систем відносять: сканери портів, сканери вразливостей та системи моніторингу захищеності [30, 32, 33].

Сканери портів являють собою програмні засоби, що виконують перевірку сервера або хостів на наявність відкритих портів, на які можуть бути здійснені атаки зловмисниками для отримання інформації з даної комп'ютерної системи. Також дані сканери дозволяють побачити, які саме пакет даних або пакети певного з'єднання передаються через транспортні протоколи з однієї IP-адреси на іншу. Таке сканування під силу сканерам Nmap, SuperScan, Wireshask, Advanced IP Scanner, Network Utility та інші [33].

Сканери вразливостей – це програмні засоби, що здійснюють діагностику і моніторинг комп’ютерних систем, підключених до мережі, для сканування мереж, систем, програм на проблеми в системі безпеці, оцінки та усування уразливостей. Дані програми потрібні для визначення наявності ділянок з низьким безпековим порогом, яким можуть скористатись хакери для отримання несанкціонованого доступу до самої системи. До таких сканерів відносять: Nessus, Retina, Wikto, ISS Internet Scanner, SAINT, XSpider, Shadow Security Scanner та інші [30, 31].

Далі піде мова про системи моніторингу захищеності. Це системи, що дозволяють виконувати аналіз всіх подій та випадків, вписаних в консоль, розглядує наслідки атак, записує та зберігає всі події пов’язані з інформаційною безпекою системи. Дані системи отримали назву SIEMсистем. Дані системи є складними в роботі та тримають в собі величезну кількість інформації, оскільки всі події зберігаються в пам’яті програми [32].

Всі вище вказані програми є відкритими у доступі, проте деякі є платними, такі як Nessus. В той же час, даний програмний елемент дає можливість скористатись безкоштовною пробною версією, що і буде зробленоо для виконання завдань даної дипломної роботи.

2.2 Огляд використаних програмних інструментів

В даній роботі, вивчивши багато інформації з відкритих джерел Інтернету, різних посібників та книг, було вирішено, що найкращими програмними засобами для вирішення задачі аналізу захищеності комп’ютерних систем будуть:

- Командна строка Windows 10 (cmd.exe);
- Програма для виявлення мережі та аудиту безпеки Nmap v7.93 – Zenmap GUI;

- Сканер вразливостей системи Nessus Vulnerability Scanner.

Дамо опис кожного програмного елемента для повного розуміння методу рішення даної задачі.

Командна строка cmd.exe є керуючим інтерпретатором текстових команд, що вводяться користувачем, розроблений спеціально для операційної системи Windows компанією Microsoft. Дана програма емулює багато можливостей командного рядка, присутніх в MS-DOS. Дана програма

виконує та автоматизує завдання за допомогою певних сценаріїв та пакетних файлів, допомагаючи користувачу даної програми в адмініструванні операційної системи, в усуненні та вирішенні певних проблем даної операційної системи. За допомогою програми можливо отримати доступ до повної інформації системи, директорій, робити зміни в них або видаляти певні елементи комп'ютерної системи, працюючої на операційній системі Windows 10 [34].

Для відкриття даної програми потрібно натиснути комбінацію Win + R, в текстовому полі відкритого вікна ввести «cmd» та натиснути Enter. Відкриється чорне вікно з назвою «C:\WINDOWS\system32\cmd.exe», що є фактично шляхом до даної програми в файлах операційної системи Windows 10. Для отримання інформації про команди програми cmd.exe потрібно ввести команду «help» після строки «C:\Users\(ім'я користувача)>».

Покажемо інтерфейс даної програми за допомогою рис. 2.1.

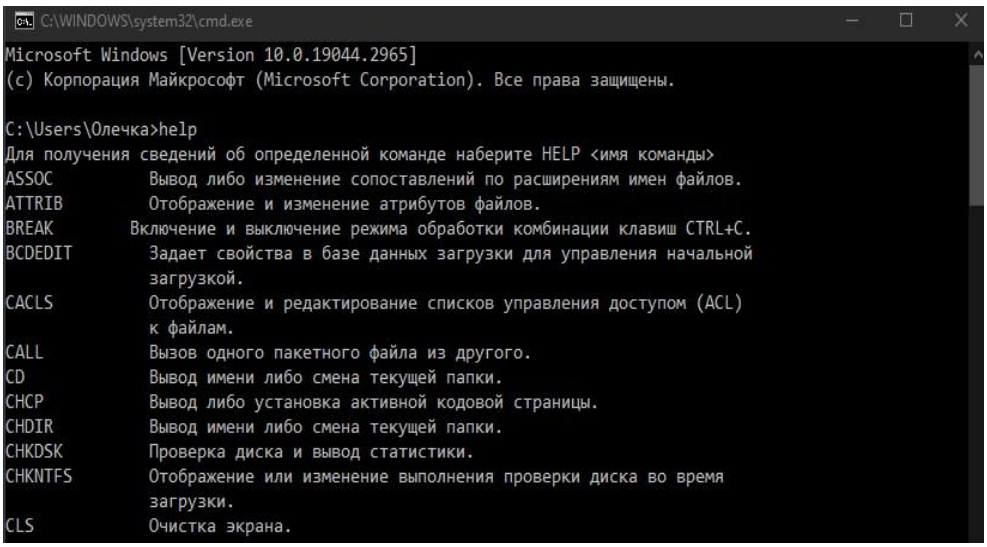


Рисунок 2.1 – Командна строка cmd.exe.

Програма Nmap – це програмний засіб для аудиту безпеки та виявлення повної інформації про мережу, до якої підключена комп'ютерна система. Дана програма є відкритою для користувачів багатьох операційних систем, таких як: Linux, Unix, Mac OS, Windows, тощо. Дана програмна утиліта є додатком до командної строки cmd.exe з власними командами для отримання інформації за допомогою методів сканування UDP, TCP, TCP-SYN та інші.

Дана програма також має графічний користувацьким інтерфейсом під назвою Nmap v7.93 – Zenmate GUI, що є більш простим для розуміння користувачів

[35].

Zenmate GUI має простий для розуміння інтерфейс з автоматичним створенням команд, що використовується в командній строці, розділяє дані на різні вкладки, виділяючи хости комп'ютерної системи, порти (відкриті та приховані), деталі хосту, виконані сканування та будує мапу мережі, позначаючи всі можливі хості присутні в комп'ютерній системі.

Програма Nmap v7.93 – Zenmap GUI була вибрана тому, що вона є безкоштовною, легкою в розумінні, з гарним користувацько-орієнтованим інтерфейсом та великим функціоналом, постійно оновлюється

професіоналами в межах довідкових сторінок та покращення інтерфейсу. Головною причиною вибору стало те, що дана програма є дуже потужною та моє можливість перевірити всі наявні порти (65535 портів) менше ніж за 30 секунд.

Вигляд даної програми з графічним користувацьким інтерфейсом можна побачити на рис. 2.2.

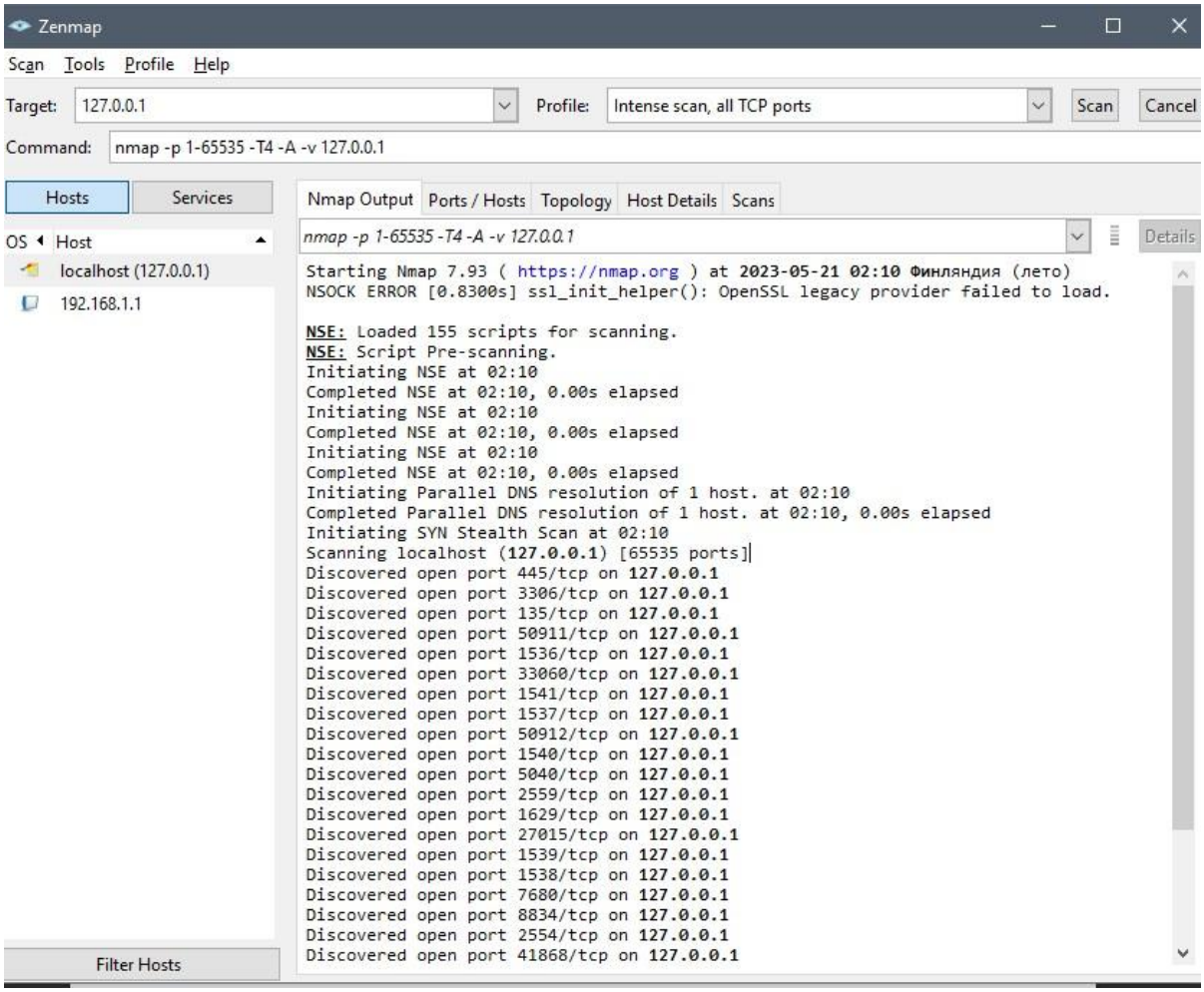


Рисунок 2.2 – Програма Nmap v7.93 – Zenmap GUI

Наступною буде розглянуто програму, яка є головною для дипломної роботи, оскільки вона шукає вади в комп'ютерній системі. Даною програмою є Nessus Vulnerability Scanner від компанії Tenable Network Security, розроблену у 1998 році. Програмний продукт створений для автоматичного знаходження вразливостей комп'ютерних систем у версіях певних служб комп'ютера та певних доменів, перевіряє паролі на

їх складність та розглядає всі можливі конфігурації з помилками в мережі інформаційної системи. Дана утиліта є серверною, з клієнтоорієнтованим інтерфейсом, що працює на базі вибраного вами браузера: Google Chrome, Mozilla Firefox, Microsoft Edge, Safari, тощо.

Відмінною рисою представленого сканера уразливостей є наявність

«розумних» плагінів. Це означає, що при будь-якому зловмисному переміщенні порта певної IP-адреси на інший порт, програма викриє це, та проведе необхідну перевірку обох портів на вразливості. Також, «розум» даної програми представлений тим, що при наявності анонімного користувача на сервері, що використовував певні порти, у даної програми не буде використано ні секунди часу на перевірку, оскільки результату це не принесе [36].

Покажемо інтерфейс даної програми сканування на рис. 2.3.

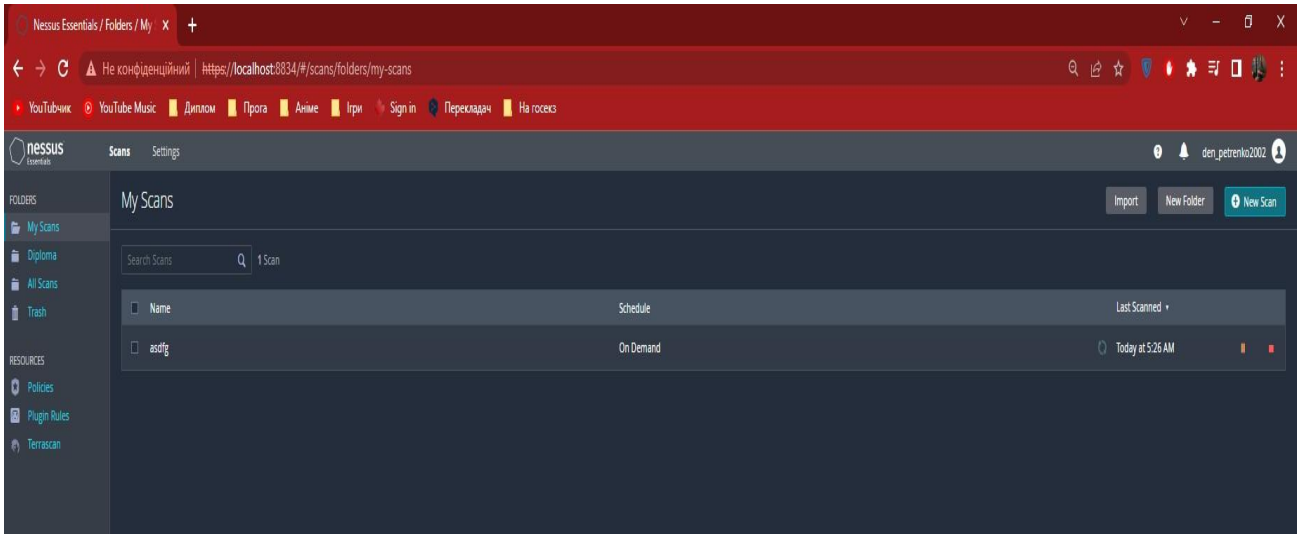


Рисунок 2.3 – Nessus Vulnerability Scanner

Покажемо також результати тестової перевірки для IP-адреси

192.168.1.0 з портом 24 на рис. 2.4:

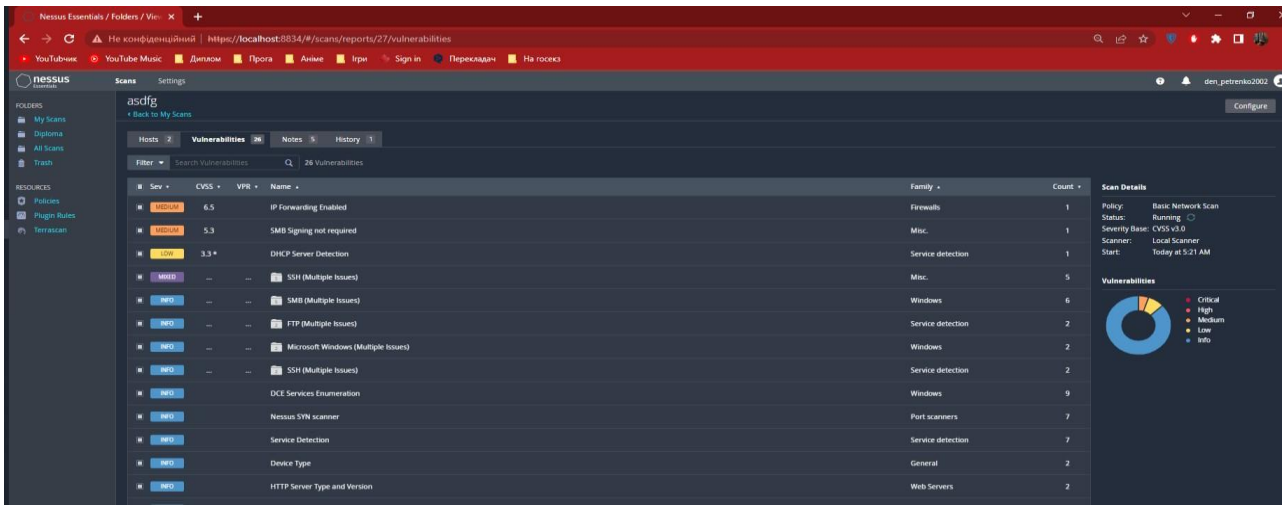


Рисунок 2.4 – Результати роботи Nessus Vulnerability Scanner на цілі

192.168.1.0/24

Розглянемо інтерфейс даної програми. Перша вкладка має назву «Hosts», вона показує список IP-адрес, де за результатами досліджень отримуються ризики, пов’язані з визначеними адресами. Результати є розмежованими на круговій діаграмі різними кольорам (синій, зелений, жовтий, помаранчевий та червоний), що означають різні ступені ризику. Наступним віконцем йде «Vulnerabilities» (укр. вразливості). В ньому доступний список, в якому можна побачити 4 фактори розбиття. Першим фактором є ступені ризику, відображеним кольором та текстовим написом рівня ризику. Правіше знаходить саме наша оцінка CVSS v3.1, потрібна для розрахунків захищеності. Наступний – це причина уразливості. При кліці на дану причину, користувач зможе побачити детальну інформацію про вразливість, та у вкладці «Solution» отримати варіант вирішення даної проблеми захищеності. За ним йде пункт «Family», що показує плагін, до якого підключена наша система. Останнім йде пункт таблиці «Count», що вказує нам на кількість підключень до кожного з плагінів «Family». Далі – віконце «Notes», яке видає більш розгорнуту інформацію по ходу сканування, з указанням певних проблем.

Останнім віконцем йде «History». В ньому зберігається інформації про всі проведені сканування, їх результати та вирішення проблем. Всю інформацію отриману з даного віконця можливо експортувати в різних розширення файлів, натиснувши кнопку «Експорт».

Характеристика сканера показує, що він є одним з кращих по функціоналу, легким в розумінні та потрібним для розв'язання поставленої задачі дипломної роботи.

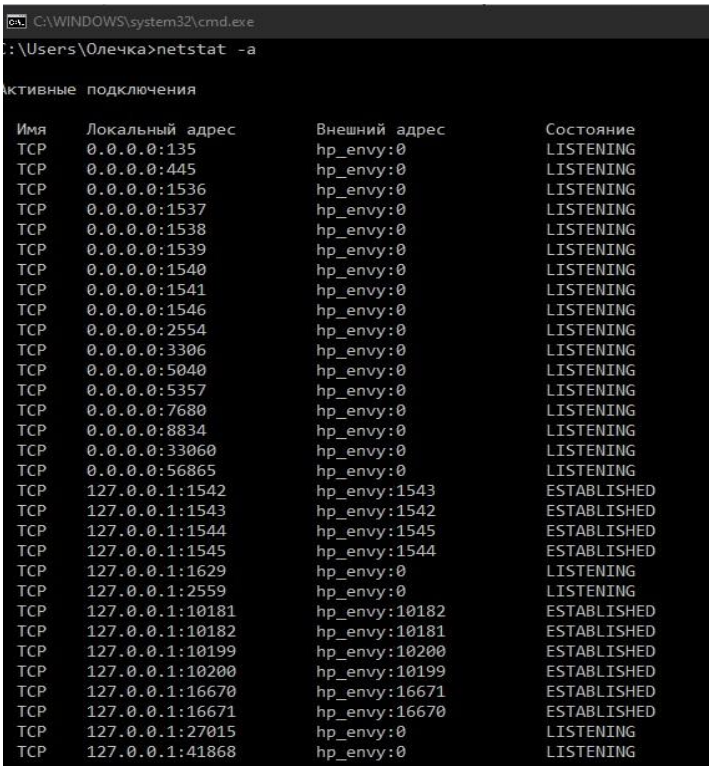
Дана програма буде робити «розумний» аналіз портів IP-адрес, отриманих двома іншими програмами, які згадувались в цьому пункті. Утиліта дає всі необхідні інформативні знання, для проведення особистого розрахунку захищеності комп'ютерних систем.

2.3 Постанова задачі експериментального дослідження

До даної дипломної роботи мною поставлена задача проаналізувати захищеність мережі моїх комп'ютерів на операційних системах Windows 10 версії 21H2 та Windows 10 версії 22H2, визначивши всі необхідні параметри та прорахувавши всі значення за допомогою заданих формул, в кінці зробивши висновок про захищеність моєї інформаційної системи.

Створимо алгоритм розв'язку даної задачі аналізу захищеності розподілених та масштабованих комп'ютерних систем [37 - 40]:

- Першим кроком алгоритму буде проведено аналіз системи за допомогою командної строки cmd.exe, визначивши присутні IP-адреси досліджуваної комп'ютерної системи. Це виконується за допомогою текстової команди в командну строку «netstat -a». Дана команда виводить всі з'єднання та порти, що використовуються інформаційною системою, включаючи прослуховування портів. Приклад виконання можна побачити на рис. 2.5.



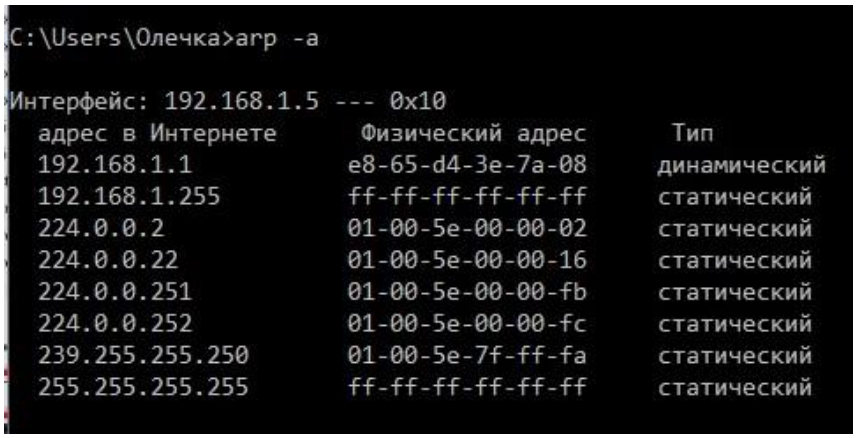
```
C:\WINDOWS\system32\cmd.exe
:\Users\Олечка>netstat -a

ктивные подключения

  Имя    Локальный адрес        Внешний адрес          Состояние
  TCP    0.0.0.0:135            hp_envy:0              LISTENING
  TCP    0.0.0.0:445            hp_envy:0              LISTENING
  TCP    0.0.0.0:1536           hp_envy:0              LISTENING
  TCP    0.0.0.0:1537           hp_envy:0              LISTENING
  TCP    0.0.0.0:1538           hp_envy:0              LISTENING
  TCP    0.0.0.0:1539           hp_envy:0              LISTENING
  TCP    0.0.0.0:1540           hp_envy:0              LISTENING
  TCP    0.0.0.0:1541           hp_envy:0              LISTENING
  TCP    0.0.0.0:1546           hp_envy:0              LISTENING
  TCP    0.0.0.0:2554           hp_envy:0              LISTENING
  TCP    0.0.0.0:3306           hp_envy:0              LISTENING
  TCP    0.0.0.0:5040           hp_envy:0              LISTENING
  TCP    0.0.0.0:5357           hp_envy:0              LISTENING
  TCP    0.0.0.0:7680           hp_envy:0              LISTENING
  TCP    0.0.0.0:8834           hp_envy:0              LISTENING
  TCP    0.0.0.0:33060          hp_envy:0              LISTENING
  TCP    0.0.0.0:56865          hp_envy:0              LISTENING
  TCP    127.0.0.1:1542         hp_envy:1543           ESTABLISHED
  TCP    127.0.0.1:1543         hp_envy:1542           ESTABLISHED
  TCP    127.0.0.1:1544         hp_envy:1545           ESTABLISHED
  TCP    127.0.0.1:1545         hp_envy:1544           ESTABLISHED
  TCP    127.0.0.1:1629         hp_envy:0              LISTENING
  TCP    127.0.0.1:2559         hp_envy:0              LISTENING
  TCP    127.0.0.1:10181        hp_envy:10182          ESTABLISHED
  TCP    127.0.0.1:10182        hp_envy:10181          ESTABLISHED
  TCP    127.0.0.1:10199        hp_envy:10200          ESTABLISHED
  TCP    127.0.0.1:10200        hp_envy:10199          ESTABLISHED
  TCP    127.0.0.1:16670        hp_envy:16671          ESTABLISHED
  TCP    127.0.0.1:16671        hp_envy:16670          ESTABLISHED
  TCP    127.0.0.1:27015        hp_envy:0              LISTENING
  TCP    127.0.0.1:41868        hp_envy:0              LISTENING
```

Рисунок 2.5 – Результати роботи cmd.exe від команди «netstat -a»

Даної команди не достатньо, оскільки під час її виконання, не виводяться всі присутні IP-адреси. Для цього є ще одна команда для cmd.exe «arp -a» - отримання всіх IP-адрес пристроїв в локальній мережі. Приклад виконання на рис. 2.6.



```
C:\Users\Олечка>arp -a

Интерфейс: 192.168.1.5 --- 0x10
  адрес в Интернете      Физический адрес      Тип
  192.168.1.1           e8-65-d4-3e-7a-08     динамический
  192.168.1.255         ff-ff-ff-ff-ff-ff     статический
  224.0.0.2             01-00-5e-00-00-02     статический
  224.0.0.22            01-00-5e-00-00-16     статический
  224.0.0.251           01-00-5e-00-00-fb     статический
  224.0.0.252           01-00-5e-00-00-fc     статический
  239.255.255.250       01-00-5e-7f-ff-fa     статический
  255.255.255.255       ff-ff-ff-ff-ff-ff     статический
```

Рисунок 2.6 – Результати роботи cmd.exe від команди «arp -a»

Дана інформація потрібна для визначення відкритих портів наступного кроку.

 Другим кроком алгоритму буде знаходження всіх відкритих портів за допомогою програмного елементу Nmap v7.93 – Zenmap GUI. Дана програма дає можливість виділити всі відкриті порти TCP, UDP, ICMP, тощо. Це потрібно для того, щоб мати змогу скористатися сканером вразливостей, який працює на даних отриманих на першому і другому кроках. Приклад роботи даної програми у текстовій та візуальній версіях показано на рис. 2.7 та рис. 2.8:

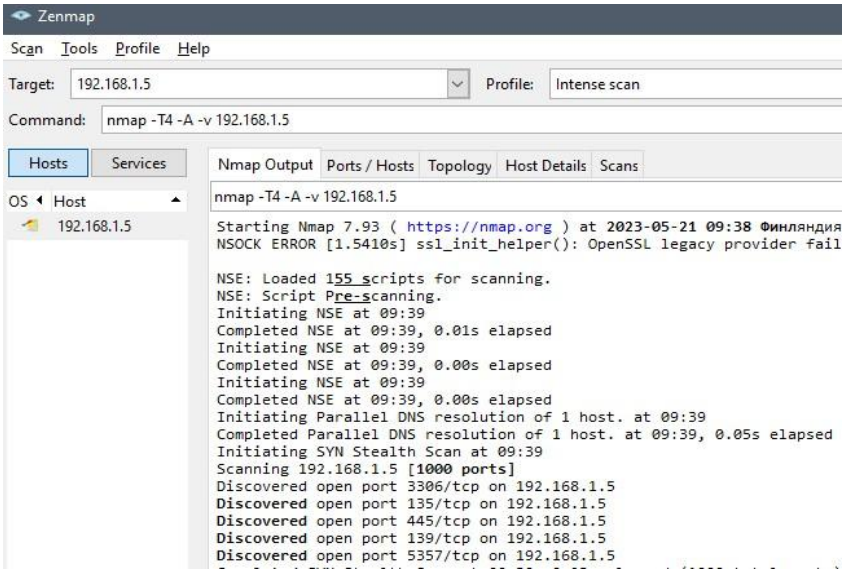


Рисунок 2.7 – Результати роботи Nmap v7.93 – Zenmap GUI на цільове IP-

адресу 192.168.1.5 (текстовий вивід)

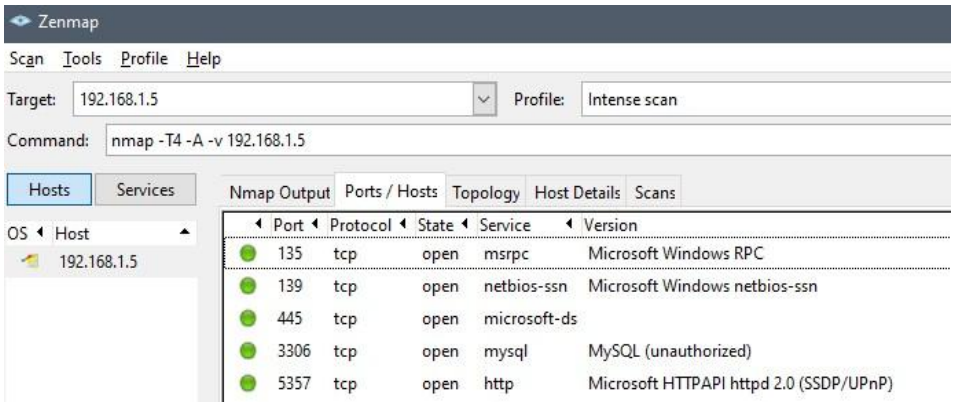


Рисунок 2.8 – Результати роботи Nmap v7.93 – Zenmap GUI на цільову IP-

адресу 192.168.1.5 (візуальний вивід)

 Третім кроком є сканування отриманих даних попередніх етапів за допомогою сканера вразливостей Nessus Vulnerability Scanner. Даний програмний продукт перевіряє всі можливі порти та виводить оцінку CVSS.

Дані оцінки будуть використані в подальшому для фінального обчислення значень вразливостей. Робота Nessus Vulnerability Scanner показана на рис.

2.9.

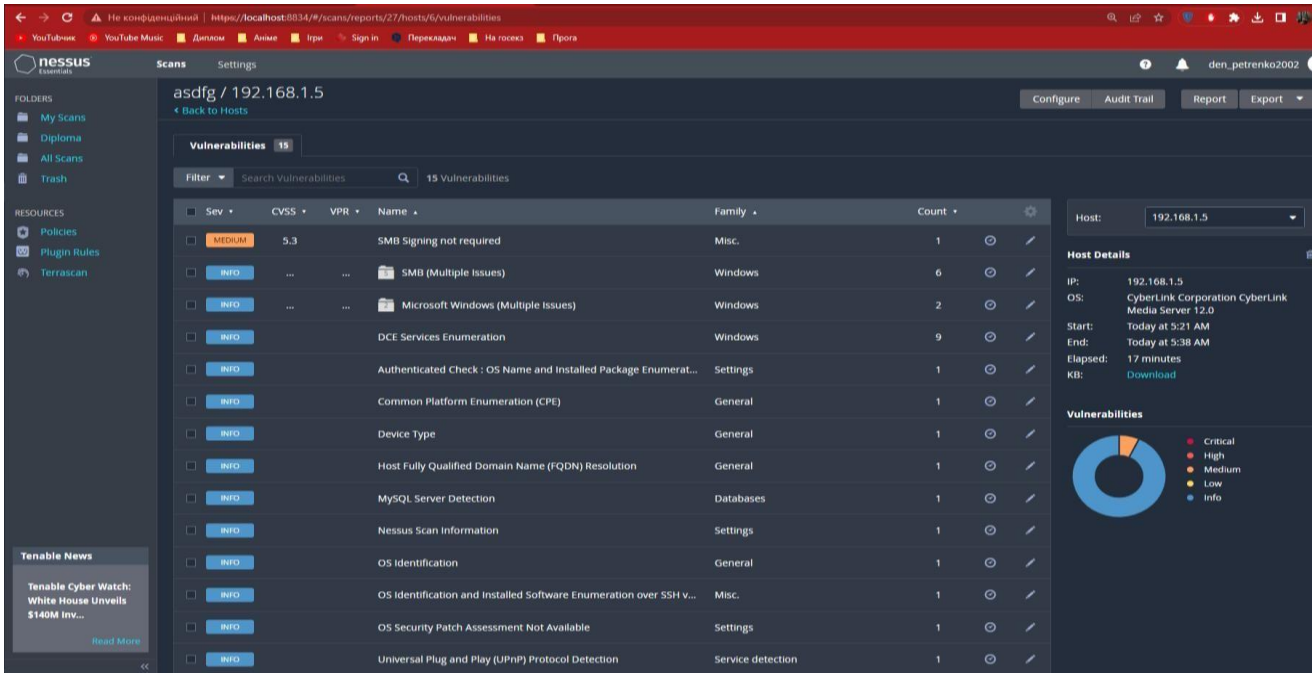


Рисунок 2.9 – Результати роботи Nessus Vulnerability Scanner на цільову IPадресу з портом 192.168.1.5/135

 Четвертим кроком є використання методології оцінки CVSS. Використавши значення отримані з попереднього кроку, буде прораховано значення захищеності представленою оцінкою CVSS. Додатково з цим буде використаний калькулятор CVSS для більш легкого підрахунку

отриманих даних. Отримані дані вносяться в таблицю, де з ними відбуваються певні перетворення, для легкості підрахунків.

 П’ятим кроком є розрахунок остаточних оцінок по кожному з отриманих на попередньому кроці значень. Це буде виконано за допомогою використання формули ризику OWASP та документації CVSS v3.1, що приведе до бажаних відповідей.

 Шостим кроком є створення висновку з отриманих аналітичних значень, що допоможуть визначити, наскільки захищеною є розглянута комп’ютерна система.

Нижче створимо UML-діаграму ходу роботи, для візуального розуміння та представлення кроків роботи плану перевірки захищеності системи. Дана діаграма буде показана на рис. 2.10.

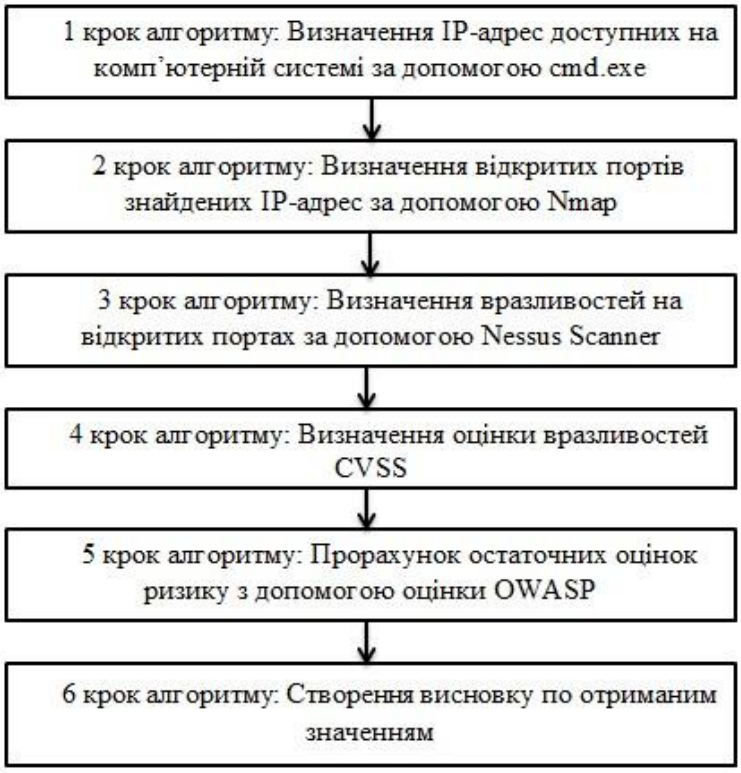


Рисунок 2.10 – UML-діаграму ходу роботи перевірки захищеності системи.

2.4 Висновки до розділу 2

У другому розділі дипломної роботи описуються сучасні програмні засоби для оцінки захищеності комп'ютерних систем. Проаналізовано існуючі застосунки для аналізу захищеності та виділено три найголовніші: сканери портів, сканери вразливостей та системи моніторингу захищеності. Дано повну характеристику даним утилітам та приведено вичерпні приклади програм для розуміння їх концепцій, а головне кількості даних систем в сучасному світі. Кожна зі згаданих програм є унікальною, зі своїми особливостями та перевагами.

З поміж усіх представлених програм було виділено три найважливіші та найбільш підходящі програми для виконання поставленої задачі дипломної роботи. До даних програм мною було віднесено: командна строка Windows 10 (cmd.exe), програма для виявлення мережі та аудиту безпеки

Nmap v7.93 – Zenmap GUI та сканер вразливостей системи Nessus

Vulnerability Scanner.

Перша програма є програмною утилітою операційної системи Windows, яка отримує інформацію про відкриті та присутні діапазони IPадрес.

Друга програма використовує інформацію з першої для того щоб, проаналізувавши отримані дані, знайти всі можливі відкриті порти внутрішніх та мережевих підключень нашої комп'ютерної системи.

Третя програма є найважливішою, оскільки вона виконує найбільшу роботу. Отримавши всі дані про порти з програми Nmap v7.93 – Zenmap GUI, Nessus Vulnerability Scanner починає перевіряти дану йому IP-адресу з портом на всі можливі уразливості, зберігаючи всю потрібну для роботи інформацію.

В розділі описано постанову задачі експериментального дослідження, яка полягає в використанні всіх вище зазначених програм, виділення всієї потрібної інформації та з використанням формул оцінок захищеностей розподілених та масштабованих систем, утворених методологіями CVSS та OWASP. Отримавши певні значення, внесемо їх у таблицю та створюємо окрему, відмінну від інших оцінку та робимо висновок по виконаній роботі. Також, було створену UML-діаграму ходу роботи, що представляє собою покроковий перехід з одного кроку на інший, візуально легкою для сприйняття з даними переходами. Головним є те, що саме такі діаграми дозволяють легко та швидко розуміти хід роботи алгоритму для виконання певного поставленого завдання.

РОЗДІЛ ТРЕТІЙ: ІМПЛЕМЕНТАЦІЇ І ІДЕЇ ПОКРАЩЕННЯ

3.1 Прототип, розроблений для роботи

Для ознайомлення з імплементацією алгоритму LSB і можливості його протестувати, було розроблено програмний прототип аплікації кодування повідомлення в картинку на операційну систему Windows, згідно з методом, описаному в другому розділі.

Уявімо наступний сценарій: нам необхідно зберігати інформацію критичної важливості (наприклад, власні паролі) – або обмінюватись нею в рамках контрольованого нами, але не цілковито захищеного стеганографічного каналу (наприклад, фізичне передання USB-флешнакопичувача з “галереєю” заповнених картинок з рук в руки). В будь-який момент в нас можуть запросити доступ до канала передачі або до нашого пристрою, з якого ми проводили кодування чи декодування. Наша мета – замаскувати комунікацію на всіх етапах передачі і залишати факт проведення комунікацій невикритим і не запідозрюваним. Фінальний протокол має включати в себе послідовність дій, модифікацію коду наданої аплікації і патерни поведінки.

Розглянемо структуру аплікації. Вона розділена на два основні модулі, кодування і графічного інтерфейсу. Графічний інтерфейс в даному випадку є мінімалістичним і має лише виконувати функцію для ознайомлення. Запустивши аплікацію командою “python gui.py”, де python - системна змінна, що веде до шляху коду версії пітона на пристрої використовуючого. Ми бачимо мінімалістичне і інтуїтивне головне меню, єдині два вікна якого дозволяють нам закодувати чи декодувати світлину. (рис. 3.1)

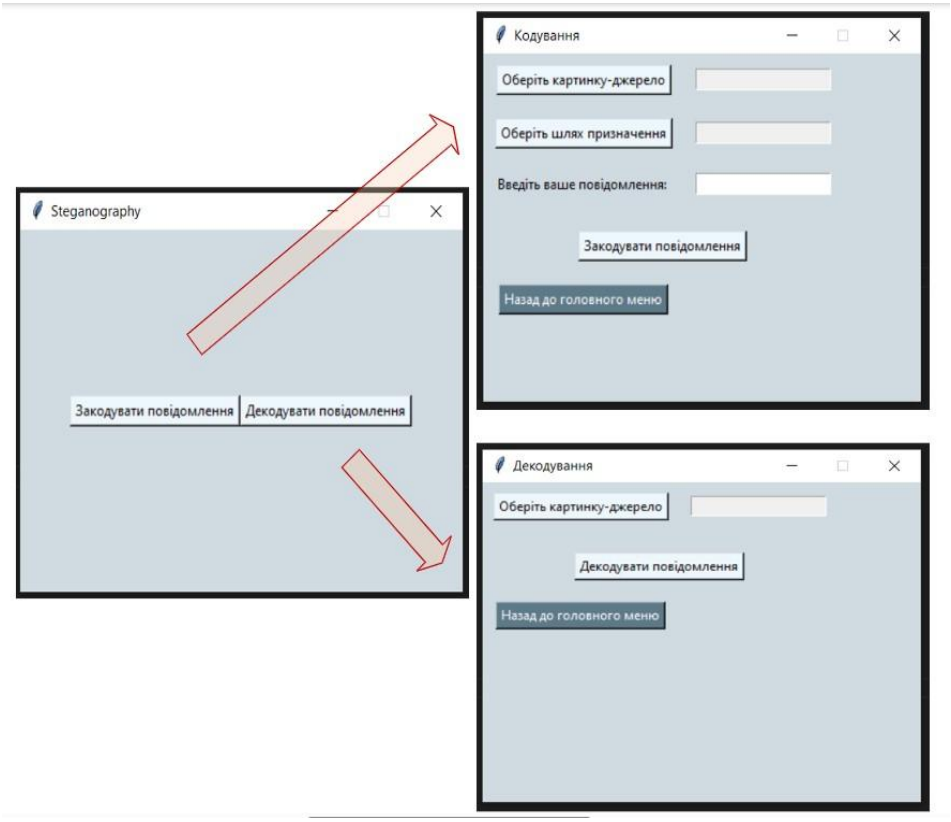


Рисунок 3.1 – інтерфейс прототипу 1

Перед використанням застосунку, користувач має обрати, як зазначено раніше, унікальні стеганоконтейнери, які будуть використані для передання (зберігання) інформації. Хоча ми будемо працювати таким чином, щоб зміни здавались випадковими і в крайньому випадку звинувачувати в них неточність аналогового сигналу, або погрішність, створену компресією файлу, найкращим варіантом буде, щоб файл-контейнер секретної інформації після шифрування став єдиним екземпляром цього файлу. Як файл-контейнер ідеально використовувати звичайне, непідозріле фото, зроблене власними руками – наприклад, свіжа фотографія домашнього улюбленця, їжі або селфі. Після шифрування в це фото повідомлення, оригінал необхідно видалити.

В програмі обираємо пустий стеганоконтейнер як джерело, і задаємо повний шлях файлу призначення.

Далі, в програмі же задаємо повідомлення, яке збираємося приховувати – для приклада, просте текстове повідомлення (про інші можливості див. Розділ 3.2). Як ми визначили в першому розділі, чим коротші дані, тим легше і ефективніше їх можна приховати, а за другим визначили, які саме найкращі для приховування. Також увага до того, що повідомлення, яке вводиться, має бути вже зашифрованим обраним користувачем криптографічним методом – сам застосунок не імплементує криптографії. Кодування символів відбувається за UTF-8, що підтримує кирилицю.

Алгоритм кодування використовує бібліотеку для python PIL (Pillow), щоб працювати з пікселями завантаженого повідомлення. Також в біти, які не затронуті кодуванням повідомлення, додається шум – для цього використовується бібліотека random.

В індикованому місці коду задана функція, яка використовується для кодування повідомлення в картинку і слугує селектором бітів повідомлення. В прототипі, для прикладу, це steps(x) = 2x, stepb(x) = 1, що в коді виглядає так:

```
def writable_pixel(i: int) -> bool:
    return (i // 3) % 2 == 0
```

Функція повертає булеве значення, чи поточний піксель стосується повідомлення. Вираз після ‘return’ виражає собою функцію, в нашому випадку, обрання набору з трьох пікселів через один пропуск.

Користувач вільний і навіть має змінити цю функцію на складнішу, яка буде відома лише користувачу і отримувачу (якщо отримувач відрізняється від користувача). В ідеальному випадку варто змінювати функції від повідомлення до повідомлення, щоб атакуючому було важко виявити кореляцію за умови, що він має декілька наших контейнерів і підозрює наявність в них повідомлень. При побудові цієї функції також можно використовувати алгоритми, описані в п 2.2 та 2.3 або інші алгоритми вибору бітів.

В випадку спроби декодування контейнеру, в якому немає повідомлення, алгоритм не видає помилок, а видає той набір випадкових символів, який спробував

Якщо користувач відрізняється від отримувача, то першим етапом протоколу можна вважати обмін ключами іншим методом (або стеганографічним, за допомоги старого відомого ключа).

Робота алгоритма прототипу за замовчуванням досконало представлена на рисунку 3.2 на прикладі ASCII.

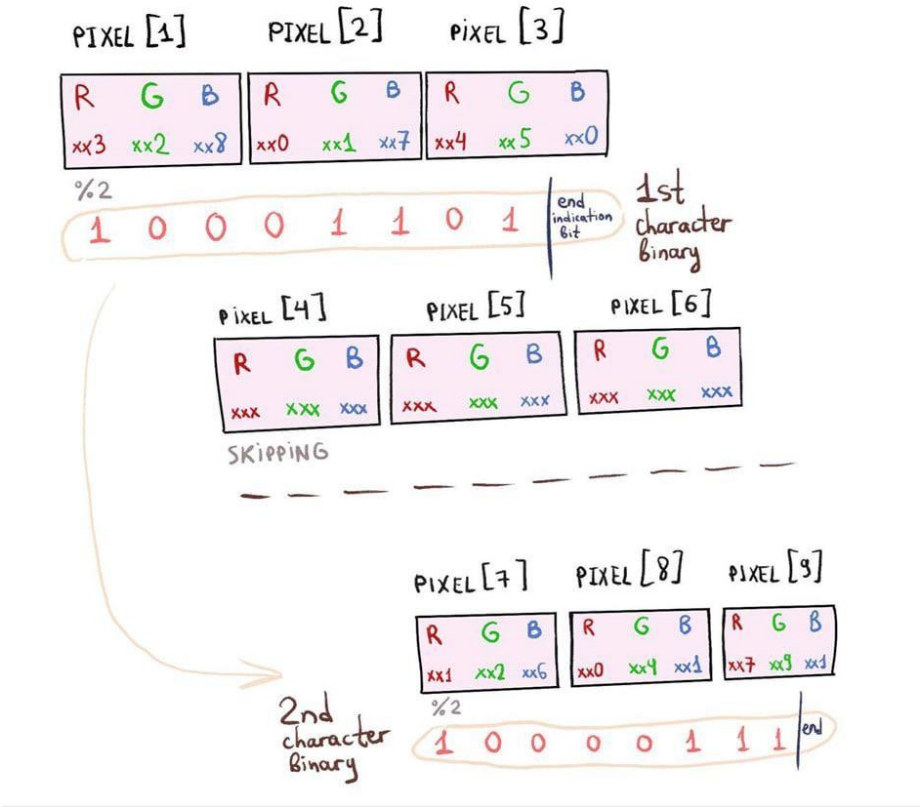


Рис. 3.2 – ілюстрація алгоритму дефолтної функції, вбудованої в прототип 1

Але залишається проблема того, що якщо програма для кодування буде в нас викрита, то це одразу підвищить підозру і може спровокувати атакуючого більш обережно і досконало шукати в нас інформацію, що може призвести до

викриття комунікації. Як було зазначено в розглядаємому сценарії, атакуючий може в будь-який момент отримати доступ до всіх наших пристроїв. Повний протокол має не лише працювати з стеганографічним каналом, а й з не гарантовно захищеними кінцями передачі.

Розглянемо можливі легко імплементуємі модифікації протоколу.

По-перше, щоб дистанціювати компрометуючу програму від нашого пристрою, переносимо її на USB-флеш-накопичувач, за її природою вона портабельна. Однак цього, звичайно, все ще недостатньо.

По-друге, важливим кроку протоколу буде розділення модулів графічного інтерфейсу і кодування. Графічний інтерфейс для атакуючого не представляє ніякої суттєвої цінності, і за легких модифікацій може бути схожим на будь-яку програму криптографічного захисту.

Повертаємось до того, що програма призначена бути використаною на Windows, оскільки більшість не-підозрілих користувачів мають саме її як операційну систему. Це грає нам на руку.

Далі, все ще як частина протоколу, пропонується перенести зміст файла з алогритмом, тобто всі його функції, в будь-який файл бібліотек справжнього пітона на пристрої користувача – в ідеальному випадку також перейменовуючи функції кодування. Необхідно змінити включення файлів графічного інтерфейсу на включення бібліотеки, в яку був скопійований код. Ідеальним також буде кожен раз при використанні стирати лінію коду з підключенням, щоб не давати атакуючому індикацій того, де спробувати шукати алгоритм, а в випадках питань відмовляти, що це “недописана програма в рамках тренувань”.

Таким чином, маємо графічний інтерфейс на USB-флеш-накопичувачі, використання якого залежить від пристрою, який має в собі додатковий невеликий ключ знання того, яку бібліотеку необхідно підімкнути, щоб все запрацювало, і викриття якого не дасть атакуючому ніякої інформації. А наш алгоритм прихований десь глибоко в коді пітону, і навіть за його викриття, для

декодування функція обрання біту в ньому має бути модифікована таким чином, який знає лише користувач (і отримувач, якщо вони різняться) – і немає жодних причин вірити, що ці функції взагалі були ним використані, якщо не порівнювати зі справжнім кодом тої ж версії пітона – тому можна також поквапитись про те, щоб встановити стару або менш відому версію, аніж, наприклад, Python 3.9.

Звичайно, в такому випадку все ще залишається шанс, що комунікація може бути викритою, тому що це не стовідсотково стійка система. Але ймовірність того, що атакуючий, що не має до нас жодних підозр, одночасно вкраде і застосує з нашим пристроєм наш USB-флеш-накопичувач, вирішить просканувати на зміни доволі типові бібліотеки, що мають багато комп'ютерів, і після цього, виявивши функції для стеганографії, вирішить спробувати застосувати їх на світлинах і якимось чином вгадати ключ, а потім розгаданий набір виглядаючий випадковими символів вирішить спробувати розшифрувати відомими криптографічними методами, – вкрай мала.

Отже, навіть такий мінімізований протокол може послугувати великим кроком в захисті критичних даних.

3.2 Ідеї для покращення і перспективи

Запропонований застосунок, звичайно, не є повноцінною реалізацією потенціала методу, або навіть його частиною в класичному розумінні. Існує безліч можливостей модифікувати його таким чином, щоб він ставав ефективніше. Розглянемо кілька цікавих ідей, що вар'юються від простих в імплементації до складних.

Першою ідеєю, простішою, є модифікація генерування шуму і перетворення ключів (інформації про функцію). Як відомо, генератор випадкових чисел, влаштований в пітон, використовує псевдовипадковий алгоритм. Тому, оскільки ми заповнюємо контейнер шумом в усіх бітах, окрім

інформаційних, теоретично атакуючий міг би виявити кореляцію від зворотнього, виявивши справжні біти з повідомленням. Також використання конкретної, навіть складної функції, все одно представляє собою патерн, які програми статистичного аналізу теоретично можуть помітити. Отож, ідеєю буде замість функції використовувати ключ довжини повідомлення, який буде генеруватись з більш стійко випадкових чисел, вигляду “17239834129”, де кожне число (чи два) буде індикувати крок між бітами або символами. Цей ключ має бути переданий іншим (або зазначеним в прикладі методом), але своїм форматом являє собою ідеальний набір даних, які можна сховати.

Наступною ідеєю буде, звичайно, перенесення алгоритму на компільовану мову. Якщо атакуючий ще може подивитись початковий код бібліотек пітону і підвищити рівень підозри після виявлення стеганографічного алгоритму, то якщо функції нашого алгоритму буду лежати в .dll серед бібліотек, наприклад, Qt, або, ще краще, якоїсь гри на нашому пристрої, то він з меншою ймовірністю буде щось підозрювати.

З боку графічного інтерфейсу також можна зробити модифікації, які суттєво підвищать ефективність. Якщо пояснювати присутність на флешнакопичувачі “недописаної програми” може накликати непотрібні підозри, то до, наприклад, “Libre Office Portable” не має виникнути жодних питань. Можна модифікувати відомі open-source застосунки (в ідеальному випадку, якщо вони самі написані на компільованих мовах) таким чином, щоб вони при запуску і при поверхневому аналізі структури, виглядали і працювали, як звичайні. Але, лише користувач знає, що, наприклад, якщо натиснути певну комбінацію клавішу, обравши жирний шрифт, то відкриється графічний інтерфейс застосунку кодування, який був надійно схований всередині.

Останньою ідеєю складних модифікацій буде перенести програму за тим же принципом на мобільний пристрій. Оскільки найкращим контейнером є фото, що існує в одному екземплярі, то ідеальним не привертаючим уваги

графічним інтерфейсом, що запускає кодувальник, буде сам застосунок “камери”. Уявімо наступну систему: спершу, в нотатках на телефоні звичайними словами записане повідомлення. В іншому заданому файлі нотаток, записана, здавалося б, звичайна інформація, але насправді, довжина кожного слова в ній – це ключ того, який крок використовується в обранні біту. Потім, користувач робить фотографію за допомоги модифікованого застосунку камери, що автоматично підвантажує повідомлення з нотаток, інкрустуючи в зроблену фотографію, і за можливості також стираючи початкове повідомлення з нотаток. Таким чином зроблена фотографія автоматично стає єдиним екземпляром і готова для проведення секретних комунікацій в найкоротші строки.

В випадку застосування останньої ідеї, можна також поквапитись про моментальне вивантаження контейнеру з повідомленням в cloud, що може дати змогу провести комунікацію в небезпечних для життя ситуаціях, при цьому роблячи дії, які мають виглядати невинно, такі як письмо у власних нотатках і фотографування себе.

Загалом, стеганографія має неймовірні перспективи. Сучасні досягнення в розвитку моделей штучного інтелекту по сутності можуть дозволити нам автоматизувати шифрування за допомоги бесшумного методу стеганографії (тобто, методу, що не залишає слідів на контейнері) .

Але й в більш вузькому просторі стеганографії картинок мається великий потенціал потенційних досліджень. Вище були розглянуті варіанти для покращення захисту методом змінення протоколу. Але мається також великий потенціал для зберігання величніших обсягів інформації.

Доволі легко уявити собі модифікацію алгоритму таким чином, щоб він послідовно записував дані у серію фотографій – в такому випадку, ми можемо розраховувати на набагато більший простір для інформації. Так, невинний жорский диск з сімейними фотографіями може стати повноцінною секретною

базою даних. Яку саме послідовність необхідно обрати для дешифрування може стати додатковим ключем захисту.

З цікавого також, можна розглянути варіант нелінійного запису даних – до генерації кроків необхідно буде підійти з особливою обережністю, обираючи функцію, яка точно не призведе до обрання одного і того самого біту двічі. Зате, в такому випадку ми отримуємо новий рівень захисту за рахунок того, що можемо розділити передання світлин в кілька різних етапів, а без повного набору неможливо буде відновити початкове повідомлення. Як варіант також, знову ж таки, обирати строго зростаючу або убиваючу функцію, але застосовувати результати почергово до усіх світлин набору (рис. 3.3).

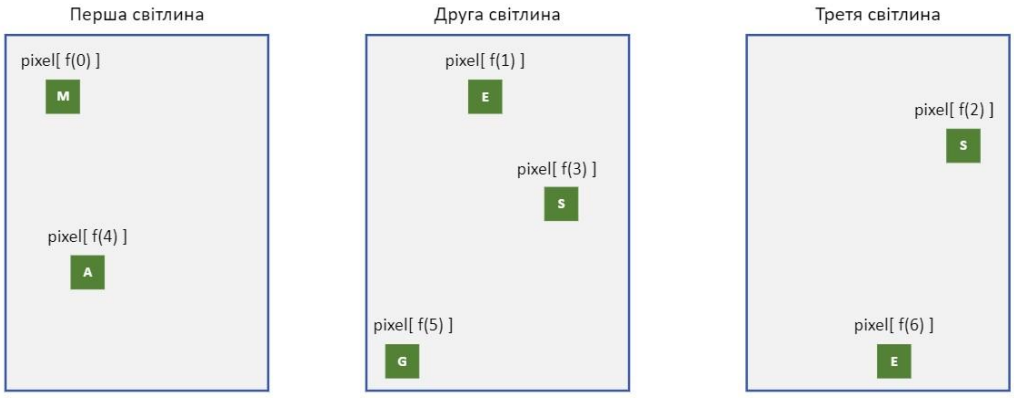


Рисунок 3.3 – Приховування повідомлення в кількох світлинах

Повертаючись до тем розвитку штучного інтелекту, окрім того, з просуванням моделей, їх можна буде використовувати для пошуку патернів і згодом подібні алгоритми і конкретні функції може бути легше виявляти. Але суть з самого початку полягала в тому, щоб додати так багато кроків, щоб навіть якщо в решті решт повідомлення змогли відновити, шлях до нього здавався таким випадковим і складним, що здавалося, що це просто випадковість і можна було продовжувати відмовляти своє до нього

відношення. Оскільки в теорії, таким чином з остаточних бітів світлин можна скласти абсолютно будь-яке повідомлення.

Заплутаність шляху до інформації, хоч і ускладнює нам життя, але додає ефективності, згідно з параметрами методу правдоподібного заперечення.

3.3 Стеганографія та IoT

В процесі роботи і розмірковувань також виникла ідея використання LSB в потоці даних з IoT. Постійна і росповсюджена проблема збору інформації з датчиків для аналізу часових рядів, потенційно прогнозування і детекції аномалій, дозволяє проводити обміни даних в великих кількостях, не викликаючи підозр.

Було розроблено складові сценарію, які передбачені за звичайного функціоналу системи, адаптовані запропонованими позиціями, що дозволять обмінюватись секретними повідомленнями.

Сценарій представляє собою розповсюджену проблему збору інформації про температуру пристроя, яку вивантажують в базу даних на cloud, для подальшого аналізу, візуалізації і налаштування слідкуваннь.

В якості брокера було використано Mosquitto, як розповсюджений вибір для таких цілей. В якості бази даних і провайдера було використано InfluxDB – також як популярний вибір для даних часових рядів. Для достовірності було інтегровано Grafana з попередженнями про перегріви, як в звичайній системі контролю.

Було також розроблено скріпт для імітації даних з датчиків для різних сценаріїв – звичайної роботи, аномального стрибку температури, і перегріву. В випадку з аномальним стрибком, попередження про перегрів не мають на це реагувати, оскільки попередження налаштовані таким чином, що реагують на середнє значення за певний останній проміжок часу.

Всі посередники і проміжкові скрипти були розроблені мовою програмування python, також для побудови реквестів і тестування було використано SQL, а для налаштування Grafana також Flux.

Робота системи в звичайному режимі представляє собою наступне: девайс, за допомоги скрипта, що крутиться на ньому, считує інформацію з датчиків і публікує її на брокер. Опісля, скрипт-підписник брокеру публікує її в базу даних на cloud, до якої законфігурована окремо Grafana виводить графік-звіт роботи, і з якої будуються попередження і розсилаються на пошти відповідальних. Налаштований звіт Grafana виглядає, як показано на рисунку 3.4.

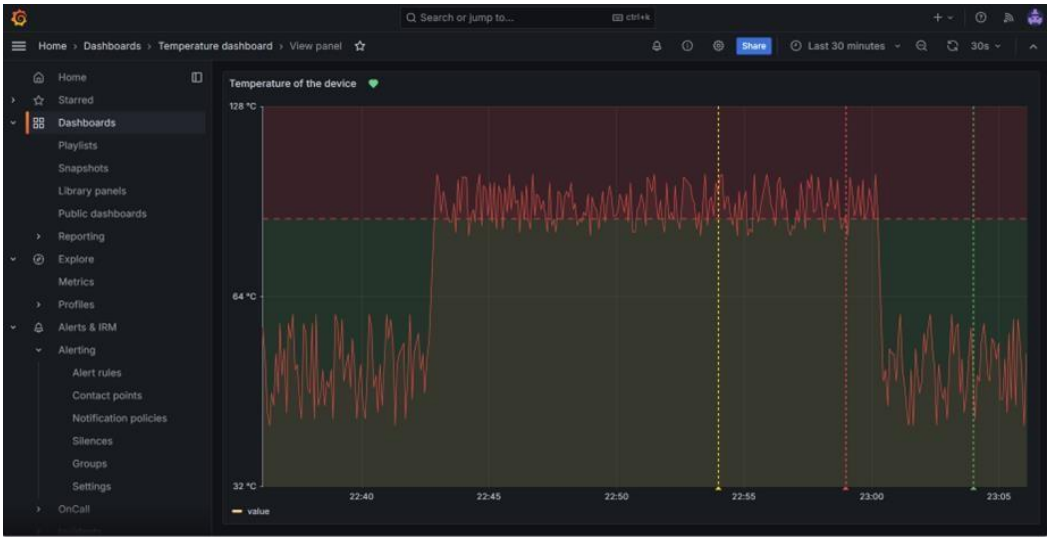


Рисунок 3.4 – Візуалізація даних в Grafana

Інкрустувати інформацію всередину повідомлення пропонується на етапі публікації даних з девайсу, або на етапі передачі їх в базу даних (рис.3.5) – там, де задіяні кастомізовані скрипти. Опісля, будь-хто, хто має доступ до бази даних, може забрати їх звідти селект реквестом і застосувати функцію декодування, де ключем буде таймштамп початку – і можлива функція кроку.

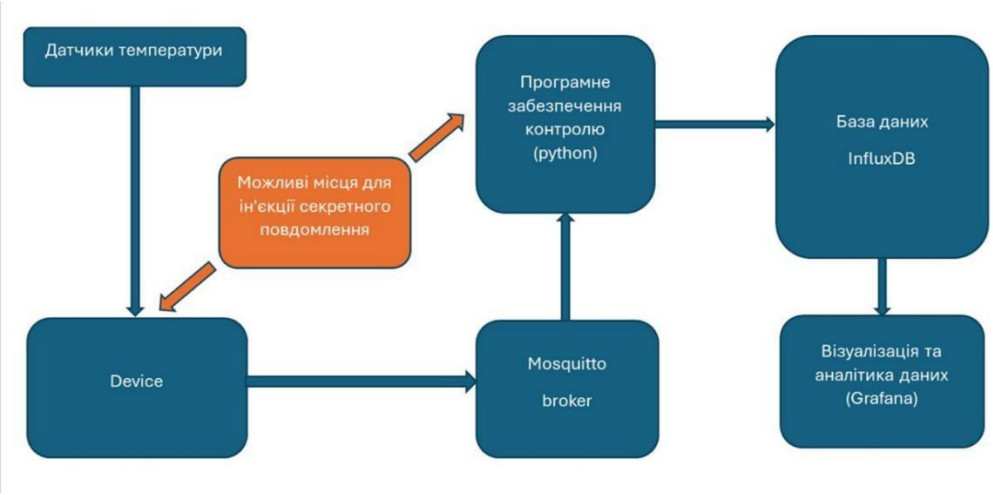


Рисунок 3.5 – схема системи прототипу 2

Влаштовування повідомлення полягає в тому, що на одному з запропонованих етапів ми змінимо парність певного біту значення – в прикладі це в одиницях, але в залежності від точності датчиків конкретної системи, це можуть бути будь-які малі цифри після коми.

Зручність використання стеганографії з IoT буде полягати в тому, що згодом повідомлення саме буде знищене, також за природою даних можна не очікувати пошкоджень повідомлення, і в решті решт, його буде складніше помітити, оскільки мається на увазі великий поток даних, а точність часуключа важко вгадати. Це вид передаваємої на постійній основі інформації, яка не викликає особливих підозр і важко піддається аналізу.

За підсумком, прототип 2 імплементує всі принципи правдоподібного заперечення і може дозволити повноцінну і самодеструктуючуся коммунікацію.

ВИСНОВКИ

Представлена робота спрямована на вивчення стеганографічного методу, креативних способів його застосування і моделей систем, які зможуть дозволити правдоподібне заперечення.

В ході дослідження було розглянуто та проаналізовано теоретичні аспекти методологій оцінки захищеності комп'ютерних систем, різних її варіацій, типи комп'ютерних систем та їх особливості.

Досліджено існуючі загрози інформаційних систем, представлено класифікацію цих загроз та додано пояснення для більш повного розуміння даної теми. На противагу загрозам, було введено поняття про захист інформації та захист інформаційних систем, розглянуто окремі види забезпечення безпеки комп'ютерних систем.

В роботі було описано сучасні програмні засоби для оцінки захищеності комп'ютерних систем. Проаналізовано існуючі застосунки для аналізу захищеності та виділено три найголовніші: сканери портів, сканери вразливостей та системи моніторингу захищеності. Дано повну характеристику даним утилітам та подано вичерпні приклади програм використані в дипломній роботі. Цими програмами є cmd.exe, Nmap v7.93 – Zenmap GUI та Nessus Vulnerability Scanner.

Було створено постанову задачі експериментального дослідження, яка полягає в використанні всіх вище зазначених програм, виділення всієї потрібної інформації та з використанням формул оцінок захищеностей розподілених та масштабованих систем, утворених методологіями CVSS та OWASP отримати остаточні оцінки.

В ході роботи було проаналізовано дві домашні комп'ютерні системи з різними версіями операційної системи Windows 10, а саме 21H2 та 22H2. Результатами аналізу даних систем є виведення оцінки ризику, за допомогою використання власно створеної оцінки захищеності на базі оцінки OWASP.

Результати було розміщено в таблицях, графіках та діаграмах для проведення більш детального та розгорнутого аналізу двох досліджуваних систем. Зроблено ґрунтовний аналіз отриманих даних та представлено висновок по даним системам, на основі отриманих даних.

Отже, результати цього дослідження покликані надати цінну інформацію та практичні рекомендації для оцінювання захищеності комп'ютерних систем. Отримані оцінки захищеності можна використовувати в наступних роботах, присвячених оцінкам захищеності комп'ютерних систем, видаючи актуальні та вичерпні результати для аналізу комп'ютерних систем. За результатами, був розроблений ознайомлювальний програмний прототип, який дозволяє заховати чутливу інформацію у фотографіях, або для подальшого персонального використання (наприклад, зберігання важливих паролів) або для обміну чутливою інформацією. Для цього, за допомоги алгоритма модифікації кольорів LSB, в нього шифрується прихована інформація. Потрібну інформацію зберігає не кожен піксель, а лише обрані за функцією, яка влаштована в програму і має бути задана користувачем. Решта же пікселів теж модифікується, щоб створити «шум». Також було розроблено систему, яка застосовує аналогічний принцип в поєднанні з IoT системами для збору температурних даних, дозволяючи розглядання потокових даних часових рядів як потенційного каналу секретних комунікацій.

В роботі також був проведений докладний аналіз історичного розвитку правдоподібного заперечення в інформатиці, розбір алгоритмів посилення стійкості, і були запропоновані потенційні модифікації системи задля покращення методу. Були також розглянуті перспективи майбутнього і затронуті методи протидії.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Zhang, Q., Jia, S., Chang, B. *et al.* Ensuring data confidentiality via plausibly deniable encryption and secure deletion – a survey. *Cybersecur* 1, 1 (2018). Access mode: https://doi.org/10.1186/s42400-018-0005-8
2. Microsoft Security, What is an authentication? Access mode: https://www.microsoft.com/en-us/security/business/security-101/what-isauthentication (дата звернення: 18.03.24)
3. Global Partners Digital, World map of encryption laws and policies / London, 2022. – Access mode: https://www.gp-digital.org/world-map-of-encryption/#
4. Sahu, Aditya Kumar and Sahu, Monalisa. "Digital image steganography and steganalysis: A journey of the past three decades" *Open Computer Science*, vol. 10, no. 1, 2020, pp. 296-342. Access mode: https://doi.org/10.1515/comp-20200136
5. Abdelrahman Desoky, Noiseless steganography: the key to covert communications / CRC Press , An Auerbach Book , 2012
6. Inas Jawad Kadhim, Prashan Premaratne, Peter James Vial, Brendan Halloran, Comprehensive survey of image steganography: Techniques, Evaluations, and trends in future research, Neurocomputing, Volume 335, 2019, Pages 299-326, ISSN 0925-2312, Access mode: https://doi.org/10.1016/j.neucom.2018.06.075
7. Henry Clinton Mask Letter to John Burgoyne, August 18, 1777. Henry Clinton Papers / William L. Clements Library, University of Michigan. Access mode: https://clements.umich.edu/exhibit/spy-letters-of-the-americanrevolution/gallery-of-letters/clinton-burgoyne-mask-letter/
8. krypt3ia, “Al-Qaida goes “Old School” With Tradecraft and Steganography”, 13.03.2010, Access mode: http://krypt3ia.wordpress.com/2010/03/13/al-qaidagoes-old-school-with-tradecraft-and-steganography/
9. Bailey, Karen and Kevin Curran. “An evaluation of image based steganography methods.” *Multimedia Tools and Applications* 30 (2006): 55-88.

10. Wang, Yongjie & Stojiljković, Nina & Jehle, Johannes. (2016). 2010-Wang et al-J Virol Methods.

11. M. Sownya, G. Manikandan, Clustering based steganographic approach for secure data transfer / Contemporary Engineering Sciences, Vol. 8, 2015, no. 12, 525-531