Федеральное государственное автономное образовательное учреждение высшего образования

«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ «ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»

Московский институт электроники и математики им. Тихонова

Департамент электронной инженерии

ОТЧЕТ

О ПРАКТИЧЕСКОЙ РАБОТЕ №3

по дисциплине «Программные и аппаратные средства защиты информации»

«Стеганографичекие средства защиты информации»

Студент гр. БИБ201

Шадрунов Алексей

Дата выполнения: 12 марта 2023 г.

Преподаватель:

Перов А. А.

«___» _____ 2023 г.

Москва, 2023

Содержание

1 Цель работы

Целью данной лабораторной работы является изучение программно-аппаратных средств стеганографии. Изучить возможности и практику работы с ImageSpyer и RedJPEG. Написать собственное программное средство для внедрения сообщения.

2 Ход работы

2.1 Image Spyer G2

Установим Image Spyer G2 на виртуальную машину. Окно программы приведено на рисунке 1.

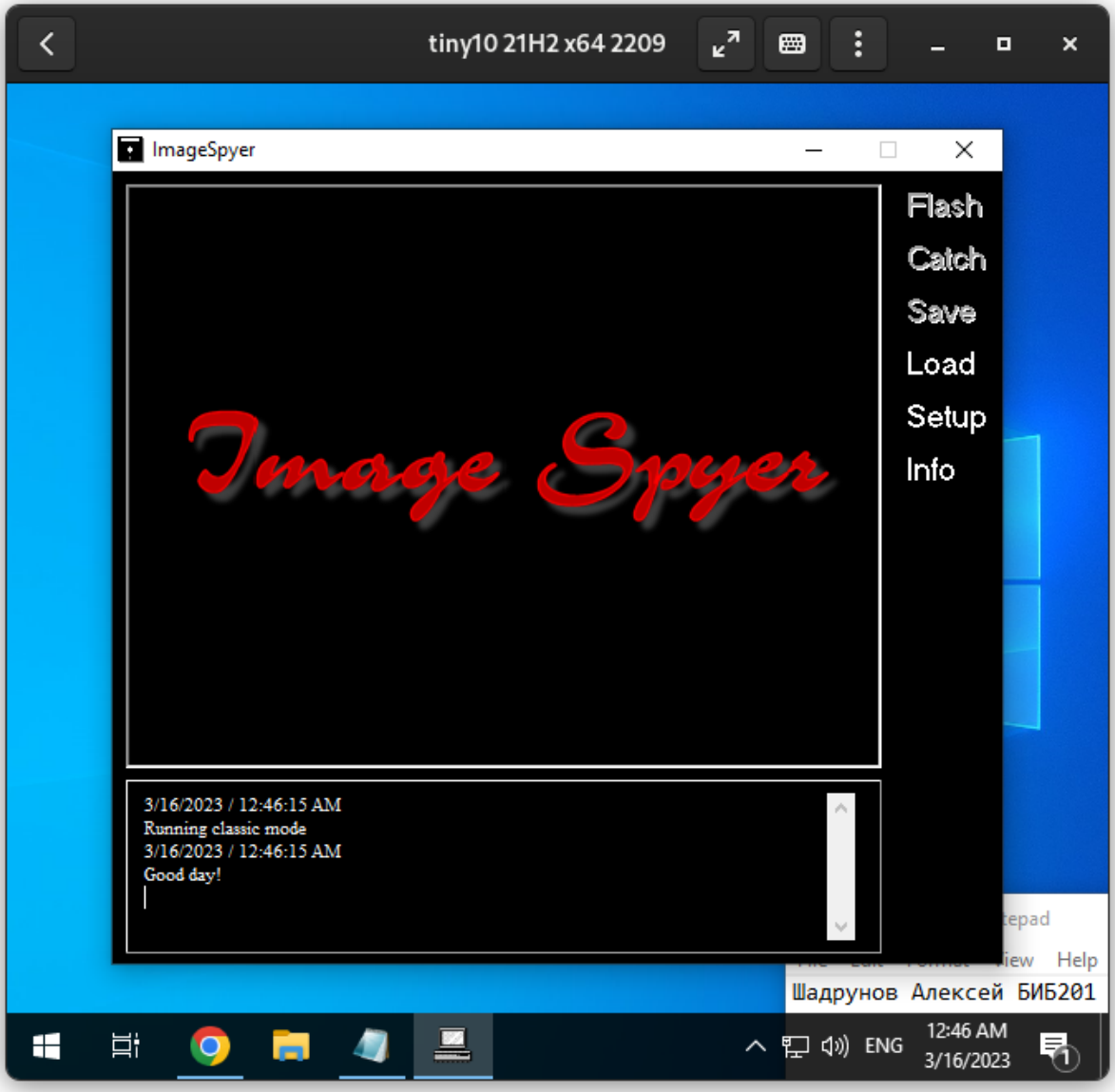

Рисунок 1 – Окно программы

Сначала встроим текстовый файл в картинку (рисунки 2-7). Затем встроим полученную картинку в оригинал (рисунок 8).

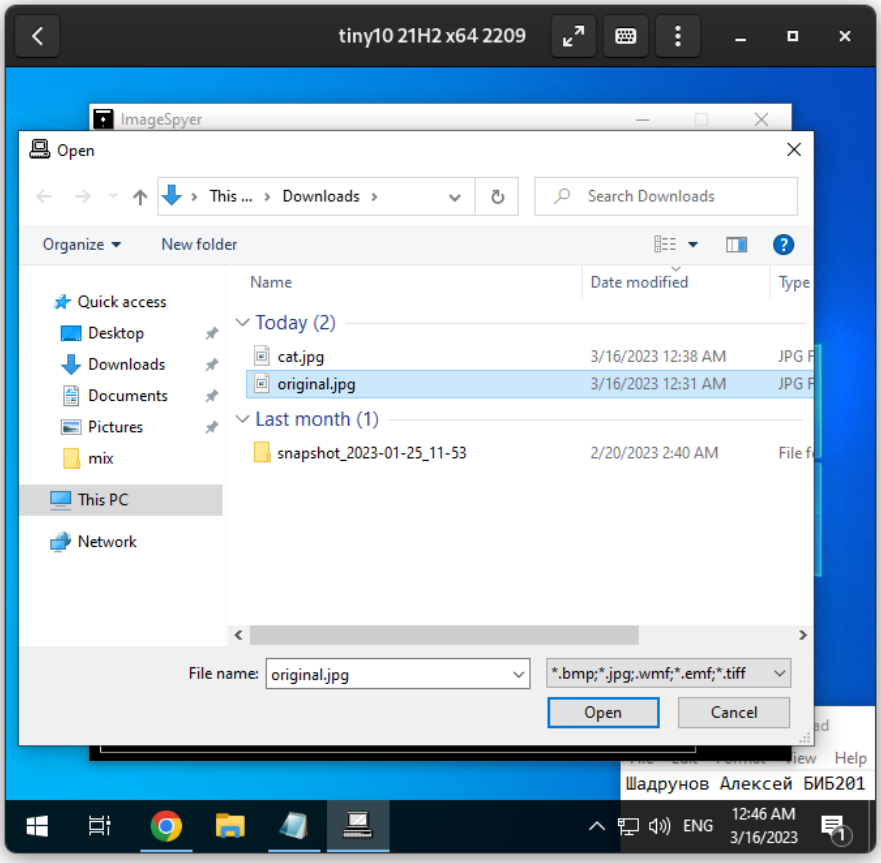

Рисунок 2 – Выбор начальной картинки

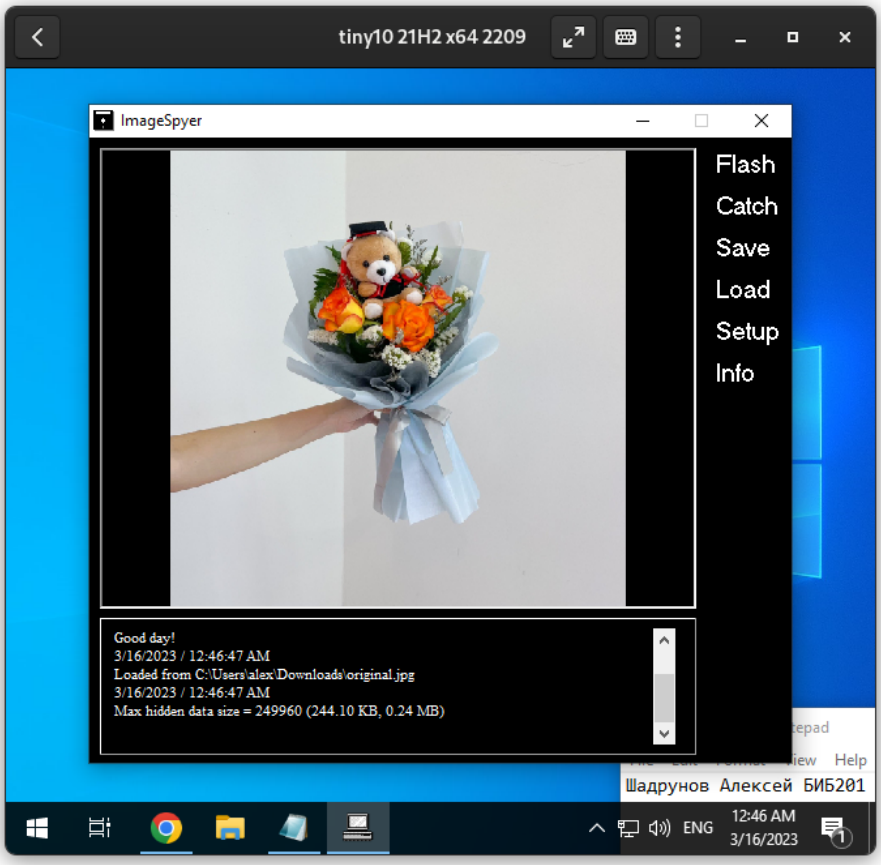

Рисунок 3 – Начальная картинка

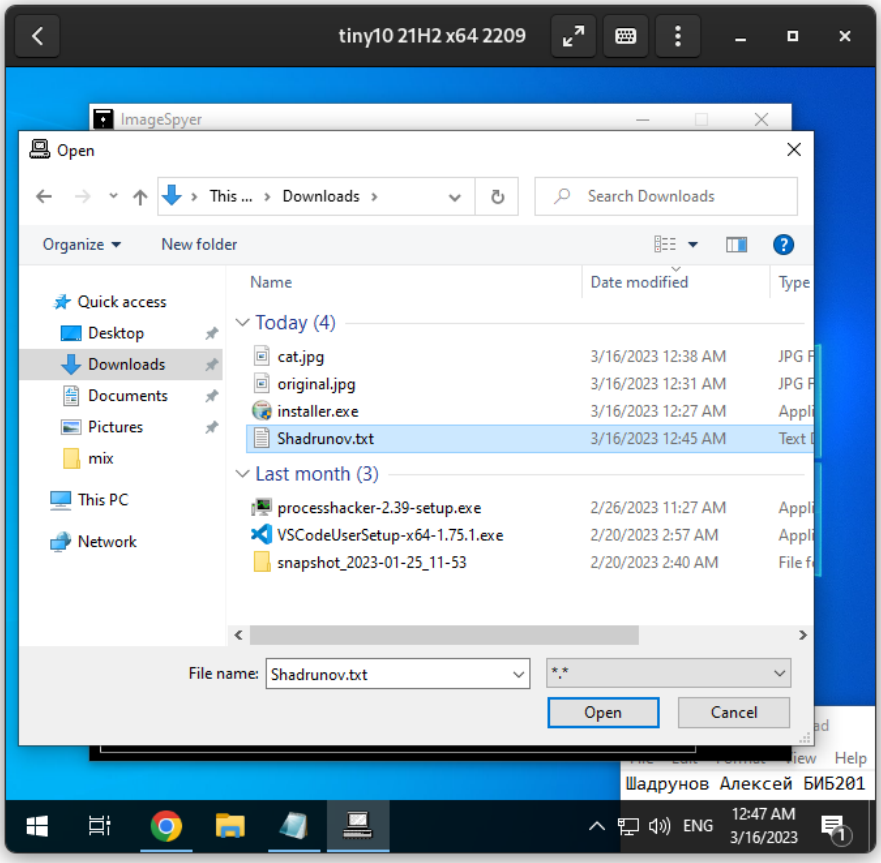

Рисунок 4 – Выбор текстового файла

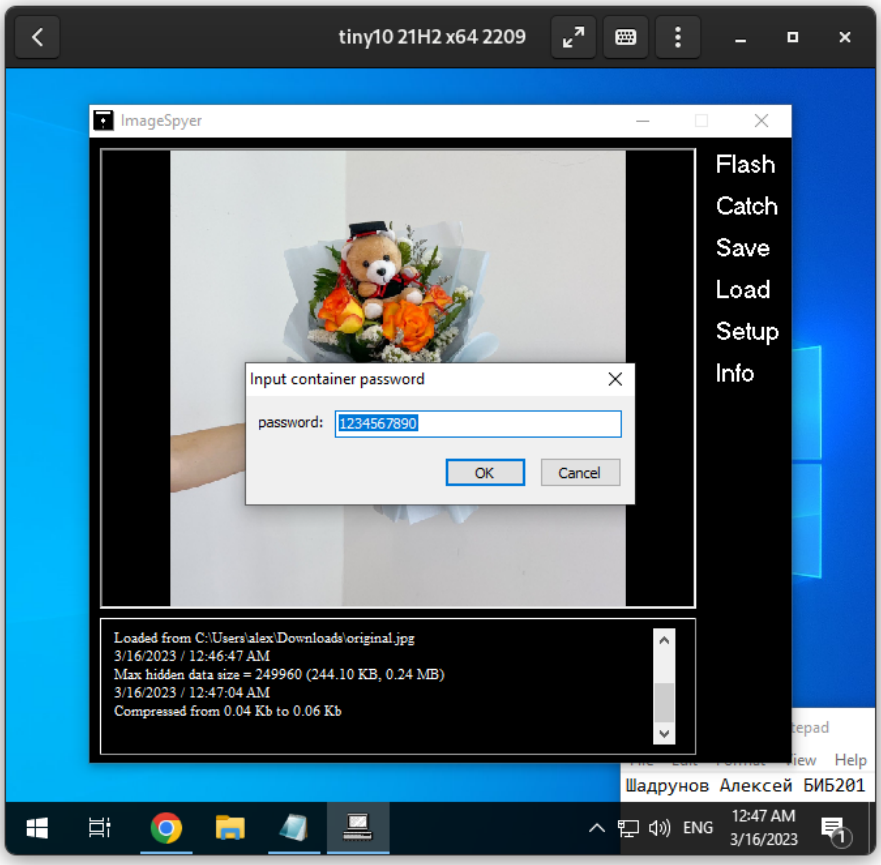

Рисунок 5 – Задание пароля

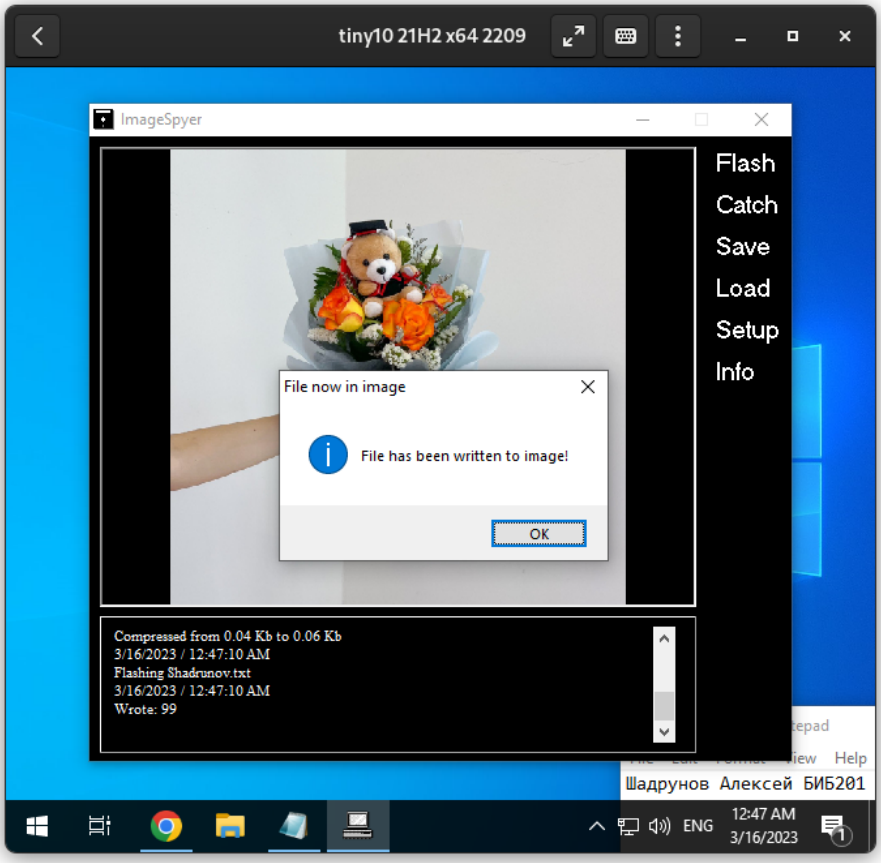

Рисунок 6 – Успешное встраивание

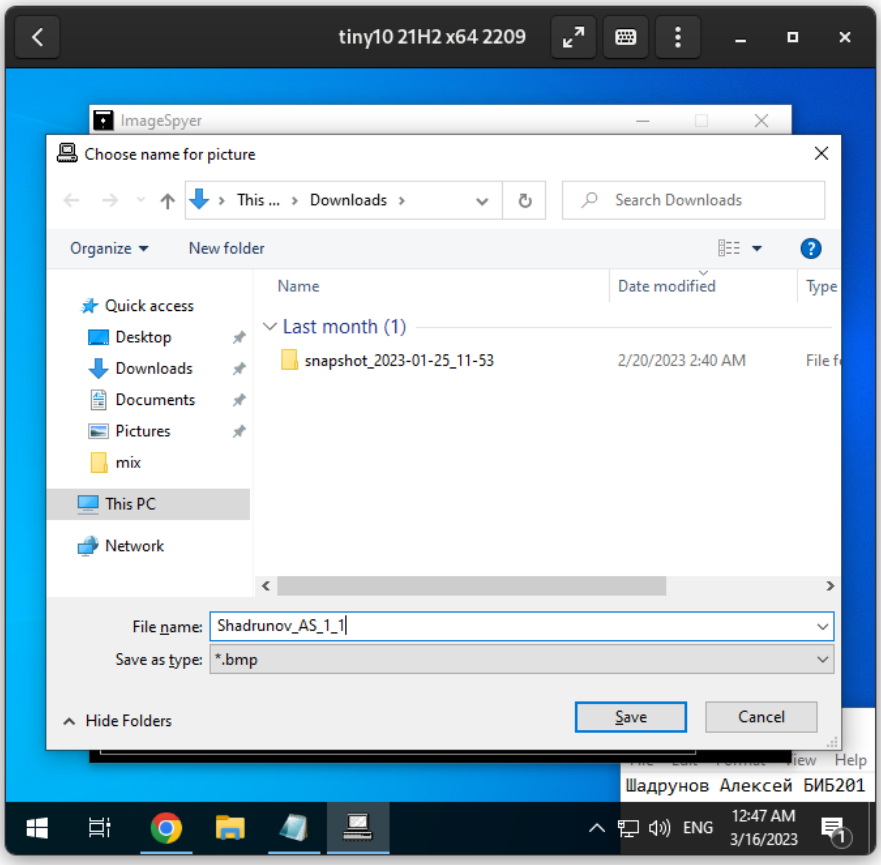

Рисунок 7 – Сохранение файла

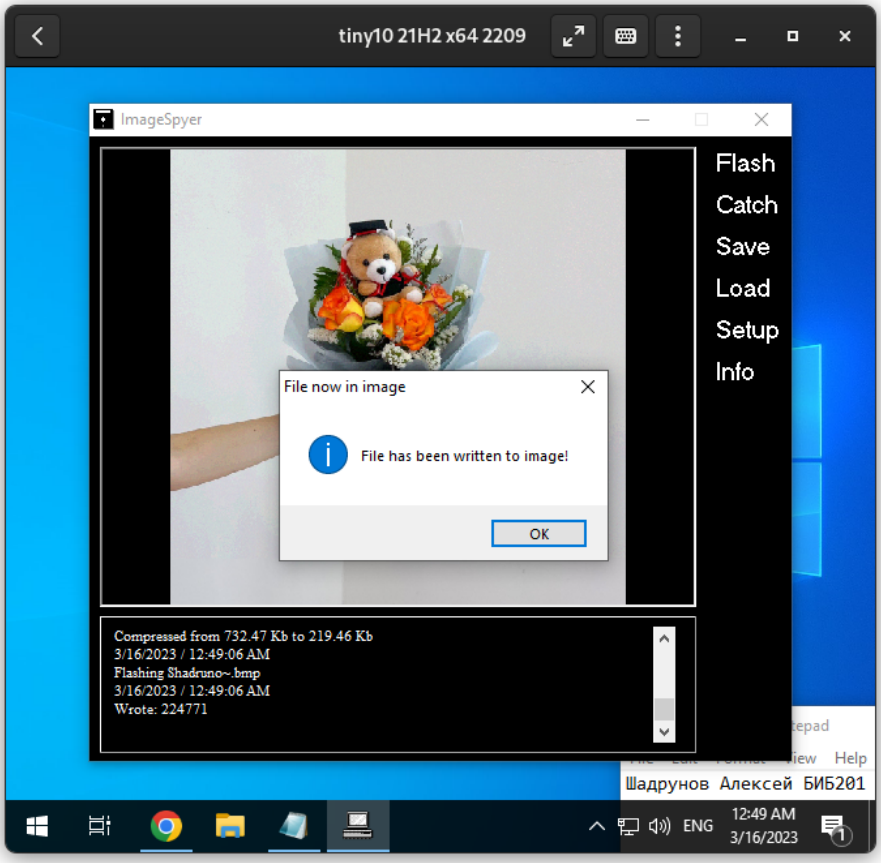

Рисунок 8 – Другой файл встроили в картинку

Результат встраивания отображён на рисунках 9 и 10.

Рисунок 9 – Текстовый файл в картинке

Рисунок 10 – Картинка в картинках

2.2 RedJPEG

Установим RedJPEG на виртуальную машину. Окно программы приведено на рисунке 11.

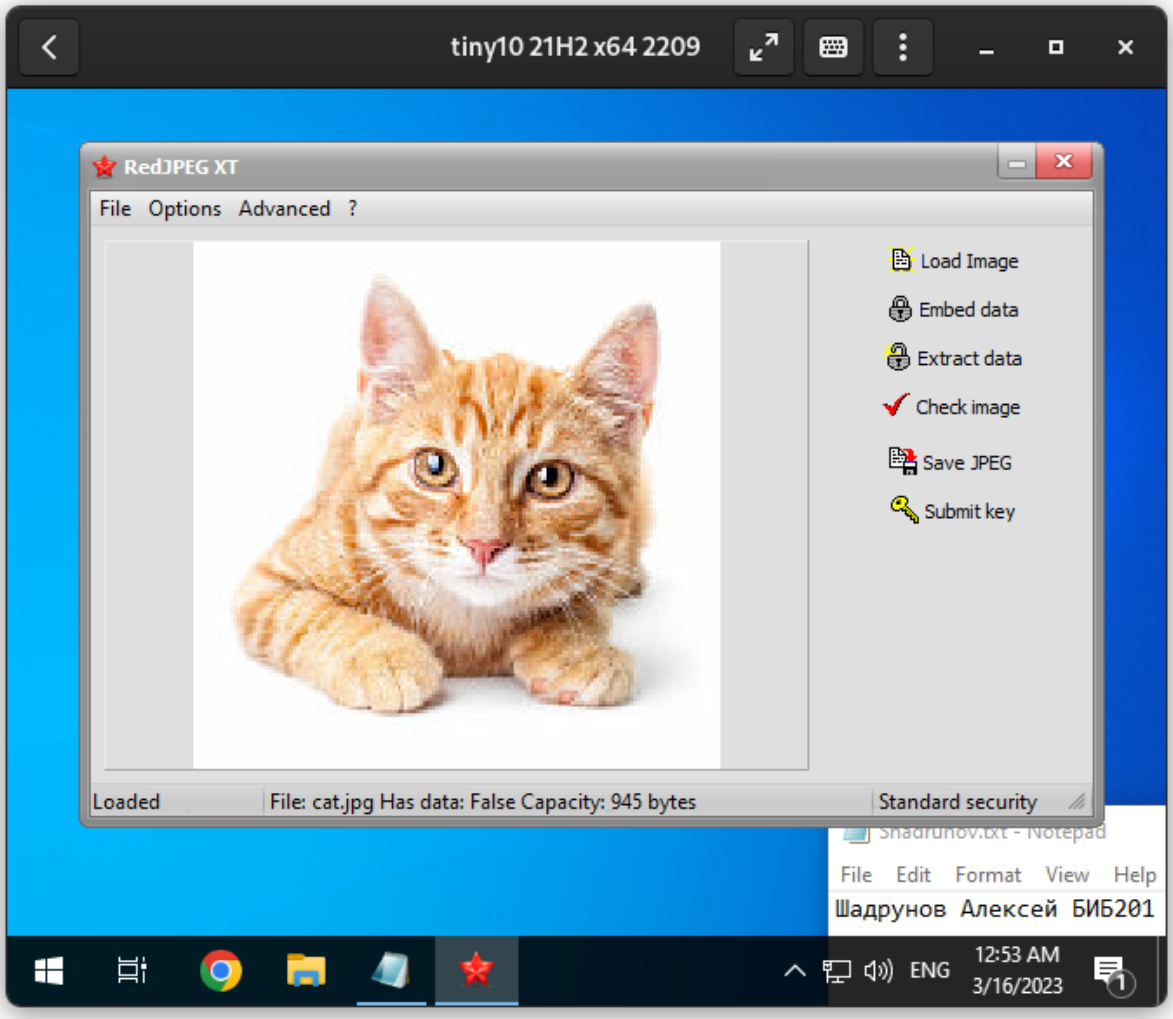

Рисунок 11 – Окно программы RedJPEG

Встроим текстовый файл в картинку (рисунки 12-13).

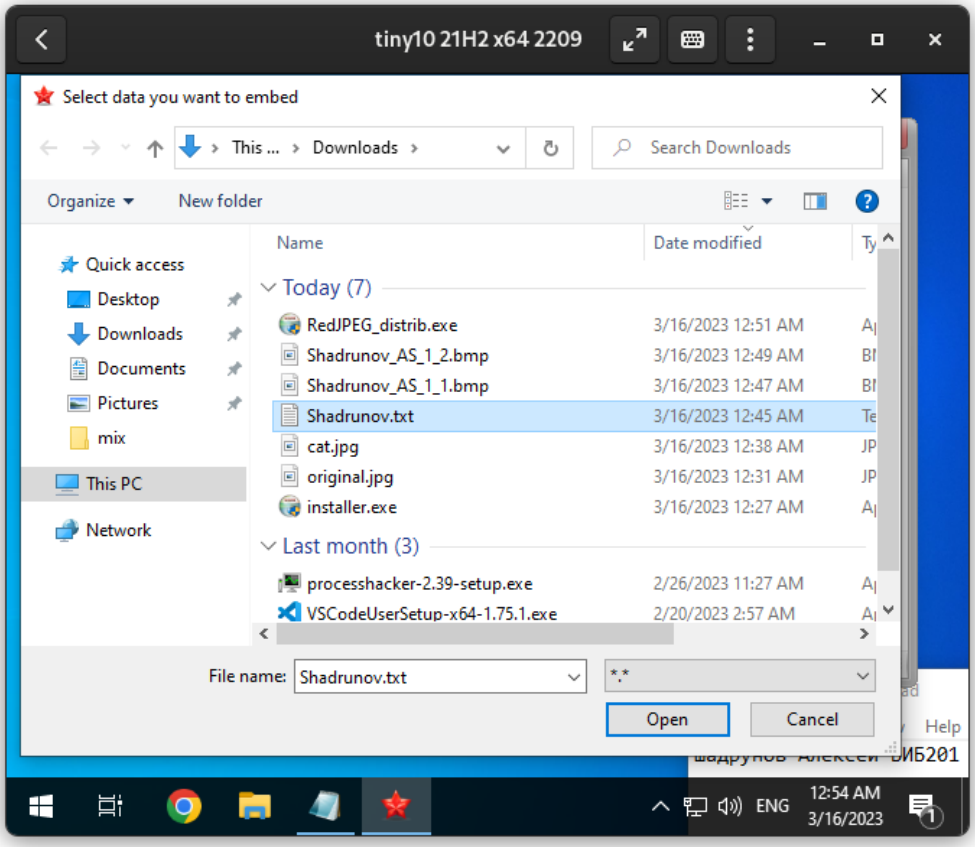

Рисунок 12 – Текстовый файл

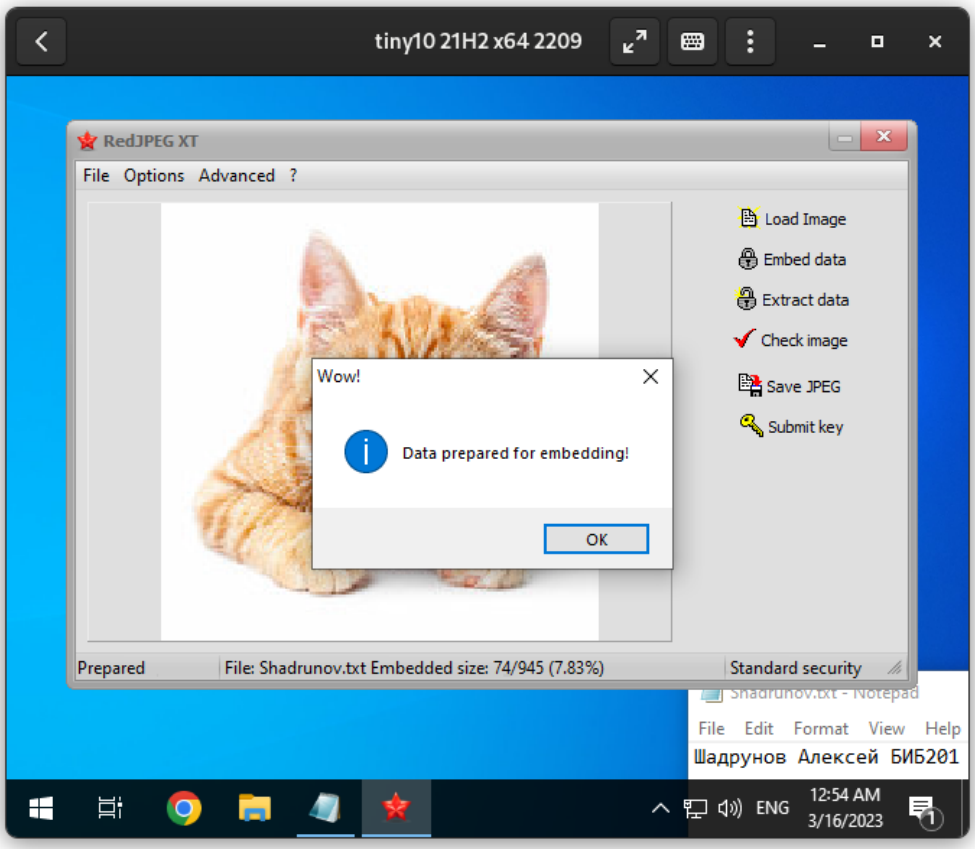

Рисунок 13 – Успешно встроено

Результат встраивания отображён на рисунке 14.

Рисунок 14 – Текстовый файл в картинке

2.3 Программа на python

Сделаем программу для встраивания текста вручную.

Алгоритм встраивания:

- открыть файл. прочитать заголовок, вычислить сдвиг (offset).

- считать строку для встраивания. привести к байтовому представлению.

- снова открыть файл на чтение, открыть файл для записи, прочитать и записать начало файла до сдвига.

- каждый бит из байтового представления длины строки вставить в LSB картинки, записать.

- для каждого байта строки каждый бит записать в LSB очередного байта картинки, записать.

- записать остаток картинки без изменения.

Алгоритм извлечения:

- открыть файл. прочитать заголовок, вычислить сдвиг (offset).

- считать строку для встраивания. привести к байтовому представлению.

- снова открыть файл на чтение, открыть файл для записи, прочитать и записать начало файла до сдвига.

- каждый бит из байтового представления длины строки вставить в LSB картин-

ки, записать.

- для каждого байта строки каждый бит записать в LSB очередного байта картинки, записать.

- записать остаток картинки без изменения.

Код программ приведён в приложении. Результат работы программы и сравнение картинки в hex-редакторе приведены на рисунках 15-16.

```
(.venv) alex@alex-nb ~/D/y/h/3_stageno (main)> python encode.py
Reading file...
File size: 87094
Offset: 54
Enter string to encode: Very secret message!
(.venv) alex@alex-nb ~/D/y/h/3_stageno (main)> python decode.py
Reading file...
File size: 87094
Offset: 54
Decoded string is: Very secret message!
(.venv) alex@alex-nb ~/D/y/h/3_stageno (main)>
```

Рисунок 15 – Работа программы

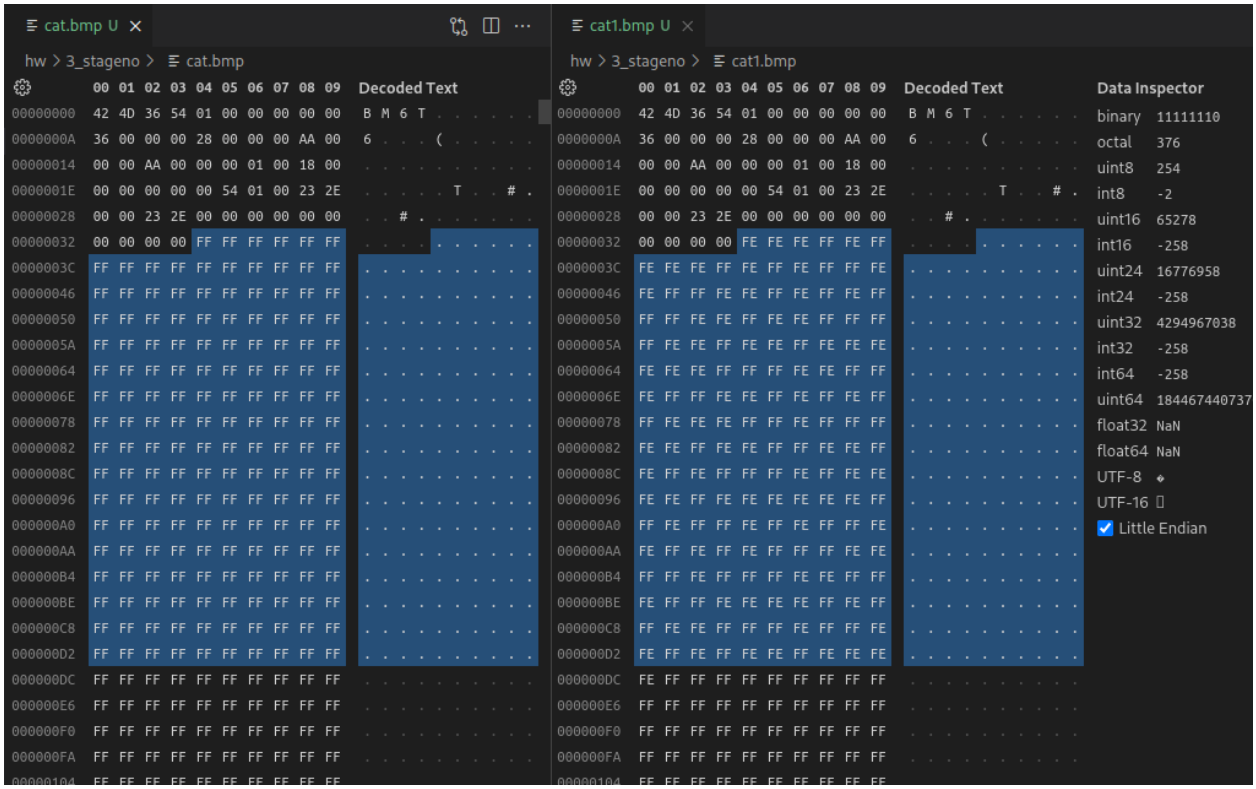

Рисунок 16 – Сравнение в hex-редакторе

3 Выводы о проделанной работе

Я изучил программно-аппаратных средства стеганографии: ImageSpyer и RedJPEG, а также написал собственное программное средство для внедрения сообщения.

Приложение А. Код encode.py

```
""" encode """

import struct

input_file = "cat.bmp"
output_file = "cat1.bmp"

# read offset
with open(input_file, "rb") as f:
    print("Reading file...")
    f.read(2)
    print("File size:", struct.unpack("<i", f.read(4))[0])
    f.read(4)
    offset = struct.unpack("<i", f.read(4))[0]
    print("Offset:", offset)

# prepare string
s = input("Enter string to encode: ")
sb = bytearray(s, "ascii")

# read from input_file and write to output_file
with open(input_file, "rb") as f:
    with open(output_file, "wb") as output_file:
        for i in range(offset):
            output_file.write(f.read(1))  # copy beginning

        sb_len = len(sb) % 255  # byte string length

        for i in range(8):  # record each bit of length
            symbol = (sb_len >> (7 - i)) & 1  # get one bit: 0 or 1
            b = int.from_bytes(f.read(1), "little")  # read one byte
            b = b & 0b11111110  # clear LSB
            b = b | symbol  # write LSB
            output_file.write(b.to_bytes(1, "little"))  # write to file

        for ch in sb:  # record bytes from string
            for i in range(8):  # each byte bit by bit
                symbol = (ch >> (7 - i)) & 1  # get one bit: 0 or 1
                b = int.from_bytes(f.read(1), "little")  # read one byte
                b = b & 0b11111110  # clear LSB
                b = b | symbol  # write LSB
                output_file.write(b.to_bytes(1, "little"))  # write to file

        while b := f.read():  # write file till the end
            output_file.write(b)
```

Приложение Б. Код decode.py

```
""" decode """

import struct

input_file = "cat1.bmp"

with open(input_file, "rb") as f:
    print("Reading file...")
    f.read(2)
    print("File size:", struct.unpack("<i", f.read(4))[0])
    f.read(4)
    offset = struct.unpack("<i", f.read(4))[0]
    print("Offset:", offset)

sb = bytearray("", "ascii")  # prepare string

with open(input_file, "rb") as f:
    f.read(offset)  # skip offset

    # get string length from file
    sb_len = 0
    for i in range(8):
        symbol = f.read(1)
        symbol = int.from_bytes(symbol, "little")
        symbol = symbol & 1  # get LSB
        sb_len = (symbol << (7 - i)) | sb_len  # construct byte from LSB

    # get string from file
    for j in range(sb_len):
        ch = 0
        for i in range(8):
            symbol = f.read(1)
            symbol = int.from_bytes(symbol, "little")
            symbol = symbol & 1  # get LSB
            ch = (symbol << (7 - i)) | ch  # construct byte from LSB
        sb.append(ch)  # add byte to string

print("Decoded string is:", sb.decode("ascii"))
```