

**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»

«Стеганографические 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 | 3 |
| 2 | Ход работы | 3 |
| 2.1 | Image Spyer G2 | 3 |
| 2.2 | RedJPEG | 9 |
| 2.3 | Программа на python | 11 |
| 3 | Выводы о проделанной работе | 13 |
| | Приложение А. Код encode.py | 14 |
| | Приложение Б. Код decode.py | 15 |

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.

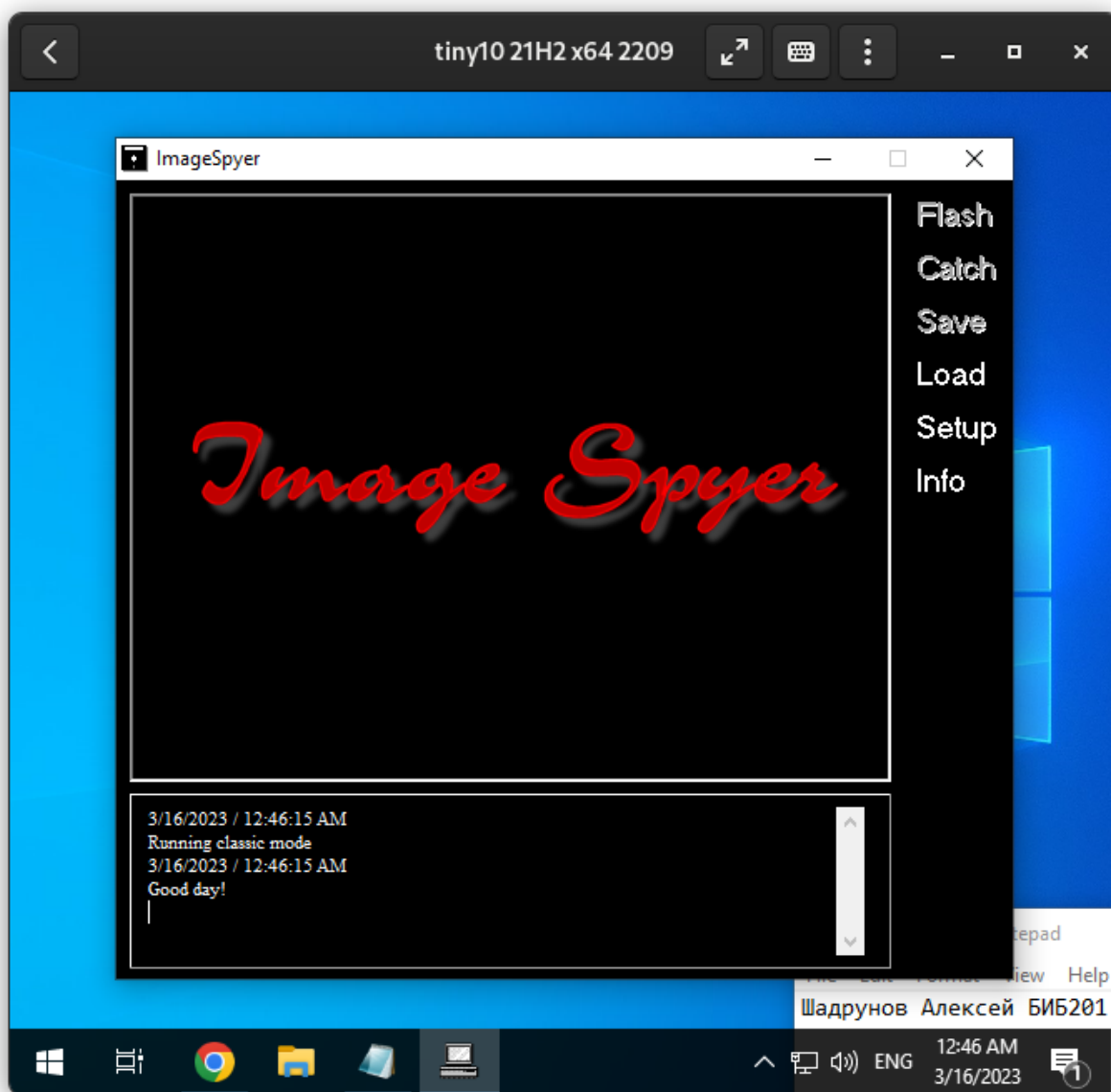


Рисунок 1 – Окно программы

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

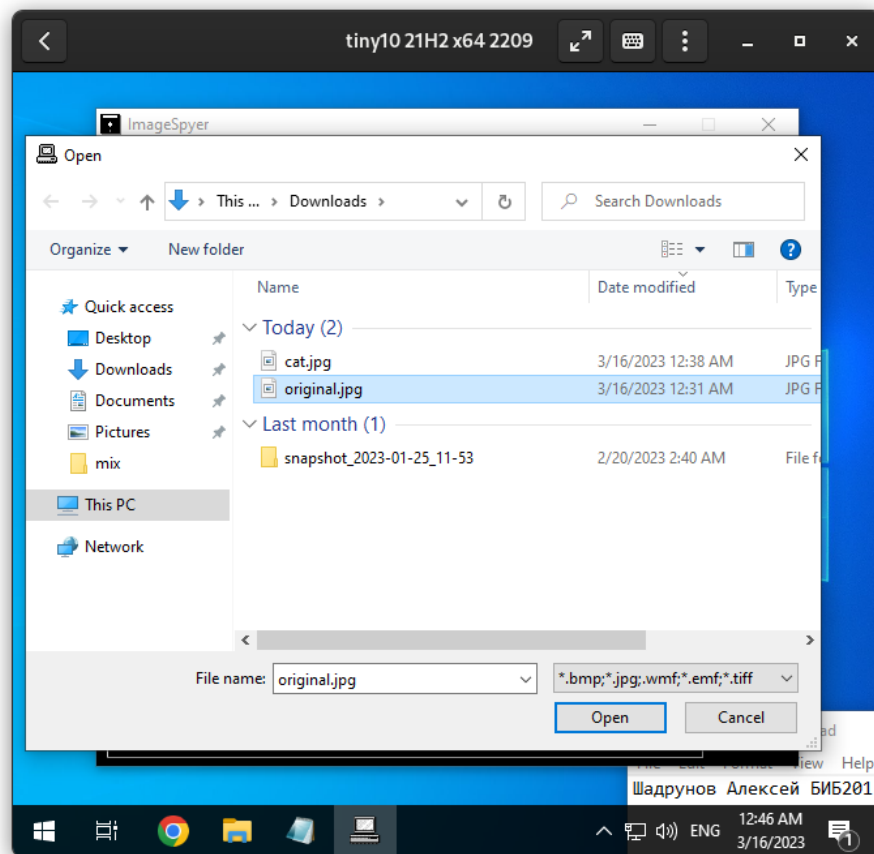


Рисунок 2 – Выбор начальной картинки

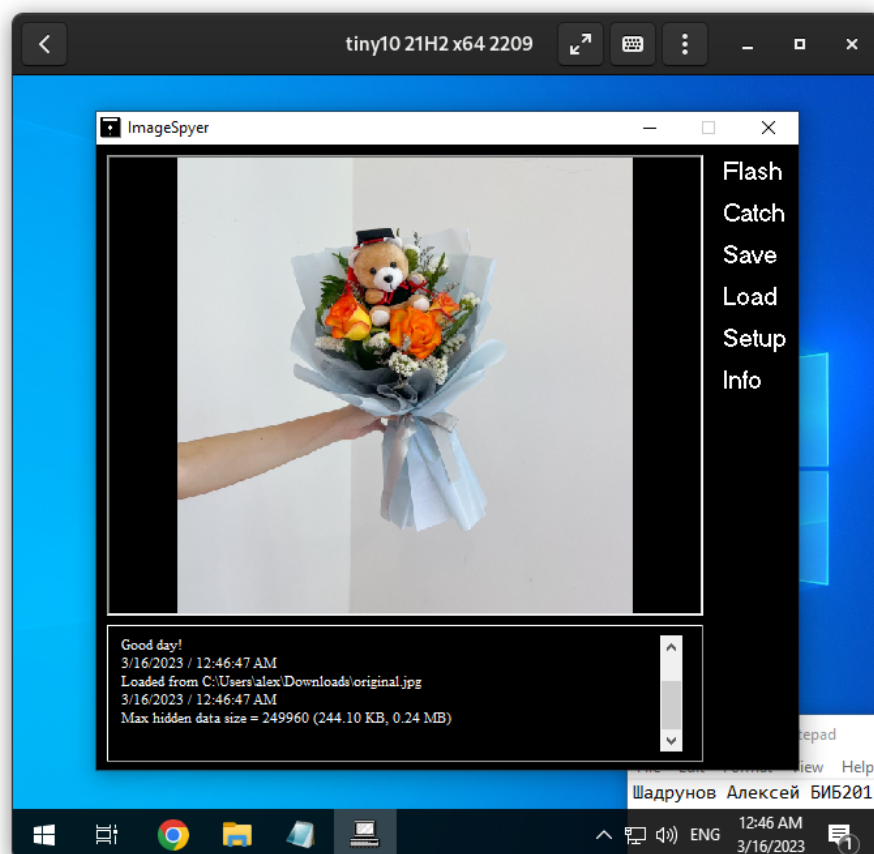


Рисунок 3 – Начальная картинка

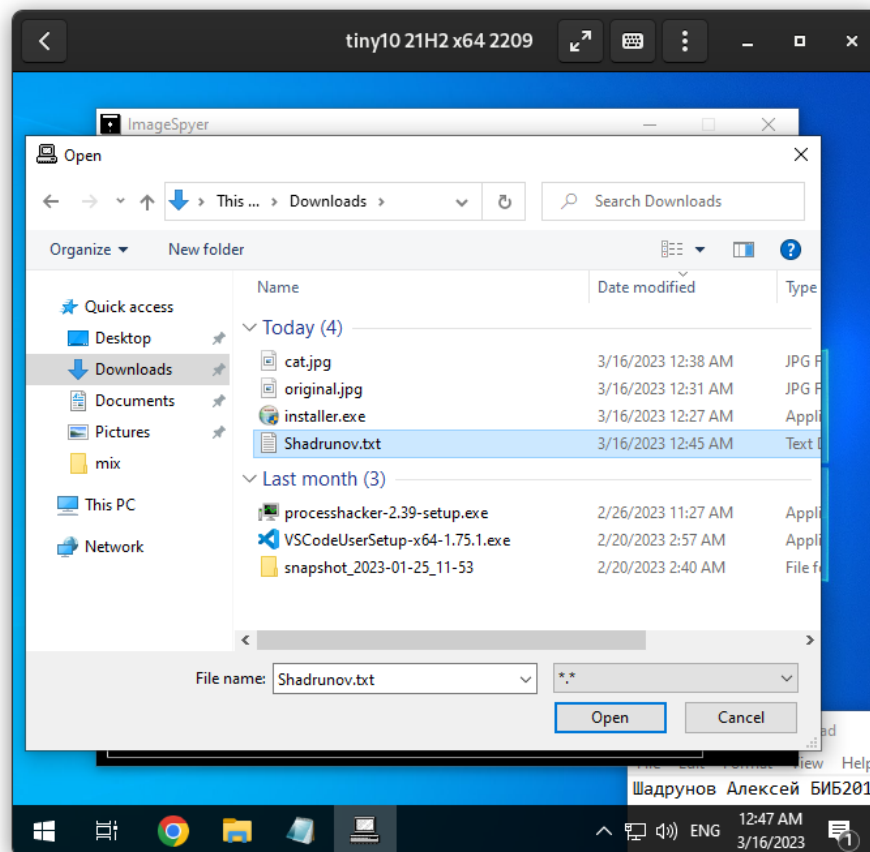


Рисунок 4 – Выбор текстового файла

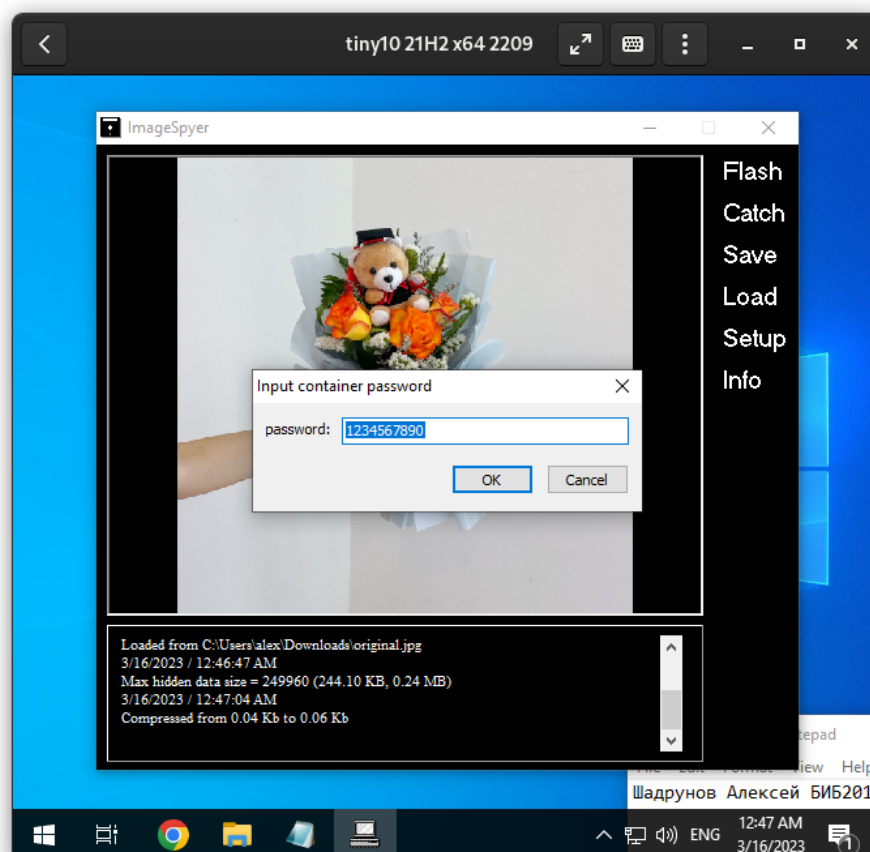


Рисунок 5 – Задание пароля

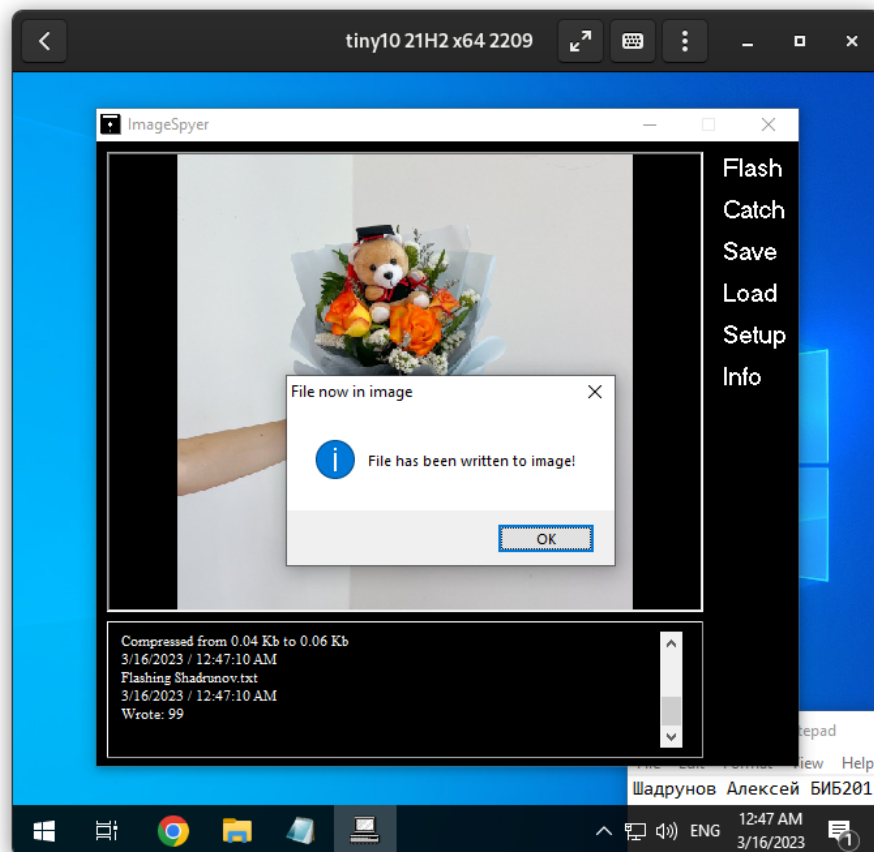


Рисунок 6 – Успешное встраивание

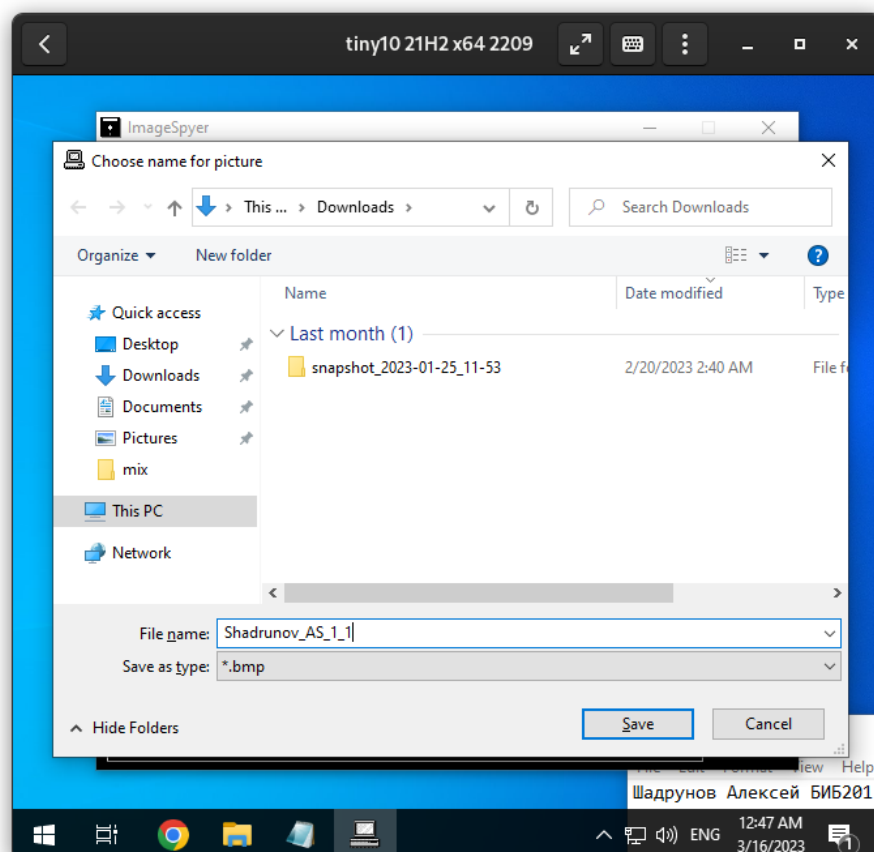


Рисунок 7 – Сохранение файла

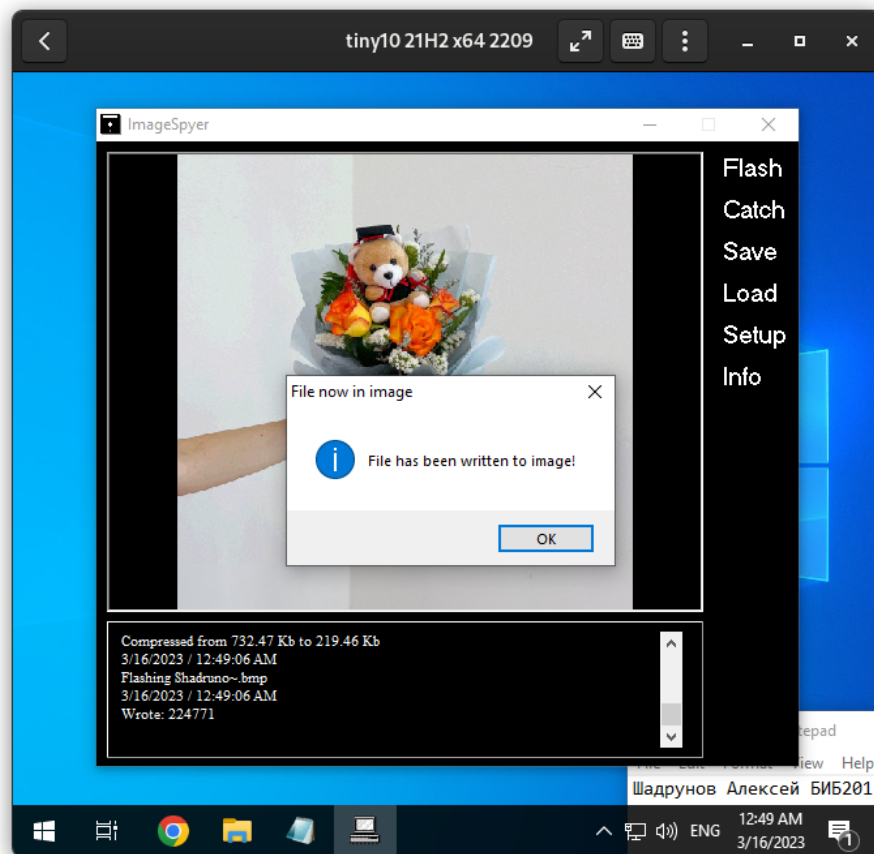


Рисунок 8 – Другой файл встроили в картинку

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.

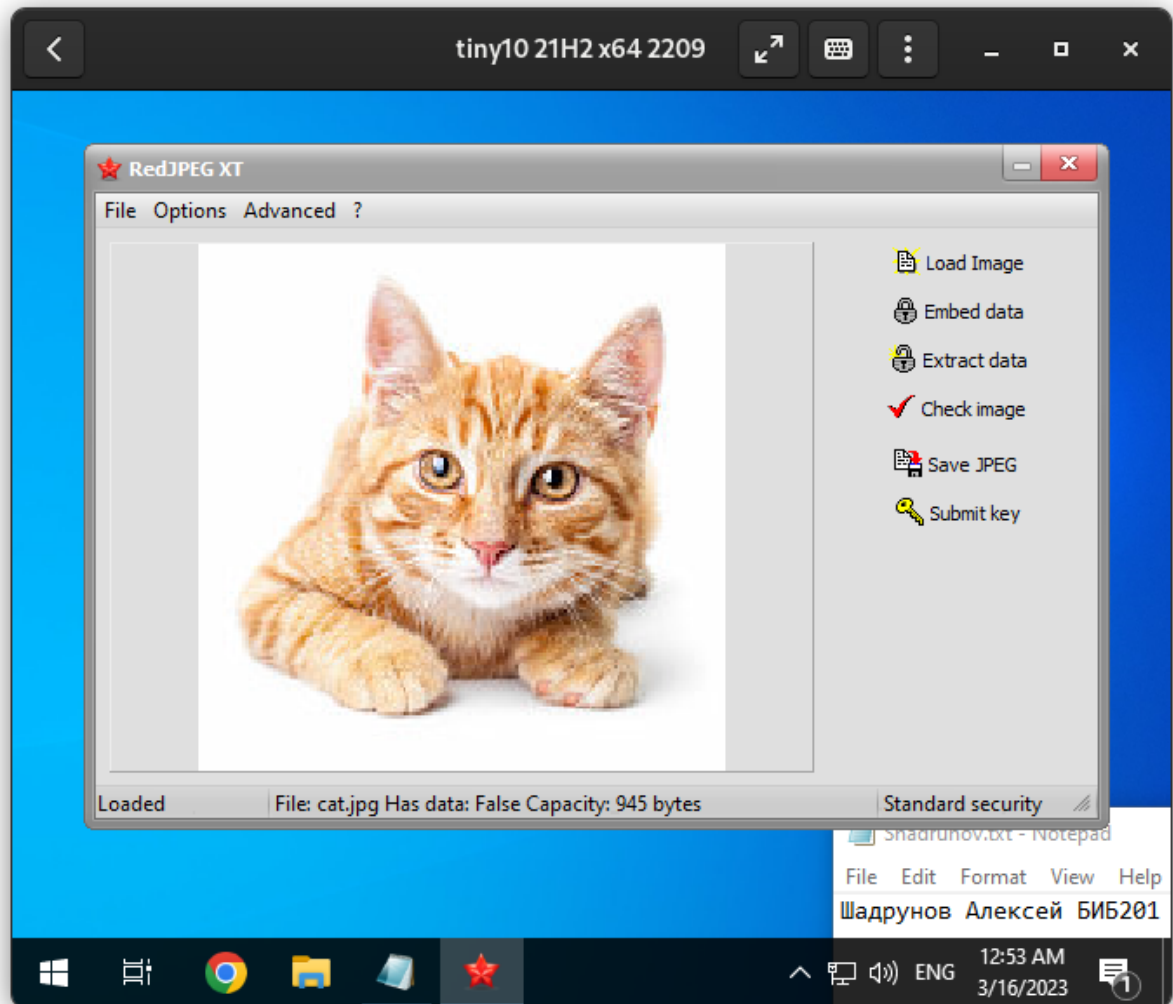


Рисунок 11 – Окно программы RedJPEG

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

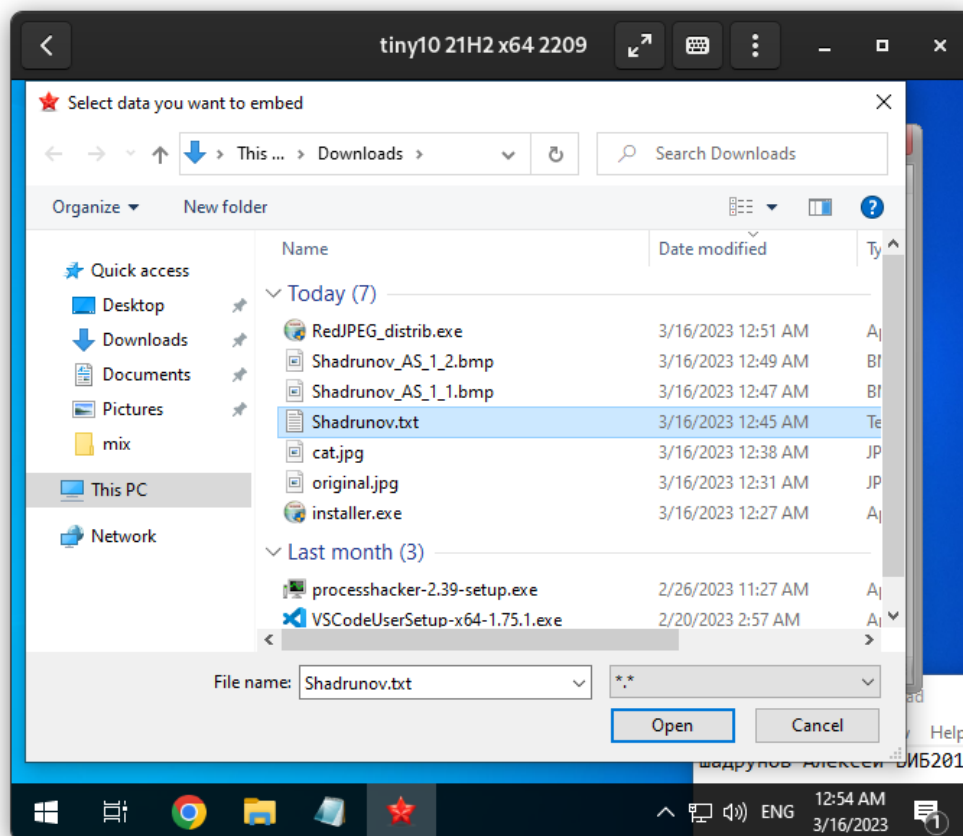


Рисунок 12 – Текстовый файл

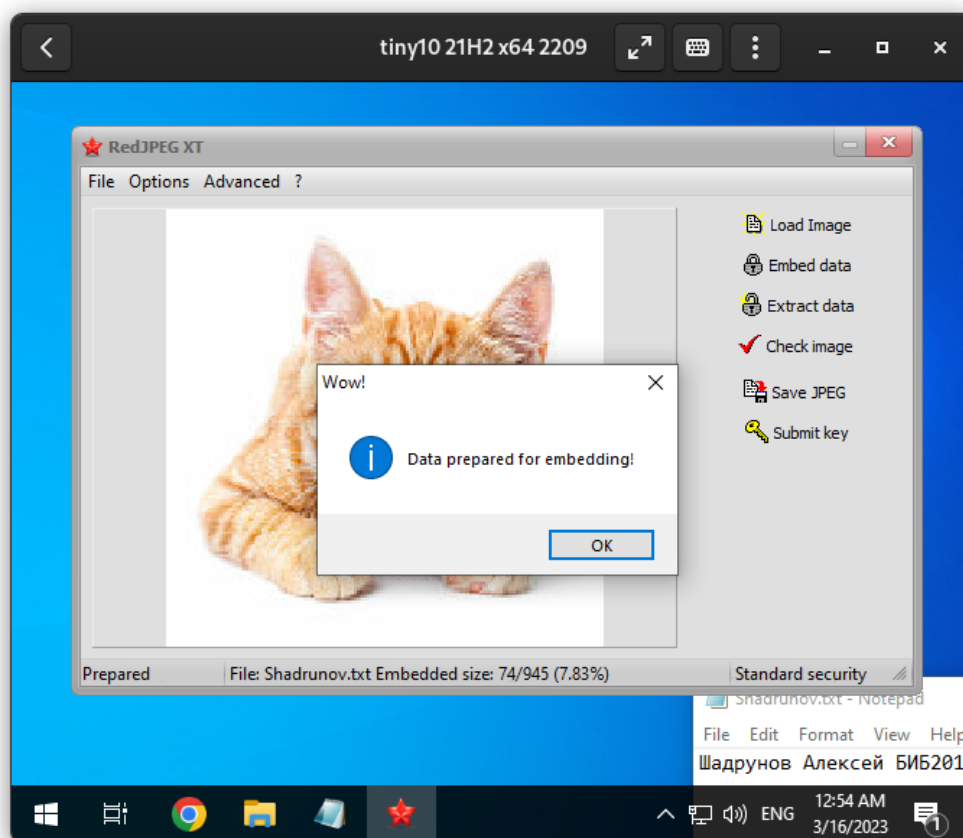


Рисунок 13 – Успешно встроено

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 хекс-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

cat.bmp U x

hw > 3_stageno > cat.bmp

| | 00 | 01 | 02 | 03 | 04 | 05 | 06 | 07 | 08 | 09 | Decoded Text |
|----------|----|----|----|----|----|----|----|----|----|----|-----------------------|
| 00000000 | 42 | 4D | 36 | 54 | 01 | 00 | 00 | 00 | 00 | 00 | B M 6 T |
| 0000000A | 36 | 00 | 00 | 00 | 28 | 00 | 00 | 00 | AA | 00 | 6 (. |
| 00000014 | 00 | 00 | AA | 00 | 00 | 00 | 01 | 00 | 18 | 00 | |
| 0000001E | 00 | 00 | 00 | 00 | 00 | 54 | 01 | 00 | 23 | 2E | T . . # . . |
| 00000028 | 00 | 00 | 23 | 2E | 00 | 00 | 00 | 00 | 00 | 00 | . . # |
| 00000032 | 00 | 00 | 00 | 00 | FF | FF | FF | FF | FF | FF | |
| 0000003C | FF | FF | FF | FF | FF | FF | FF | FF | FF | FF | |
| 00000046 | FF | FF | FF | FF | FF | FF | FF | FF | FF | FF | |
| 00000050 | FF | FF | FF | FF | FF | FF | FF | FF | FF | FF | |
| 0000005A | FF | FF | FF | FF | FF | FF | FF | FF | FF | FF | |
| 00000064 | FF | FF | FF | FF | FF | FF | FF | FF | FF | FF | |
| 0000006E | FF | FF | FF | FF | FF | FF | FF | FF | FF | FF | |
| 00000078 | FF | FF | FF | FF | FF | FF | FF | FF | FF | FF | |
| 00000082 | FF | FF | FF | FF | FF | FF | FF | FF | FF | FF | |
| 0000008C | FF | FF | FF | FF | FF | FF | FF | FF | FF | FF | |
| 00000096 | FF | FF | FF | FF | FF | FF | FF | FF | FF | FF | |
| 000000A0 | FF | FF | FF | FF | FF | FF | FF | FF | FF | FF | |
| 000000AA | FF | FF | FF | FF | FF | FF | FF | FF | FF | FF | |
| 000000B4 | FF | FF | FF | FF | FF | FF | FF | FF | FF | FF | |
| 000000BE | FF | FF | FF | FF | FF | FF | FF | FF | FF | FF | |
| 000000C8 | FF | FF | FF | FF | FF | FF | FF | FF | FF | FF | |
| 000000D2 | FF | FF | FF | FF | FF | FF | FF | FF | FF | FF | |
| 000000DC | FF | FF | FF | FF | FF | FF | FF | FF | FF | FF | |
| 000000E6 | FF | FF | FF | FF | FF | FF | FF | FF | FF | FF | |
| 000000F0 | FF | FF | FF | FF | FF | FF | FF | FF | FF | FF | |
| 000000FA | FF | FF | FF | FF | FF | FF | FF | FF | FF | FF | |
| 00000104 | FF | FF | FF | FF | FF | FF | FF | FF | FF | FF | |

cat1.bmp U x

hw > 3_stageno > cat1.bmp

| | 00 | 01 | 02 | 03 | 04 | 05 | 06 | 07 | 08 | 09 | Decoded Text | Data Inspector |
|----------|----|----|----|----|----|----|----|----|----|----|-----------------------|---|
| 00000000 | 42 | 4D | 36 | 54 | 01 | 00 | 00 | 00 | 00 | 00 | B M 6 T | binary 11111110 |
| 0000000A | 36 | 00 | 00 | 00 | 28 | 00 | 00 | 00 | AA | 00 | 6 (. | octal 376 |
| 00000014 | 00 | 00 | AA | 00 | 00 | 00 | 01 | 00 | 18 | 00 | | uint8 254 |
| 0000001E | 00 | 00 | 00 | 00 | 00 | 54 | 01 | 00 | 23 | 2E | T . . # . . | int8 -2 |
| 00000028 | 00 | 00 | 23 | 2E | 00 | 00 | 00 | 00 | 00 | 00 | . . # | uint16 65278 |
| 00000032 | 00 | 00 | 00 | 00 | FE | FE | FE | FE | FE | FE | | int16 -258 |
| 0000003C | FE | FE | FE | FE | FE | FE | FE | FE | FE | FE | | uint24 16776958 |
| 00000046 | FE | FE | FE | FE | FE | FE | FE | FE | FE | FE | | int24 -258 |
| 00000050 | FF | FF | FE | FE | FE | FE | FE | FE | FE | FE | | uint32 4294967038 |
| 0000005A | FF | FE | FE | FE | FE | FE | FE | FE | FE | FE | | int32 -258 |
| 00000064 | FE | FE | FE | FE | FE | FE | FE | FE | FE | FE | | int64 -258 |
| 0000006E | FE | FF | FE | FE | FE | FE | FE | FE | FE | FE | | uint64 184467440737 |
| 00000078 | FF | FE | FE | FE | FE | FE | FE | FE | FE | FE | | float32 NaN |
| 00000082 | FE | FE | FE | FE | FE | FE | FE | FE | FE | FE | | float64 NaN |
| 0000008C | FE | FE | FE | FE | FE | FE | FE | FE | FE | FE | | UTF-8 |
| 00000096 | FE | FE | FE | FE | FE | FE | FE | FE | FE | FE | | UTF-16 |
| 000000A0 | FF | FE | FE | FE | FE | FE | FE | FE | FE | FE | | <input checked="" type="checkbox"/> Little Endian |
| 000000AA | FF | FE | FE | FE | FE | FE | FE | FE | FE | FE | | |
| 000000B4 | FF | FF | FE | FE | FE | FE | FE | FE | FE | FE | | |
| 000000BE | FE | FF | FE | FE | FE | FE | FE | FE | FE | FE | | |
| 000000C8 | FF | FE | FE | FE | FE | FE | FE | FE | FE | FE | | |
| 000000D2 | FE | FF | FE | FE | FE | FE | FE | FE | FE | FE | | |
| 000000DC | FE | FF | FF | FF | FF | FF | FF | FF | FF | FF | | |
| 000000E6 | FF | FF | FF | FF | FF | FF | FF | FF | FF | FF | | |
| 000000F0 | FF | FF | FF | FF | FF | FF | FF | FF | FF | FF | | |
| 000000FA | FF | FF | FF | FF | FF | FF | FF | FF | FF | FF | | |
| 00000104 | FF | FF | FF | FF | FF | FF | FF | FF | FF | FF | | |

Рисунок 16 – Сравнение в хекс-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

```
1  """ encode """
2
3  import struct
4
5  input_file = "cat.bmp"
6  output_file = "cat1.bmp"
7
8  # read offset
9  with open(input_file, "rb") as f:
10     print("Reading file...")
11     f.read(2)
12     print("File size:", struct.unpack("<i", f.read(4))[0])
13     f.read(4)
14     offset = struct.unpack("<i", f.read(4))[0]
15     print("Offset:", offset)
16
17 # prepare string
18 s = input("Enter string to encode: ")
19 sb = bytearray(s, "ascii")
20
21 # read from input_file and write to output_file
22 with open(input_file, "rb") as f:
23     with open(output_file, "wb") as output_file:
24         for i in range(offset):
25             output_file.write(f.read(1)) # copy beginning
26
27         sb_len = len(sb) % 255 # byte string length
28
29         for i in range(8): # record each bit of length
30             symbol = (sb_len >> (7 - i)) & 1 # get one bit: 0 or 1
31             b = int.from_bytes(f.read(1), "little") # read one byte
32             b = b & 0b11111110 # clear LSB
33             b = b | symbol # write LSB
34             output_file.write(b.to_bytes(1, "little")) # write to file
35
36         for ch in sb: # record bytes from string
37             for i in range(8): # each byte bit by bit
38                 symbol = (ch >> (7 - i)) & 1 # get one bit: 0 or 1
39                 b = int.from_bytes(f.read(1), "little") # read one byte
40                 b = b & 0b11111110 # clear LSB
41                 b = b | symbol # write LSB
42                 output_file.write(b.to_bytes(1, "little")) # write to file
43
44         while b := f.read(): # write file till the end
45             output_file.write(b)
```

Приложение Б. Код decode.py

```
1  """ decode """
2
3  import struct
4
5  input_file = "cat1.bmp"
6
7  with open(input_file, "rb") as f:
8      print("Reading file...")
9      f.read(2)
10     print("File size:", struct.unpack("<i", f.read(4))[0])
11     f.read(4)
12     offset = struct.unpack("<i", f.read(4))[0]
13     print("Offset:", offset)
14
15  sb = bytearray("", "ascii") # prepare string
16
17  with open(input_file, "rb") as f:
18      f.read(offset) # skip offset
19
20     # get string length from file
21     sb_len = 0
22     for i in range(8):
23         symbol = f.read(1)
24         symbol = int.from_bytes(symbol, "little")
25         symbol = symbol & 1 # get LSB
26         sb_len = (symbol << (7 - i)) | sb_len # construct byte from LSB
27
28     # get string from file
29     for j in range(sb_len):
30         ch = 0
31         for i in range(8):
32             symbol = f.read(1)
33             symbol = int.from_bytes(symbol, "little")
34             symbol = symbol & 1 # get LSB
35             ch = (symbol << (7 - i)) | ch # construct byte from LSB
36         sb.append(ch) # add byte to string
37
38  print("Decoded string is:", sb.decode("ascii"))
```