Project title

Steganography

Project Done

By

Cyber strikers

Koutharam Venkata siva Sairam (21G25A3105)

Bodagala sujith kumar (20G21A3109)

Yata jaya mohan (20G21A3159)

Atthuluri gokul sai (20G21A3105)

TABLE OF CONTENTS

CHAPTERS		DECRIPTION	PAGE NO
		ABSRACT	3
1		**INTRODUCTION**	
	1.1	Introduction	5
	1.2	Problem Statement	5
	1.3	project Objective	6
	1.4	Project Scope and Limitations	6
2		**EXISTING SOLUTIONS**	
	2.1	Existing Solutions	8
3		**PROPOSED SOLUTIONS**	
	3.1	Steghide Installation and Running	11
	3.2	Embedding Data in The Image	12
	3.3	Extraction of Data Via Steghide	13
	3.4	Verbose Mode	14
	3.5	Compression Mode	15
4		**SYSTEM REQUIREMETS**	
	4.1	System Requirements and Tools	17
5		**RESULTS AND DISCUSSION**	
	5.1	Results of steghide	20
6		**CONCLUSION AND REFERENCES**	
	6.1	CONCLUSION	23

ABSTRACT

Steganography, an ancient practice dating back to ancient Greece and Rome, has seen a resurgence in modern times as a crucial component of data security strategies. With the increasing prevalence of digital communication and the need for secure transmission of sensitive information, steganography offers a covert means of hiding data within seemingly innocuous files, such as images, audio, or video.

In this project, we focus on implementing steganography using Linux, a popular open-source operating system known for its robustness and flexibility. By harnessing the power of Linux, we aim to provide a seamless and efficient method for embedding and extracting hidden data within digital images.

Central to our implementation is the utilization of the steghide tool, a versatile steganography utility specifically designed for Linux platforms. Steghide employs sophisticated algorithms to embed confidential information within image files, ensuring that the visual appearance of the carrier remains virtually unchanged. This capability is crucial for maintaining the covert nature of the hidden data, as any perceptible alterations to the carrier file could potentially compromise the secrecy of the embedded message.

CHAPTER 1

INTRODUCTION

1. INTRODUCTION

1.1. Introduction

Steghide is a command-line tool for hiding data within various types of files, such as images and audio files. The tool uses a technique called steganography to conceal the data within the file without changing its appearance or functionality. Steganography is a method of hiding information within a medium such as text, images, or audio, in a way that is not easily detectable. Unlike cryptography, which focuses on making a message unreadable to unauthorized parties, steganography is about hiding the existence of the message itself.

Steghide encrypts the data using advanced encryption standards (AES-256) and then embeds it within the least significant bits of the file. This process makes the hidden data difficult to detect with the naked eye and most analysis tools. Steghide is a popular tool for secure data storage and transmission, as it provides a high level of protection against unauthorized access. However, it should be used responsibly and in accordance with applicable laws and regulations

Features:

- The current version of steghide is 0.5.1
- compression of embedded data *)
- encryption of embedded data *)
- embedding of a checksum to verify the integrity of the extracted data *)
- support for JPEG, BMP, WAV and AU files

1.2. Problem Statement

One of the significant challenges in steganography projects, especially those involving digital media like images or audio, is ensuring the robustness of the hidden message against various attacks and transformations applied to the carrier medium. These attacks could include compression, resizing, or format

conversions, which may inadvertently alter or corrupt the hidden data, rendering it unrecoverable or detectable.

1.3. Project Objective

To develop a robust and scalable steganalysis solution capable of detecting hidden data within various types of media files, including images, audio, and video, with a high degree of accuracy and efficiency. The system aims to enhance security measures by identifying and mitigating the risks associated with covert communication channels established through steganographic techniques. The solution should be adaptable to evolving steganographic methods and scalable to handle large volumes of media data in real-time or batch processing scenarios.

1.4 Scope and Limitations of the Project

Project scope :

1. Embedding Data in Cover Files
2. Data Compression
3. Data Encryption

Limitations of the Project:

1. Steghide only supports embedding data in JPEG, BMP, WAV, and AU file formats, restricting compatibility to these types
2. Steghide embeds data only in JPEG, BMP, WAV, and AU formats, restricting compatibility to these specific file types.
3. Steghide's security relies on passphrase-based encryption, effectiveness varying with passphrase strength.

CHAPTER2

EXISTING SOLUTIONS

2. EXISTING SOLUTIONS

2.1 Existing Solutions

1. SteganoJPEG

SteganoJPEG is a Python library designed for embedding and extracting hidden data within JPEG images. This library offers steganographic functionalities similar to Steghide but is implemented in Python, making it platform-independent. It allows users to hide information in the color channels of JPEG images, maintaining the appearance of the image while concealing the embedded data. SteganoJPEG provides a straightforward API for developers to integrate steganography capabilities into their Python applications.

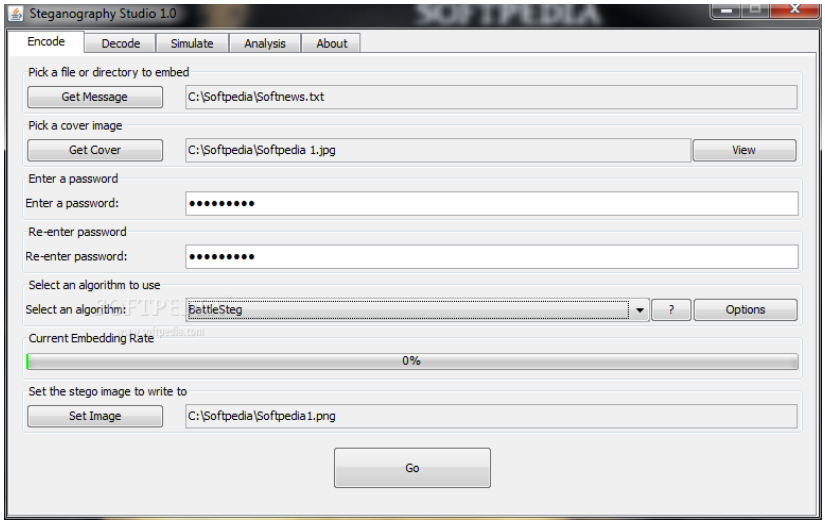
2. OpenStego

OpenStego is an open-source steganography software that supports hiding data in various image and audio file formats. It provides a user-friendly graphical interface, making it accessible for users without extensive technical knowledge. OpenStego offers encryption options to enhance the security of embedded data and supports popular image formats like JPEG and PNG, as well as audio formats such as WAV. This tool is ideal for individuals looking for an easy-to-use steganography solution with basic encryption capabilities.

OPENSTEGO

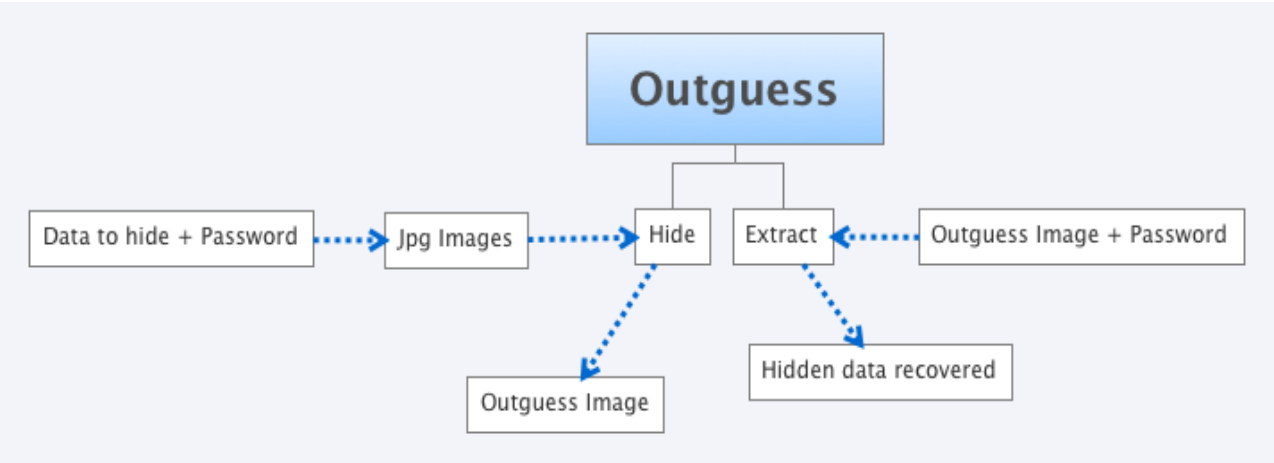
3. Steganography Studio

Steganography Studio is a steganography tool featuring a graphical user interface (GUI) for embedding and extracting hidden messages within image and audio files. It supports multiple file formats and offers encryption features to protect the concealed data. Steganography Studio is designed to be intuitive and user-friendly, suitable for individuals seeking a straightforward steganography solution with visual interaction. It provides a convenient way to hide sensitive information within digital media files.



4. OutGuess

OutGuess is a steganographic tool specifically designed for hiding data within JPEG images. It utilizes a technique called security through obscurity, concealing information within the image's digital data. OutGuess aims to maintain the visual quality of the image while embedding data to avoid detection. This tool is suitable for scenarios where users need to hide information within JPEG images for privacy or covert communication purposes.



CHAPTER 3

PROPOSED SOLUTION

3. PROPOSED SOLUTIONS

3.1 Steghide Installation and Running

To contribute to Steghide or use it on your system, follow these steps based on your operating system:

Linux / Unix:

Ensure dependencies (libmhash, libmcrypt, libjpeg, zlib) are installed.

Download and unpack the source distribution.

Run ./configure, make, make check, and make install as root.

Windows:

Download the precompiled binary from the Steghide website.

If compiling from source, set up a C++ compiler and follow the specific instructions for your compiler.

Quick Start:

Here are some basic examples of how to use Steghide:

Update system packages

update all available repositories in the system

```
┌──(kali㉿kali)-[~]
└─$ sudo apt update
[sudo] password for kali:
Hit:1 http://kali.download/kali kali-rolling InRelease
Reading package lists ... Done
Building dependency tree ... Done
Reading state information ... Done
1508 packages can be upgraded. Run 'apt list --upgradable' to see them.
```

Install the steghide tool

```
┌──(kali㉿kali)-[~]
└─$ sudo apt-get install steghide
Reading package lists ... Done
Building dependency tree ... Done
Reading state information ... Done
The following additional packages will be installed:
  libmcrypt4 libmhash2
Suggested packages:
  libmcrypt-dev mcrypt
The following NEW packages will be installed:
  libmcrypt4 libmhash2 steghide
0 upgraded, 3 newly installed, 0 to remove and 1508 not upgraded.
Need to get 311 kB of archives.
After this operation, 907 kB of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:1 http://kali.download/kali kali-rolling/main amd64 libmcrypt4 amd64 2.5.
8-7 [72.6 kB]
```

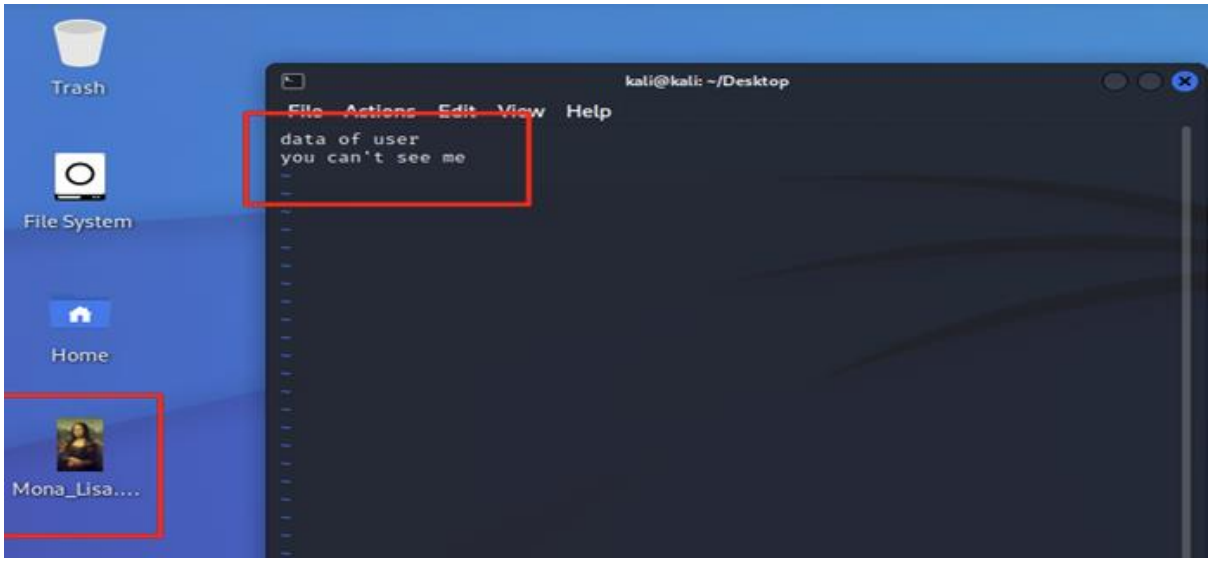
Getting help in Steghide

```
kali@kali: ~
File  Actions  Edit  View  Help
┌──(kali㉿kali)-[~]
└─$ steghide --help
steghide version 0.5.1

the first argument must be one of the following:
 embed, --embed          embed data
 extract, --extract      extract data
 info, --info            display information about a cover- or stego-file
   info <filename>       display information about <filename>
 encinfo, --encinfo      display a list of supported encryption algorithms
 version, --version      display version information
 license, --license      display steghide's license
 help, --help            display this usage information

embedding options:
 -ef, --embedfile        select file to be embedded
   -ef <filename>        embed the file <filename>
 -cf, --coverfile        select cover-file
   -cf <filename>        embed into the file <filename>
 -p, --passphrase        specify passphrase
   -p <passphrase>       use <passphrase> to embed data
 -sf, --stegofile        select stego file
   -sf <filename>        write result to <filename> instead of cover-file
 -e, --encryption        select encryption parameters
   -e <a>[<m>]|<m>[<a>]  specify an encryption algorithm and/or mode
   -e none               do not encrypt data before embedding
 -z, --compress          compress data before embedding (default)
   -z <l>                 using level <l> (1 best speed ... 9 best compression)
 -Z, --dontcompress      do not compress data before embedding
 -K, --nochecksum        do not embed crc32 checksum of embedded data
 -N, --dontembedname     do not embed the name of the original file
 -f, --force             overwrite existing files
 -q, --quiet             suppress information messages
 -v, --verbose           display detailed information

extracting options:
 -sf, --stegofile        select stego file
   -sf <filename>        extract data from <filename>
 -p, --passphrase        specify passphrase
   -p <passphrase>       use <passphrase> to extract data
 -xf, --extractfile      select file name for extracted data
```

3.2 Embedding Data in The Image

We hide data in the image using Steghide so that only the person who acknowledges it can read that. So, we made a text file named as user.txt in which we wrote our confidential data and image.jpeg is that file in which we are embedding our data. To achieve this, we'll be executing the following command:

steghide embed -ef<txt filename> -ef<media filename>



```
┌──(kali㉿kali)-[~/Desktop]
└─$ steghide embed -ef user.txt -cf image.jpg
Enter passphrase:
Re-Enter passphrase:
embedding "user.txt" in "image.jpg"... done

┌──(kali㉿kali)-[~/Desktop]
└─$
```

Here we use passphrase as a password to embed file in image, and this password we will use when extracting data from image. And we can set password in command itself, like this:

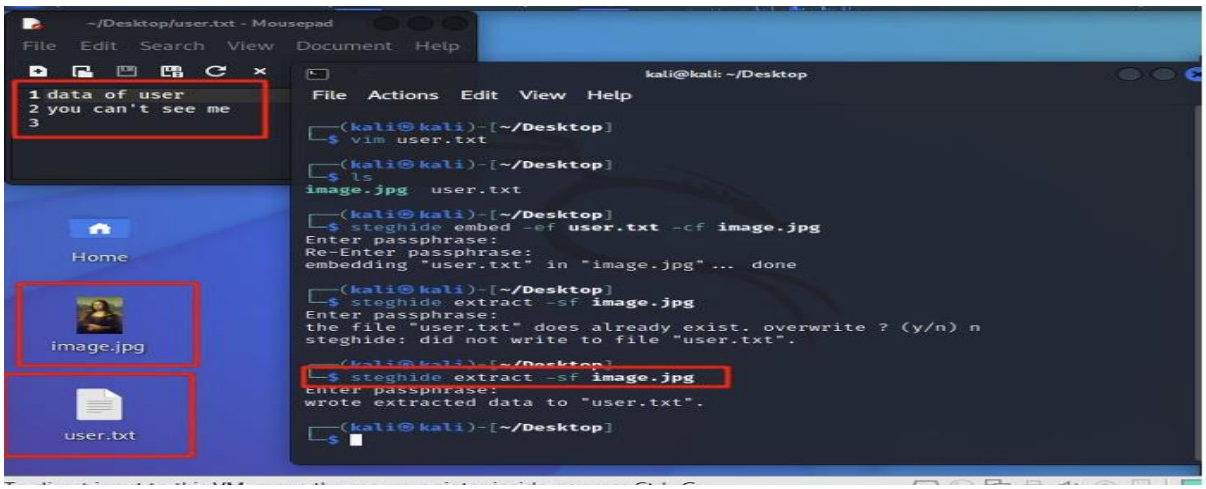
steghide embed -ef<txt filename> -ef<media filename>-p<password>

ef and cf are termed as embedded file and cover file respectively. And the image still like:

3.3 Extraction of Data Via Steghide

Using Steghide adds an extra layer of security by allowing us to use a password for it. Now, to extract the hidden data use the following command:

steghide extract -sf <media filename>sf is a secret file



view file information before extracting

```
┌──(kali㉿kali)-[~/Desktop]
└─$ steghide info image.jpg
"image.jpg":
  format: jpeg
  capacity: 73.1 KB                    with data
Try to get information about embedded data ? (y/n) y
Enter passphrase:
  embedded file "user.txt":
    size: 31.0 Byte
    encrypted: rijndael-128, cbc
    compressed: yes

┌──(kali㉿kali)-[~/Desktop]
└─$ steghide info Mona_Lisa.jpg
"Mona_Lisa.jpg":
  format: jpeg
  capacity: 73.1 KB                    without data
Try to get information about embedded data ? (y/n) y
Enter passphrase:
steghide: could not extract any data with that passphrase!

┌──(kali㉿kali)-[~/Desktop]
└─$
```

Remove Steghide Tool

To remove the steghide we can use this command:

sudo apt-get remove steghide

3.4 Verbose Mode

Steghide's verbose mode is an option that allows you to see more detailed information about the steganography process as it occurs. When verbose mode is enabled, Steghide will display additional information in the output, such as the percentage of the embedding process completed, the number of bytes embedded, and any errors that occur during the process.

To enable verbose mode in Steghide, you can use the -v or --verbose option when running the embed or extract command.

In embed: steghide embed -cf [cover file] -ef [data file] -p [passphrase] -v

In extract: steghide extract -sf [steganographic file] -p [passphrase] -v

```
┌──(kali㉿kali)-[~/Desktop]
└─$ steghide embed -ef user.txt -cf Mona_Lisa.jpg -p toor -v
reading secret file "user.txt" ... done
reading cover file "Mona_Lisa.jpg" ... done
creating the graph ... 72 sample values, 373 vertices, 68070 edges
executing Static Minimum Degree Construction Heuristic ... 99.7% (1.0) done

┌──(kali㉿kali)-[~/Desktop]
└─$ steghide extract -sf Mona_Lisa.jpg -p toor
the file "user.txt" does already exist. overwrite ? (y/n) y
wrote extracted data to "user.txt".

┌──(kali㉿kali)-[~/Desktop]
└─$ steghide extract -sf Mona_Lisa.jpg -p toor -v
reading stego file "Mona_Lisa.jpg" ... done
extracting data ... done
checking crc32 checksum ... ok
writing extracted data to "user.txt" ... the file "user.txt" does already exist
. overwrite ? (y/n) y
 done

┌──(kali㉿kali)-[~/Desktop]
└─$
```

3.5 Compression Mode

Steghide's compression mode is an option that allows you to compress the data before embedding it in an image file. This can be useful for reducing the size of the data, making it easier to embed in the image file.

To enable compression mode in Steghide, you can use the -Z or --compress option when running the embed command. For example, to enable compression mode when embedding data in an image file, you can use the following command:

steghide embed -cf [cover file] -ef [data file] -p [passphrase] -Z

```
┌──(kali㉿kali)-[~/Desktop]
└─$ steghide embed -ef user.txt -cf image.jpg -p root -Z
embedding "user.txt" in "image.jpg"... done
```

CHAPTER 4

SYSTEMREQUIREMETS AND TOOLS

4.SYSTEM REQURIEMENTS AND TOOLS

4.1 System Requirements & Tools

1. Hardware Requirements:

Steghide can run on standard hardware configurations typically found in modern desktop or laptop computers, including any recent processor (e.g., Intel Core series, AMD Ryzen) and a minimum of 2 GB of RAM for smooth operation. Adequate disk space is also necessary to accommodate the operating system, Steghide installation, and the files involved in steganography.

2. Operating System Compatibility:

Steghide is compatible with a variety of operating systems, making it versatile for different user preferences:

Linux: Supports major distributions like Ubuntu, Debian, CentOS, Fedora, etc.

Unix: Compatible with Unix-based systems such as BSD variants.

3. **Windows:**

Works on supported versions like Windows 7, 8, and 10, with the use of a Cygwin environment for compiling from source.

Mac OS X: Can be compiled and used on macOS systems.

4. Internet Connection:

An internet connection is required primarily for downloading Steghide and its dependencies, if they are not already available on the system.

5. Software Dependencies:

To successfully compile and use Steghide, several libraries and tools are necessary:

libmhash: Provides hash algorithms and cryptographic key generation capabilities.

Source: http://mhash.sourceforge.net/

libmcrypt: Facilitates symmetric encryption algorithms.

Source: http://mcrypt.sourceforge.net/

libjpeg: Supports manipulation of JPEG image files.

Source: http://www.ijg.org/

zlib: Enables lossless data compression.

Source: http://www.gzip.org/zlib/

CHAPTER 5

RESULTS ANDDISSCUSION

5. RESULTS ANDDISSCUSION

5.1 Results of steghide :

```
┌──(kali㉿kali)-[~]
└─$ sudo apt update
[sudo] password for kali:
Hit:1 http://kali.download/kali kali-rolling InRelease
Reading package lists ... Done
Building dependency tree ... Done
Reading state information ... Done
1508 packages can be upgraded. Run 'apt list --upgradable' to see them.
```

5.2 Update system packages

```
┌──(kali㉿kali)-[~]
└─$ sudo apt-get install steghide
Reading package lists ... Done
Building dependency tree ... Done
Reading state information ... Done
The following additional packages will be installed:
  libmcrypt4 libmhash2
Suggested packages:
  libmcrypt-dev mcrypt
The following NEW packages will be installed:
  libmcrypt4 libmhash2 steghide
0 upgraded, 3 newly installed, 0 to remove and 1508 not upgraded.
Need to get 311 kB of archives.
After this operation, 907 kB of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:1 http://kali.download/kali kali-rolling/main amd64 libmcrypt4 amd64 2.5.
8-7 [72.6 kB]
```

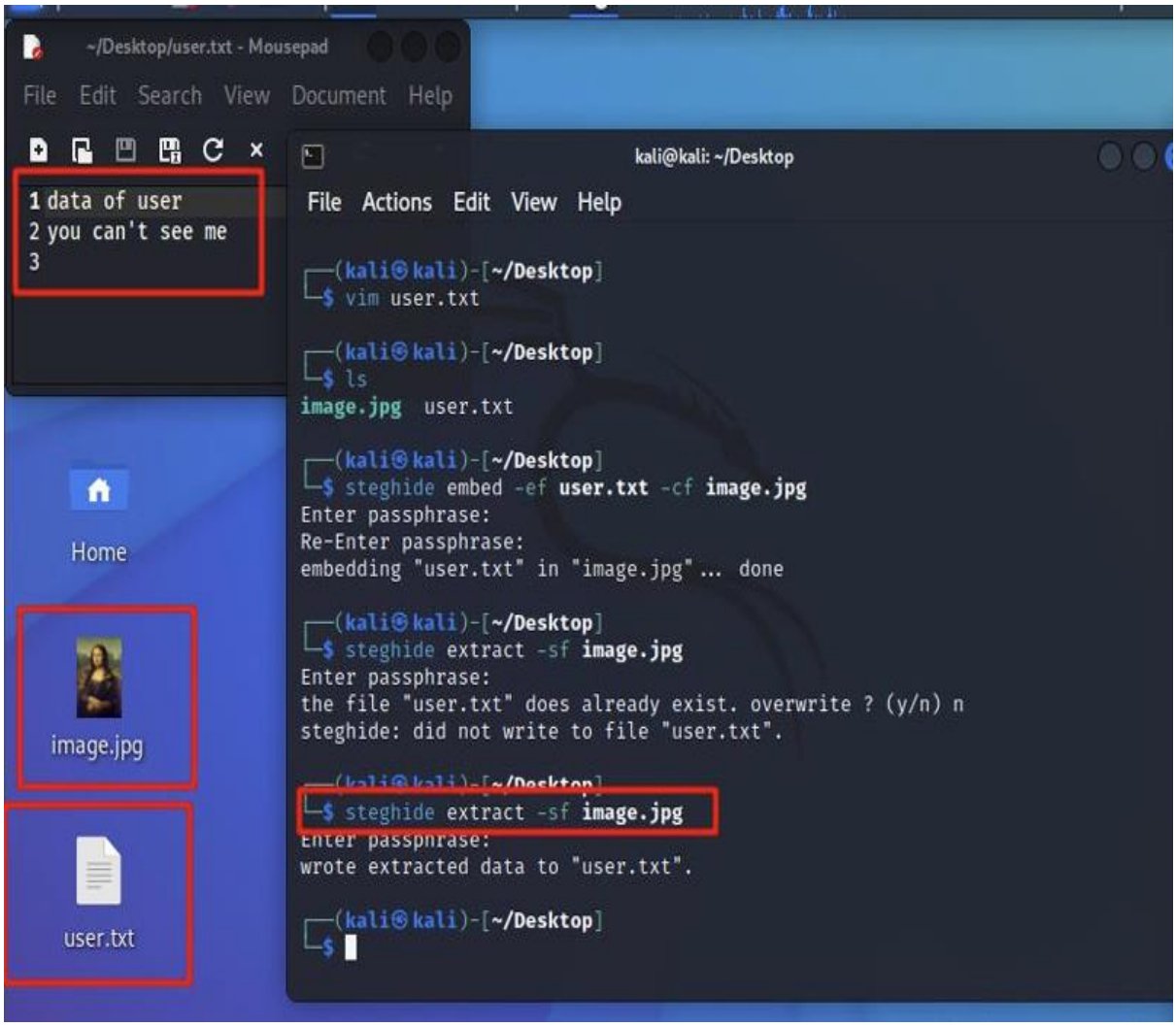
5.3 Install the steghide tool

```
kali@kali: ~
File  Actions  Edit  View  Help

┌──(kali㉿kali)-[~]
└─$ steghide --help

steghide version 0.5.1

the first argument must be one of the following:
 embed, --embed          embed data
 extract, --extract      extract data
 info, --info            display information about a cover- or stego-file
   info <filename>       display information about <filename>
 encinfo, --encinfo      display a list of supported encryption algorithms
 version, --version      display version information
 license, --license      display steghide's license
 help, --help            display this usage information

embedding options:
 -ef, --embedfile        select file to be embedded
   -ef <filename>        embed the file <filename>
 -cf, --coverfile        select cover-file
   -cf <filename>        embed into the file <filename>
 -p, --passphrase        specify passphrase
   -p <passphrase>       use <passphrase> to embed data
 -sf, --stegofile        select stego file
   -sf <filename>        write result to <filename> instead of cover-file
 -e, --encryption        select encryption parameters
   -e <a>[<m>]|<m>[<a>]  specify an encryption algorithm and/or mode
   -e none               do not encrypt data before embedding
 -z, --compress          compress data before embedding (default)
   -z <l>                 using level <l> (1 best speed ... 9 best compression)
 -Z, --dontcompress      do not compress data before embedding
 -K, --nochecksum        do not embed crc32 checksum of embedded data
 -N, --dontembedname     do not embed the name of the original file
 -f, --force             overwrite existing files
 -q, --quiet             suppress information messages
 -v, --verbose           display detailed information

extracting options:
 -sf, --stegofile        select stego file
   -sf <filename>        extract data from <filename>
 -p, --passphrase        specify passphrase
   -p <passphrase>       use <passphrase> to extract data
 -xf, --extractfile      select file name for extracted data
```

5.4 Getting help in Steghide

```
┌──(kali㉿kali)-[~/Desktop]
└─$ steghide embed -ef user.txt -cf image.jpg
Enter passphrase:
Re-Enter passphrase:
embedding "user.txt" in "image.jpg"... done

┌──(kali㉿kali)-[~/Desktop]
└─$
```

5.5 Embedding Data in The Image



5.6 Extraction of Data Via Steghide

```
┌──(kali㉿kali)-[~/Desktop]
└─$ steghide embed -ef user.txt -cf Mona_Lisa.jpg -p toor -v
reading secret file "user.txt" ... done
reading cover file "Mona_Lisa.jpg" ... done
creating the graph ... 72 sample values, 373 vertices, 68070 edges
executing Static Minimum Degree Construction Heuristic ... 99.7% (1.0) done

┌──(kali㉿kali)-[~/Desktop]
└─$ steghide extract -sf Mona_Lisa.jpg -p toor
the file "user.txt" does already exist. overwrite ? (y/n) y
wrote extracted data to "user.txt".

┌──(kali㉿kali)-[~/Desktop]
└─$ steghide extract -sf Mona_Lisa.jpg -p toor -v
reading stego file "Mona_Lisa.jpg" ... done
extracting data ... done
checking crc32 checksum ... ok
writing extracted data to "user.txt" ... the file "user.txt" does already exist
. overwrite ? (y/n) y
 done

┌──(kali㉿kali)-[~/Desktop]
└─$
```

5.7 Verbose Mode

```
┌──(kali㉿kali)-[~/Desktop]
└─$ steghide embed -ef user.txt -cf image.jpg -p root -Z
embedding "user.txt" in "image.jpg" ... done
```

5.7 Compression Mode

6. Conclusion

In conclusion, image steganography is a powerful technique for hiding data within digital images, and it has a wide range of applications in various fields. However, it is important to use this technique responsibly and in accordance with applicable laws and regulations.

Steghide is a popular tool for implementing image steganography, as it provides strong encryption and compression features, along with configurable embedding methods. The tool is open source and available on multiple platforms, making it accessible to a wide range of users.

Steghide can be used for various purposes, including secure data storage and transmission, digital watermarking, and covert communication. However, it can also be used for illicit purposes, such as hiding malware or other malicious code within seemingly harmless images.

Overall, the use of image steganography and Steghide requires careful consideration and responsible use to ensure that the technology is not misused or abused.