**The Use of Steganography to Encode a Secret Message Inside a Picture File**

Matthew Owings, Julian Thomas, Scott Roush

College of Sciences, Auburn University at Montgomery

CSCI 4080: Intro to Digital Forensics

Dr. Liu

November 28, 2021

**Abstract**

Steganography is the method of hiding information inside of a picture. Often, it is a text file (.txt) being embedded inside of a picture file (.jpg) allowing someone to distribute information that may be unlawful or illegal. The picture file would look almost identical to the original, with minor alternations, all while text data can be extracted from the picture. This can go unnoticed, as an inexperienced investigator may not see the potential of hidden information inside of a seemingly innocent picture file. It is important for Digital Forensics Investigators to know about this method of hiding information, so that all information can be known when it is time to present evidence in a court of law.

**The Secret Message**

The idea of steganography is to embed and hide a file inside of another file. In this case, a text file (.txt) being inserted into a picture file (in this case, .jpg).

The text file being used is as follows:

Text

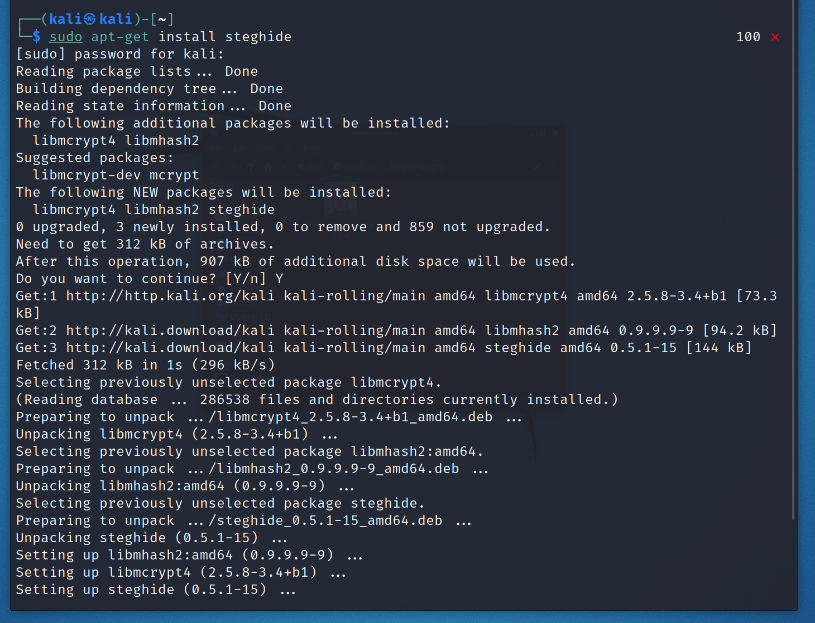
Description automatically generated

Despite being clearly ludicrous, it serves the purpose of knowing what the expected output is. The following text files were inserted into the following three images:

|  |  |  |
| --- | --- | --- |
| A picture containing person, wall, person, indoor  Description automatically generated | A cat with glowing eyes  Description automatically generated with low confidence | A picture containing tree, plant  Description automatically generated |
| Picture of Matthew Owings | Scott Roush’s dog Frankie | Julian Thomas’ Tree Photograph |

**Installing Steghide**

The first step of the process is to use Kali Linux. To achieve the desired results, we utilized the program Steghide, which is a steganography tool for insertion of text within an image. To use this program, you can open the terminal on Kali Linux and input the command **install steghide**.



This command installs the necessary dependencies and packages needed to use Steghide.

**Steghide Commands**

After this point, a variety of commands are available to you which can be viewed using the **man steghide** command. The main commands that will be used are:

To embed text into a picture**:**

**steghide embed -ef [text file] -cf [picture file] -sf [new-file-name] -p [password]**

To extract the text from a picture:

**steghide extract -sf [picture with hidden message file] -p [password] -xf [new-file-name]**

**-ef** – Embed File

**-cf** – Cover File

**-sf** – Stego File

**-p** – Password

**-xf** – Extract File

Extra Commands:

**man steghide** – View all commands

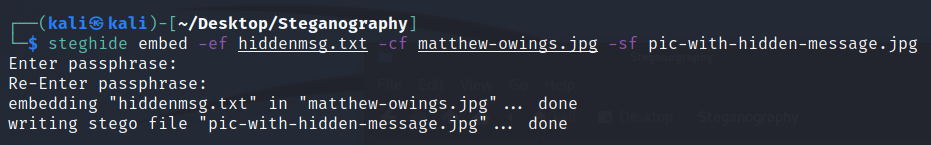
**steghide --version** – View Steghide version

**Steps to Embed Text into a Picture with Steghide**

The first step to embedding and hiding a text file inside of a picture file is to create and write a text file, and to select an image to be used to hide the contents of the text file inside of. For example,

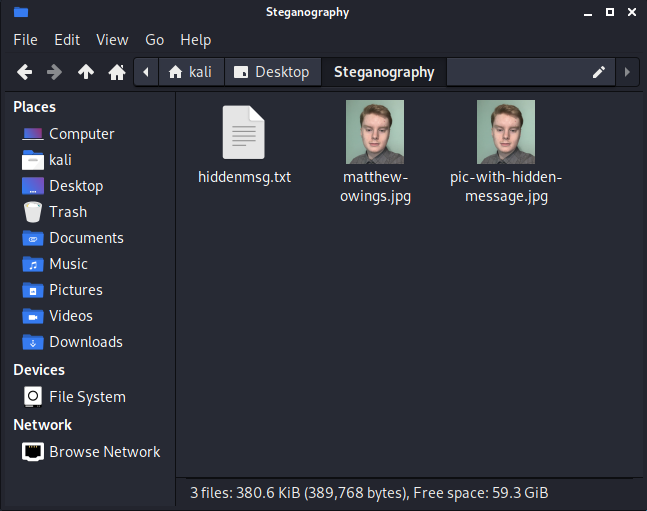
|  |  |
| --- | --- |
| hiddenmsg.txt | matthew-owings.jpg |
|  |  |

Text File **🡪 Embedded Into 🡪** Picture File



**Using the Hidden Message File**

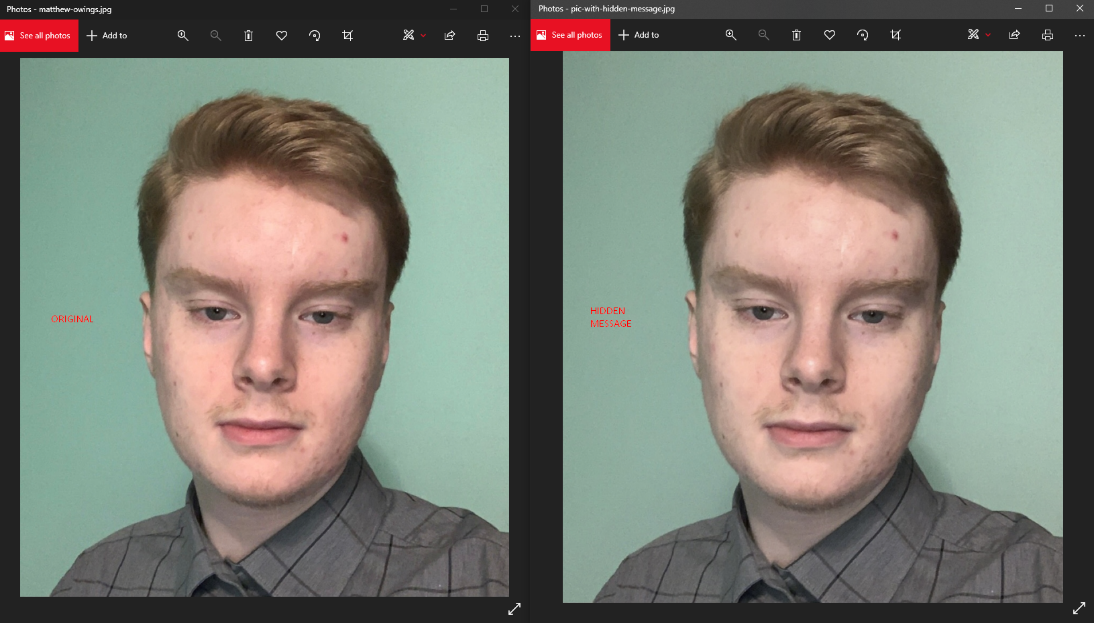
The -sf command with the previous command was set as pic-with-hidden-message.jpg, which is now created and visible in the same folder as the text file and the image file:



The pic-with-hidden-message.jpg file is now a copy of both the text file and the picture file, with very minimal differences between the two picture files. However, it is observed that the created picture with the hidden text is darker than the original, and the colors are not as vibrant and bright. The differences would be difficult to observe if the investigator did not have access to the original picture for comparison, as they would not know how the picture is supposed to look without the hidden message embedded.

**Differences between the Original Picture and the Hidden Message Picture**

As stated before, the key difference is that the hidden message picture file is somewhat darker and not as vibrant in color. The differences can be observed here:

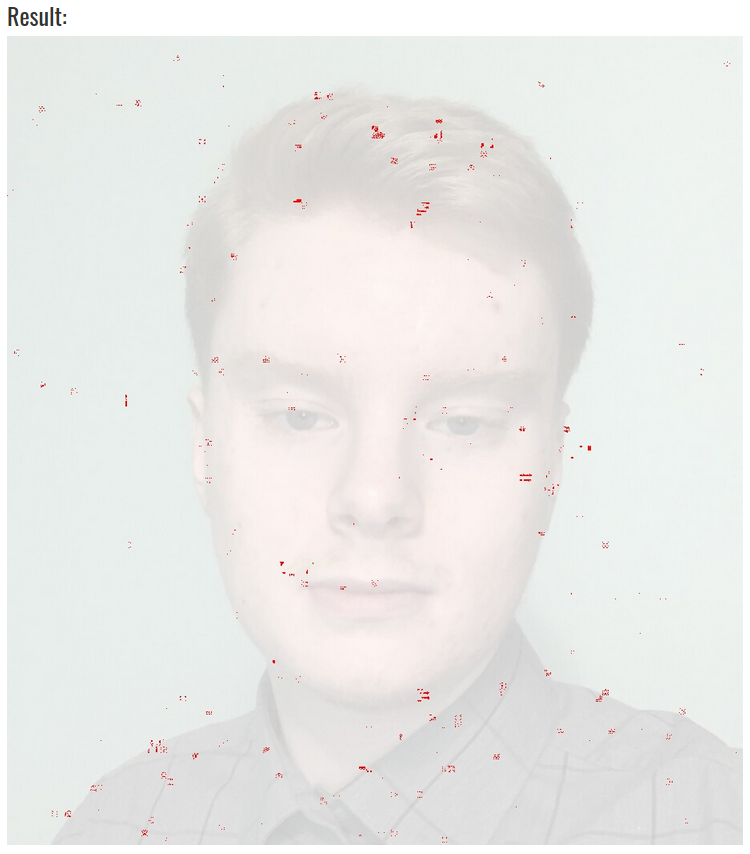


Original Hidden Message

The lighting between the original and the hidden message is much less vibrant, as the light reflecting off the skin and hair is much darker than the original. The background is a dimmer shade of green, and the shadows are of lower quality than the original. A level of compression affects the hidden message file, as it is a smaller file size, however this is due to the very small sized text file embedded into it. A text file with hundreds or thousands of words might make the hidden message picture larger in size than the original.

**Pixilation Differences**

More differences can be described on a level that computers can interpret. For example, the pixels between the two images vary slightly, extremely subtle differences that would be difficult to detect with the human eye. Luckily, computers can tell miniscule differences in coloration and the differences of individual pixels.



**Red is pixilation differences between original and hidden message picture**

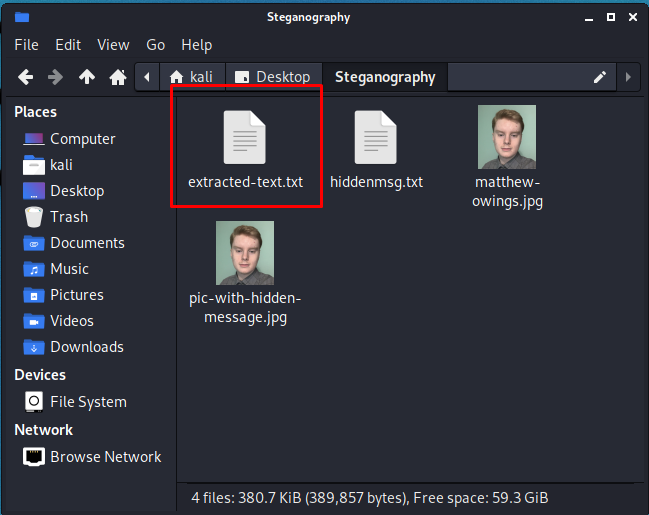
The original image is small, and therefore only a few dozen words shows a clear amount of red

**Extracting the Hidden Message**

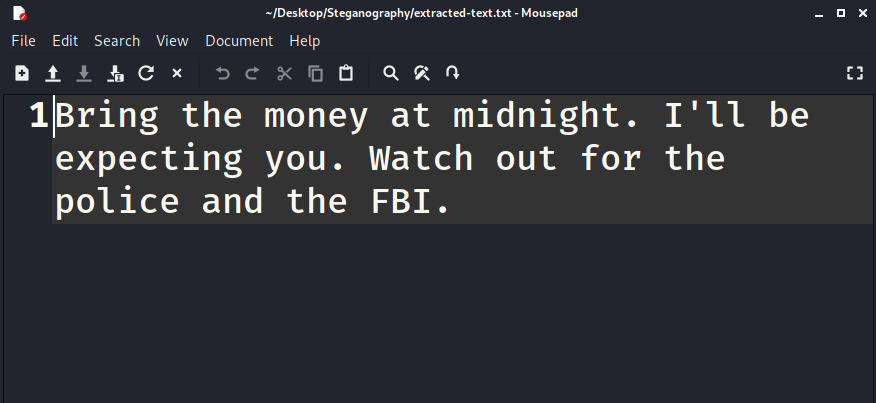
The main use of this technique is to send hidden messages without detection or without raising alarm, to appear innocent when in reality an embedded message is hidden within. This can be accomplished by using the picture with the hidden message embedded, by extracting the hidden message using Steghide.



Using the **steghide extract** command, selecting a picture with a hidden message, the password originally used to create the file, and using -xf to name the file that will store the extracted text, you can now view the hidden message hidden within the picture file.



Opening the extracted text file reveals the content originally embedded inside of the picture, as observed here:



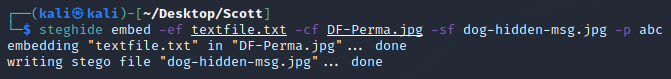
extracted-text.txt

**Second Example**

This example uses a case where the hidden message is 10,000 words long.

|  |  |
| --- | --- |
| textfile.txt | DF-Perma.jpg |
| Text  Description automatically generated | A cat with glowing eyes  Description automatically generated with low confidence |

Text File **🡪 Embedded Into 🡪** Picture File



Because the hidden message text file is 10,000 words long, the difference in pixilation is very apparent and noticeable. Results on next page.

**Background pattern

Description automatically generated**

**Red is pixilation differences between original and hidden message picture**

Because the hidden message file is so large, the red is clearly and vividly seen.

**Third Example**

|  |  |
| --- | --- |
| text.txt | trees.jpg |
| Graphical user interface  Description automatically generated | A picture containing tree  Description automatically generated |

Text File **🡪 Embedded Into 🡪** Picture File



Because the text file is extremely short, there is very minimal differentiation of pixels between the original and the hidden message picture. Results on next page

A close up of a person's face

Description automatically generated with low confidence

**Red is pixilation differences between original and hidden message picture**

Because the hidden message file is so small, very little red is seen.

**Conclusion**

In conclusion, it is extremely important for Digital Forensic Investigators to be aware that hidden messages can be transported in this fashion. To be able to allocate and find all important information regarding a legal case is essential as the information can potentially be incriminating (or prove innocence!) in a court of law. Some downsides of using Steghide is that encryption is present – hence the password used to create, which is also needed to extract – which can negatively affect the investigator’s ability to extract the information. Decrypting encrypted files can be a hassle for an investigator and can even be impossible if a strong password is used. Regardless, to be knowledgeable about Steghide is important for a Digital Forensics Investigator and to know the basic commands of embedding and extracting can be helpful in the industry.

**Works Cited**

Kali Linux. <https://www.kali.org/>

Online Image Comparison. <https://online-image-comparison.com/>

Steghide. <http://steghide.sourceforge.net/>