Master in Cybersecurity

Universidad Carlos III de Madrid

Advanced persistent threats and information leakage

Steganography Tool Kage no kotoba (影の言葉) - Shadow Words

Author

Sarvmetal

90n20

Sc4reCr0w

Fare9

Date May 31, 2020

Contents

1 Introduction **3**

2 Methodology **4**

2.1 Text Based Algorithm 4

2.2 Crawler 5

2.3 Image Based Algorithm 6

3 Results and Discussion **8**

3.1 Limitations 10

4 Conclusions **10**

A Use of the tool **11**

Abstract

Sending secure messages has always been a challenge for people, that's why different techniques to hide messages into innocent things has been developed. In this investigation an implementation of a steganography algorithm is done by using common text as news or tales which act as a key, using a word base stego and combining it with the message. To sum up a previous investigation in this field, it was added an image based stego to send the final file. With this implementation the message is hidden by two steganographic algorithms, a word and an image based. The final file can be sent through an insecure channel and if it is caught by another person, this person will not be able to read the original message even if he has some steganalysis tools.

1 Introduction

Steganography consists in methods and tools to hide information for preventing a third part can read it. Historically, the uses of steganography are huge, for example just by writing a message with special ink, making it invisible to other people but the person who receives it and knows how to recover the message [Johnson and Jajodia, 1998].

Nowadays the use of steganography has evolved to informatics, by the crescent risk of espionage in the cyberworld it is necessary to create a secure way to share messages for prevent attackers retrieve the original information and get only a bunch of useless characters.

There are different steganographic methods that can be used to hide data, all of them trying to accomplish the original steganography's goals remain undetectable and be robust enough to avoid being easily breakable [Cheddad et al., 2010]. The files that can be used to hide information are a lot, this investigation is focused on text and image-based steganography.

Text based steganography is not the preferred one at the moment of use something to hide a message because there is no redundancy unlike other steganographic methods such as images, sound and videos. Hiding information in texts result difficult because it must not present notable changes in the final output of the document but in its structure. In spite of its difficult, word based steganography is really helpful to hide something in plain sight, for instance using the newspaper to hide a message for another person, or using tales, a wiki, there are infinite options [Singh et al., 2009].

Image based steganography has different approaches, all of them pursuit the goal of hiding data and be imperceptible for the human eye in its result. There are several image hiding techniques, the most common is the LSB where the least significant bit is replaced with bits of the final message[Gutub et al., 2008]. In this study we are using other and less used technique which will be explained in detail in the Methodology section.

The name of the tool (影の言葉 or in romaji Kage no kotoba) was decided because of a possible meaning in English (Shadow words) this could be an allegory of the technique used in the tool, we want to thank Andrea Subirá for her help with the kanji and the possible meanings of this. The tool is publicly available in the github account of the group: `https://github.com/K0deless/APT-Stego-Assignment`

2 Methodology

For the steganographic tool that is developed we are using two different techniques to hide a message. Those techniques are explained of how the algorithm works for each of them. Also, we are using a crawler to automatize the texts acquisition for the algorithm.

2.1 Text Based Algorithm

This algorithm is based on the one purposed by [Agarwal, 2013] in section 3.3 which implemented text based algorithms by using paragraphs, the usage of paragraphs represent an interesting approach to hide information because public data can be used to hide a message, for instance the newspaper. In our case we are using kids tales and a wiki to get our texts. The algorithm proposed by Agarwal are:

Text-based algorithm for hiding a message:

1. Get a cover file.
2. Convert the input file to its binary equivalent (bin).
3. Read a bit (x) from the bin.
4. Read a word from the cover file and write it in stego file.
5. If start and end letter of the word is same, then read the next word of the cover file and write it in the stego file.
6. s = start letter of the word and e = end letter of the word.
7. If x = 0, write s in the stego key.
8. Else if x = 1, write e in the stego key.
9. Repeat steps 3 to 8 till the end of the bin file.

Text-based algorithm for retrieving the hidden message:

1. Read a character (c) from the stego key.

2. Read a word from the stego file.
3. If start and end letter of the word is same, then skip that word and read the next word from the stego file.
4. Get the start letter (s) and end letter (e) of the word.
5. If c = s, then bit b = 0.
6. Else if c = e, then bit b = 1.
7. Write b in a file.
8. Execute above steps repeatedly till the end of the stego key.
9. Convert the file into its character equivalent.

With this text-based algorithm, we had a good starter to make our implementation by developing some methods. For example, one method is used to clean the given text, allowed words will be only those that are 'alpha' words without numbers. It also removes non ascii characters, detects punctuation and one letter words. With this clean text we can proceed with the algorithm to hide the message.

For unhiding a key must be provided to know what information must be extracted, also it detects where this information is located and with the previous algorithm, we can get the hidden message.

2.2 Crawler

In order to automatize the texts acquisition, we developed a web crawler which became really handy at the moment of retrieving information. A web crawler is a web inspector which works with all kind of websites to retrieve information inside of them. Our web crawler searches for text and the image of a random web page inside the site.

We have two main websites as texts generators, for Spanish versions we are using a child tales web page [SL, 2020] and for English versions we make use of The Simpson characters wiki [Project, 2020].

To sum up there is a Russian version of the crawler which is developed but not implemented, this searches for random news inside [News, 2020]. The implementation was stopped due to problems in the codification of the text but is useful to continue the investigation in this field in the future.

The crawler search inside the website a random story or character and the main image, the text is used by the text-based algorithm and the image for the image-based algorithm reducing time in searching for information

2.3 Image Based Algorithm

The image-based algorithm uses a non-typical approach such as LSB or MSB, we use RGB pixels to hide data. For this we based our work Stepic [Domnitser, 2007] which is a python module that was handy at the moment of hiding information. Stepic works in the next way:

1. Pixels are read from left to right in sets of three at a time.
2. Each set contains three values red, green and blue (opacity is ignored).
3. Each group has nine values.
4. Each byte has 8 bits so each color can be modified by setting the least significant bit to zero or none.
5. These three pixels are able to store a byte with one color value left over.
6. The least significant bit of this left over value represents the end of data.
7. 0 = keep reading.
8. 1 = message is over.

Combining these algorithms and using the crawler we are capable of hide a message with strong steganographic techniques.

Hiding a message:

First, with the message to hide, the crawler gets the initial text and combines it with the message with the word-based stego algorithm to get the key. Then again the crawler takes part by getting the image corresponding to the text obtained and combining it with the key by the image-based stego, finally having an image with a key and a url in the metadata. If a person who is not the receiver has the image he will not be able to use steganalysis to get the message because the simple reason that after looking the metadata there is no strange information inside the image, the url can be confused for the origin of the file and that is how the message can be hidden in plain sight.

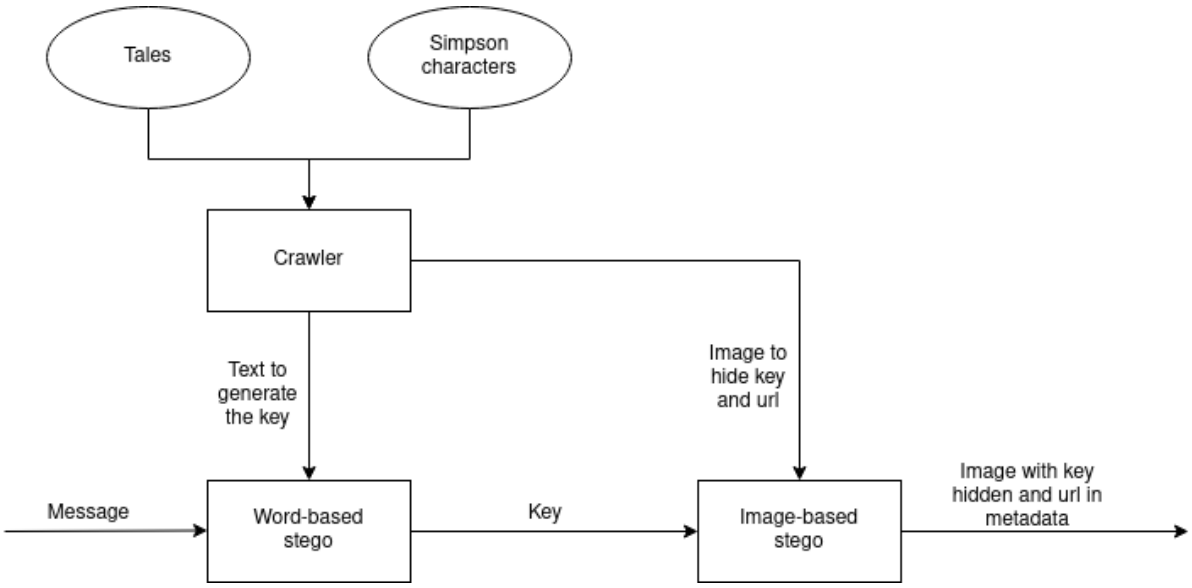

Figure 1: Arquitecture of hiding process

Unhiding a message:

For unhiding a message, the image with the key and the url in the metadata goes inside the image-based stego algorithm, so the key is passed to the word-based stego algorithm and the url is passed to the crawler, the crawler reads the url and gets the corresponding text, this text is passed to the word-based stego and combined with the key finally having the original message.

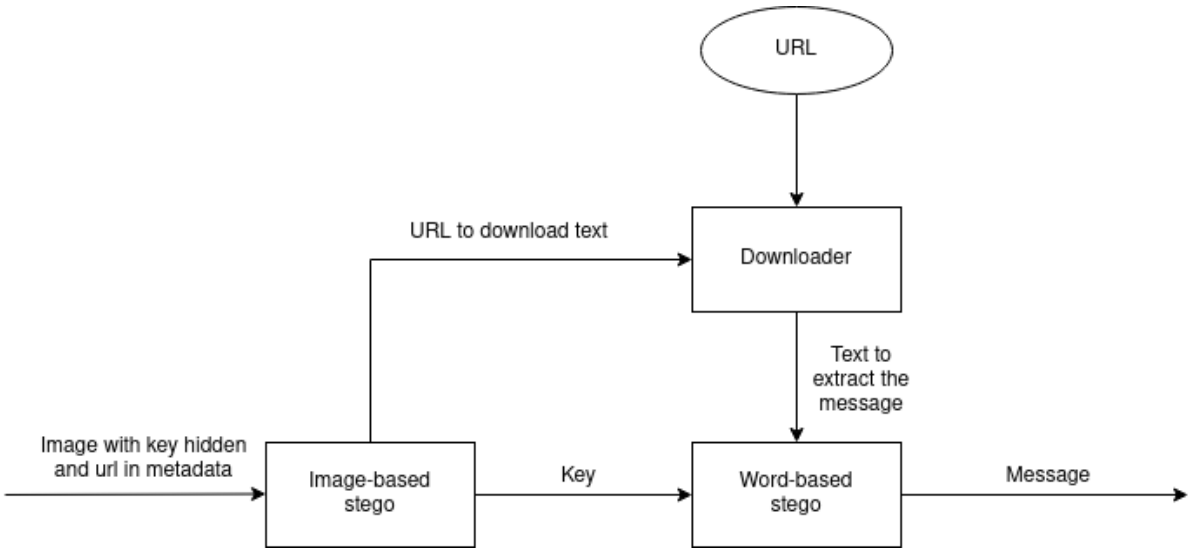

Figure 2: Arquitecture of unhiding process

3 Results and Discussion

A series of times have been written in table 1 the times are only taken for the process of hiding the message with the next and later hide the key in the message, and its extraction. The crawler work is not measured in the table. Beacuse of the size that it takes to generate the key, that it would be $key\_len = size\_of\_message \times 8$, and finally the size of data that has to be hidden in the image $key\_len \times 8$, the size of the input message is highly reduced, make it dependent of the downloaded picture mainly.

File Type	SHA1	File Size (bytes)	Hiding Time (Secs)	Unhiding Time (Secs)	Key Len (in characters)
txt	763da08cb39ecc27b459a023f45d8466c9899c45	22	0.010441	0.004734	176
txt	7deb176ffed8e3fc7e1140a40fc8b30376adb2dc	1001	0.075304	0.007702	8008
txt	021cda617a2a7893114ff8e142443aaf664f0a4d	3001	0.236355	0.012546	24008
txt	c0c85324cf1c9cf4217abde8187c6802545d978f	3501	0.229153	0.014975	28008

Table 1: Hide and unhide times for the algorithm.

Figure 3: Image with a hidden key

In the figure 3 we can see a picture with an embedded key after using our tool, a simple stegoanalysis technique was applied using the tool stegsolve [Caesum,], we found little evidences from the key at the beginning of the image, this can be seen in 4.

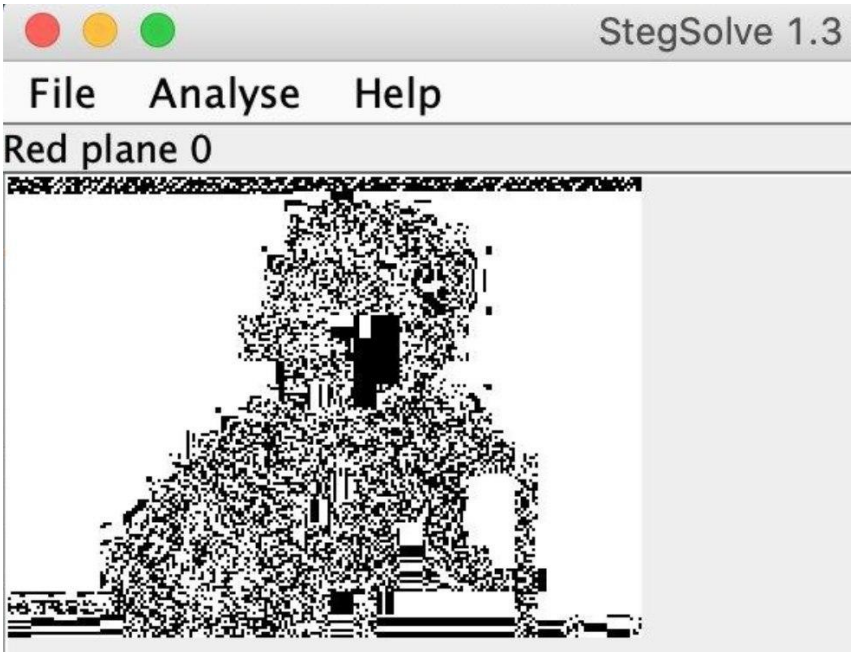

Figure 4: Stegsolve with the image of hidden message

3.1 Limitations

Many are the limitations that can be found in this tool, that mainly appears when giving big-sized files as inputs for hiding, because of dependent on the downloaded images we're don't know the exactly the size that will be downloaded so files that generate a long key cannot be hidden.

Another problem could be found in the text hiding algorithm used, if a *MitM* attack is performed and image recovered, if the attacker can extract in some way the key, and download the text (being trivial this step), it wouldn't take much to realize the approach used to recover the real message, for that reason a second layer of stego was applied to the algorithm.

4 Conclusions

We've shown a steganography way of improving the resilience of a tool joining two steganographic algorithms, even if an attacker is able to extract the url from the metadata and the key from the image file, the attacker would need to guess how the text is used and which algorithm was followed to be able to extract the real message.

One of the functionalities of the tool was the use of russian news that are written in *Cyrillic* but because of problems between ASCII and UNICODE representations we were not able to implement it properly, having issues to match the key with the text. Another limitation previously mentioned we are not supporting big files as input, but this could be solved with another possible improvement of support of new file formats that because of their size allows bigger size keys (for example "wav" files used for audio), this would affect the performance on hiding and unhiding time but it would help with bigger messages.

We did a simple command-line user interface in order to help users to use this tool together with a setup.py to install all the dependencies (still necessary to install libmagic in the system).

A Use of the tool

In this appendix we're going to show the command-line based interface with the two options.

In the figures we can see the simple process for hiding and unhiding a simple message, starting from the figure 5 which shows the simple interface with three options. if the user decides to hide a message like the one in 6 the option one will show the interface of 7.

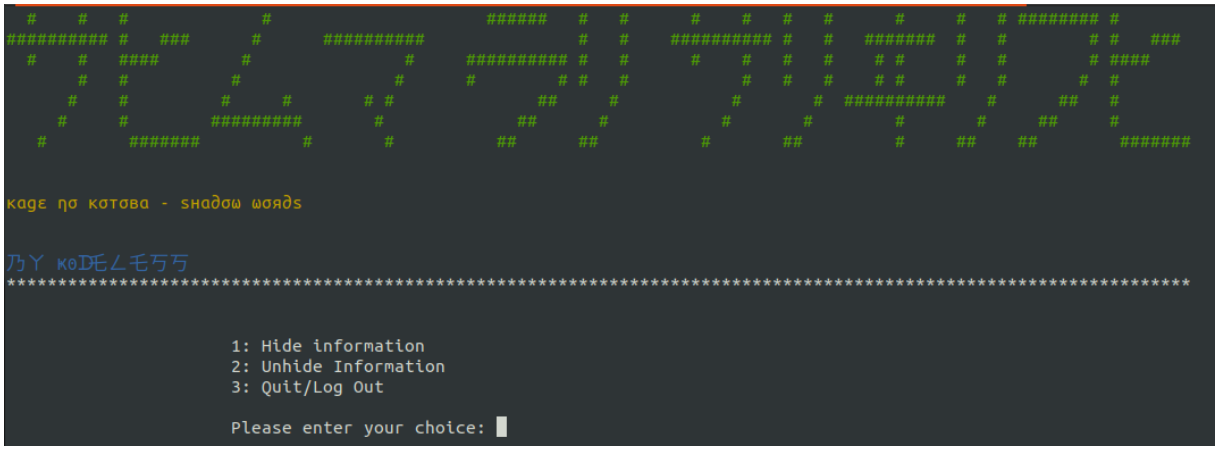

Figure 5: Command-line user interface

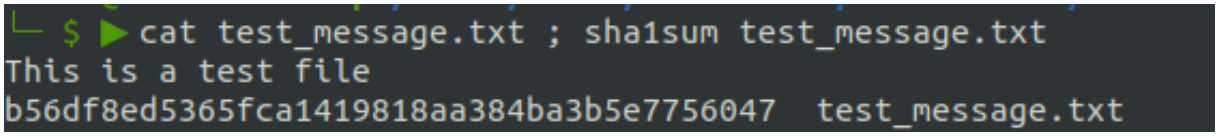

Figure 6: Simple text file to hide in an image

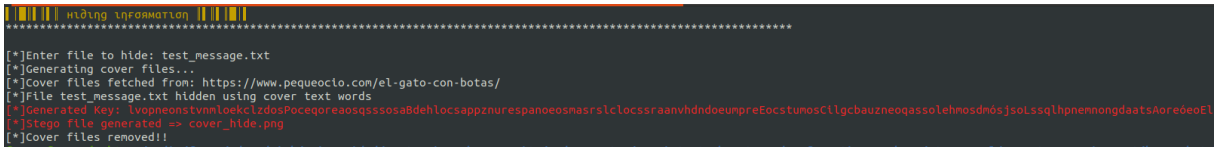

Figure 7: Hiding process with the tool

As the output of the process we get the image of the tale *Puss in Boots* which hides the key and maintain the url in its metadata 8.

Finally with the option two the user can unhide the message from the image giving the name of that image and a name for the output file (extention is decided using the MIME type),

Figure 8: Image where the key is hidden

we can see it in 9. Finally in 10 we can see that input and output messages have the same *SHA1* hashes.

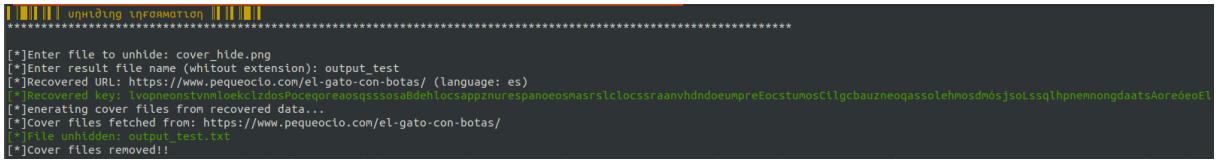

Figure 9: Unhiding process with the tool

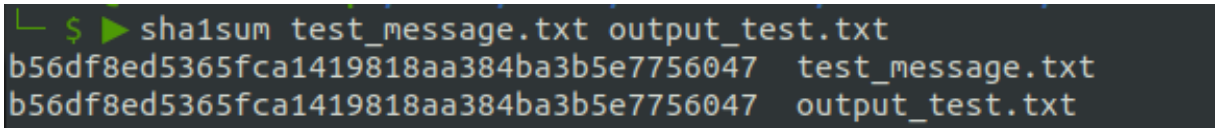

Figure 10: Hash SHA1 of input and output files

References

[Agarwal, 2013] Agarwal, M. (2013). Text steganographic approaches: a comparison. *arXiv preprint arXiv:1302.2718*.

[Caesum,] Caesum. Stegsolve. `http://www.caesum.com/handbook/Stegsolve.jar`. [Online; accessed 09-May-2020].

[Cheddad et al., 2010] Cheddad, A., Condell, J., Curran, K., and Mc Kevitt, P. (2010). Digital image steganography: Survey and analysis of current methods. *Signal processing*, 90(3):727–752.

[Domnitser, 2007] Domnitser, L. (2007). How stepic hides data in images. `https://domnit.org/stepic/doc/`. [Online; accessed 08-May-2020].

[Gutub et al., 2008] Gutub, A., Ankeer, M., Abu-Ghalioun, M., Shaheen, A., and Alvi, A. (2008). Pixel indicator high capacity technique for rgb image based steganography.

[Johnson and Jajodia, 1998] Johnson, N. F. and Jajodia, S. (1998). Exploring steganography: Seeing the unseen. *Computer*, 31(2):26–34.

[News, 2020] News, L. (2020). Russian news. `https://lenta.ru/rubrics/world/politic/`. [Online; accessed 08-May-2020].

[Project, 2020] Project, W. (2020). Simpson characters. `https://en.wikipedia.org/wiki/ListofTheSimpsonscharacters`. [Online; accessed 08-May-2020].

[Singh et al., 2009] Singh, H., Singh, P. K., and Saroha, K. (2009). A survey on text based steganography. In *Proceedings of the 3rd National Conference*, volume 3, pages 332–335. Bharati Vidyapeeth's Institute of Computer Applications and Management.

[SL, 2020] SL, K. M. (2020). Cuentos infantiles. `https://www.pequeocio.com/cuentos-infantiles/cuentos-clasicos/`. [Online; accessed 08-May-2020].